PERFORMANCE MONITORING OF PDP-11 COMPUTERS

by

Wolfgang B. Strigel

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

SCHOOL OF COMPUTER SCIENCE
McGILL UNIVERSITY
MONTREAL, P.Q., CANADA
August, 1976

ABSTRACT

Various system monitors and performance analysis methods have been surveyed. Software and hardware techniques are compared with reference to certain applications. An investigation of a general model for performance measurement tools has resulted in the development of HIMOS, a hybrid interactive monitor system. It is designed for a PDP-11 minicomputer installation. The software and hardware parts of the monitor are controlled from another PDP-11 computer. The system can work in sampling and event driven mode. Its ability to interfere actively with the performance of the measured system can be used for dynamic changes of the measurement session. The control system provides online and offline analysis of event reports. Monitor output is directed to a graphic display unit and can be concurrently recorded on tape or disk. The usage of the monitor in several application fields is illustrated through examples.

RESUME

Plusieurs moniteurs de système d'ordinateurs ainsi que des méthodes d'analyse de performance ont été étudiés. Des techniques de programmation et d'électronique ont été comparées. Un modèle général d'instrument de mesure de performance a été examiné et il en a résulté le développement de HIMOS, un système de moniteur hybride et interactif. Il est conçu pour une installation de mini-ordinateur PDP-11. Le programme et l'électronique du moniteur sont controlés par un autre PDP-11. Le système peut prendre des échantillonnages ou réagir sur des évènements. Il peut intervenir sur la performance de l'ordinateur observé, changeant ainsi la séance de mesure. Le système de controle permet l'analyse des rapports d'évènements en ligne ouverte ou fermée. Les résultats sont dirigés sur un écran cathodique et peuvent être enregistrés simultannément sur bande ou sur disque. L'emploi du moniteur est illustré par différents exemples.

ACKNOWLEDGEMENT

I would like to thank Professor N. Marovac for getting me interested in this project. Professor G.F.G. Patzer as my thesis supervisor provided guidance concerning the scope of the text. His encouragement and constructive criticism has helped me to bring this work to completion. I wish to thank Arnold Epstein for designing the hardware part of this project and also for the many fruitful discussions we have had together.

Finally I thank Ginette for her assistance in typing this thesis and for her patience and encouragement during the past year.

CONTENT

CHAPTER 1

Introduction to Performance Evaluation

1.1 Need for Evaluation

Prior to the description of the implementation and application of a practical tool for performance measurements in a computer system a discussion of the more general aspect of performance evaluation for which those tools are required will be presented.

The need for performance evaluation has not always been as apparent as it is today. First generation computer systems had a very simple structure and the central processor unit (CPU) was the dominant part of the installation. Peripheral devices such as magnetic tape drives, paper tape readers and punches were mainly used to maintain the traffic of program, data, and results between the system and the outside world. The devices were under direct control of the CPU. Computation and I/O were not overlapping because the CPU was dedicated at any time to one or the other activity. For scientific applications the power of a system was mainly dependent on the CPU speed. For commercial use the processing speed was generally limited by the performance of peripheral equipment.

The manager of a computer centre was able to charge his customers by the clock on the wall. His main concern was to insure the availability of his system. Availability can be defined as a measure of the likelihood that a system is operating properly at a given moment, or as the percentage

of time during which it is working properly.

With the advent of second and third generation systems, the complexity of information processing systems has grown considerably. For a human observer it became impossible to determine which job was executing at a certain time in a multiprogrammed environment. A scientific user could no longer expect the execution time of his program to decrease by factor two when the processing speed of the CPU was doubled. The interactions between hardware, system software, and user programs are far too complex as to allow this kind of simplified deduction.

It was not only for accounting purposes that a whole industry for performance measurement tools came into existence. Very soon it was felt that it was not enough just to write system and user software with the only goal to make it work. As the cost for maintaining computer systems went up one became more concerned about their efficient use. The performance of an existing installation had to be increased and criteria for future hard and software development had to be gathered. As a large variety of systems is offered on the market, there must be a method to compare the different products with regard to the intended application.

With the growing complexity of modern operating systems it is difficult to fully understand their behaviour. A system analyst has to know which parameters affect which aspect of the performance if he wants to tune a system. He may be confronted with side effects which he did not expect. His decisions have to be based on information that he can only

obtain through performance measurements.

1.2 Some Definitions

In order to define performance of information processing systems some terms have to be introduced first.

Throughput is the rate at which a given workload can be handled by the system. The workload can consist of payroll processing where throughput would measure the number of payroll statements per hour. If we consider a network node, this measure indicates the number of messages switched per hour. In a University environment we would be interested in the number of ALGOL or FORTRAN statements compiled per minute. A detailed discussion of workload characterization is given by Ferrari [1].

Depending on the application, another measure can be more important: Turnaround time is defined as the delay between the presentation of input to a system and the receipt of output from it [2, p.13]. Turnaround time is mostly used in connection with batch processing systems where it can specify the time span from putting cards into the hopper of a card reader and getting a complete listing on the line printer. Other papers include human factors such as operator handling time, think time, debugging time etc. into their definition (see for example [3, p.262]). But in this case we should speak about the time which is required for one test and debugging cycle.

Closely related to turnaround time is the term "response time". However this applies more to real time applications. For industrial process control response time could measure

the time elapsed between the arrival of a signal indicating the position of a work piece and the emission of some correcting signal from the computer. In a time sharing environment, response time gives the period between a user's request from the terminal and the moment at which the system is able to service this request.

It should be pointed out, that system overhead (the processing time which is spent in the operating system as opposed to the time spent in a user program) is not a fundamental measure of performance. The goal of performance improvement is not to decrease overhead but to increase throughput and to decrease response time. Overhead can be used as a descriptive parameter but it does not necessarily affect performance.

In the previous examples, the term "performance" has been used in different ways. Intuitively we can have a certain feeling for what it stands. However performance as an independent entity does not exist. It can be discussed only in the context of a specific application or a set of applications.

For example a high level language compiler may be judged by its compilation speed. This is an adequate measure in an educational environment whereas other users are much more concerned about the quality of the object code produced or the storage requirements for the compiler.

In a commercial computer center, performance is related to cost effectiveness. However this measure has to be interpreted carefully with regard to the individual application [3]. A fairly basic measure of performance,

giving the mean cost of delay to each job is suggested by Greenberger [4]. Generally we want some measure of effectiveness that is related to the ability to process jobs where jobs can be user programs or operating system tasks.

1.3 Performance Evaluation Techniques

One approach to performance evaluation is to describe the target system by means of an equivalent model. A model is an abstraction of the real system containing only the significant variables and relations. It is usually much simpler than the system it models. On one hand, the input to the model can consist of data about the actual workload of an existing computer installation. On the other hand information about the expected task volume of a projected computer system can be used. As this technique has to employ some simplifications in order to keep the volume of the model at a feasible size, the result can only give an approximation of the real performance. This method is widely used in the field of system design and research.

More precise results about the performance of a working system can be obtained from the analysis of performance measurements. Especially for the application in computing centres where observations about the actual system usage are necessary, this technique has known a growing popularity during the last ten years.

Calingaert [2] has given a very good review of the different performance evaluation techniques. The decision to use one or the other of those methods depends on the individual application.

1.3.1 Analytic Models

In analytic models, the relation between system variables and performance parameters is described by a set of mathematical equations. In accordance with real system behaviour it is assumed that events such as the arrival of a new job at the input appear in a stochastic manner. This means that variables can change their values in a random fashion but the range of values and the probability of the occurrence of a certain value is known.

We may for example know a given workload for the system but we cannot predict the sequence in which the jobs of this workload will arrive at the input. Let us consider a simple model of a batch processing system given by Graham [5,p.406]. In figure 1.1 new jobs which enter the system at the input are stored in a first-in, first-out (FIFO) queue. Time is divided into units called quanta, each of which is exactly Q seconds long. At the end of each quantum a new job can arrive at the tail of the queue. The job at the head of the queue is allocated to the processor where it is executed until completion. Once the execution is terminated it leaves the system at the output and the next job is shifted to the head of the queue. If the queue is empty, the processor remains idle.

Figure 1.1

To construct an analytical model, the job arrival time and the execution times for each job are expressed by probability distributions rather than in absolute terms. For example it can be assumed that a new job arrives in the system at the end of each quantum Q with probability pQ. The resulting job arrival distribution is a special case of the discrete Bernoulli distribution. The job's execution time can be chosen independently from a geometric distribution, s, assuming that each job requires an exact multiple of Q for processing. The probability that a job requires n quanta for execution is then expressed by s(n).

We can modify the previous model to study the round-robin scheduling policy which may be used in time-sharing systems (fig.1.2). In this case the processor is only allocated to one job for exactly one quantum of time. If the execution was not completed at the end of one quantum the job is fed back to the end of the queue. There it waits together with other incomplete or newly arrived jobs for its next time slice in the processor. Assuming that a job's execution time is exactly nQ, it will have travelled n times though the system before it has completed.

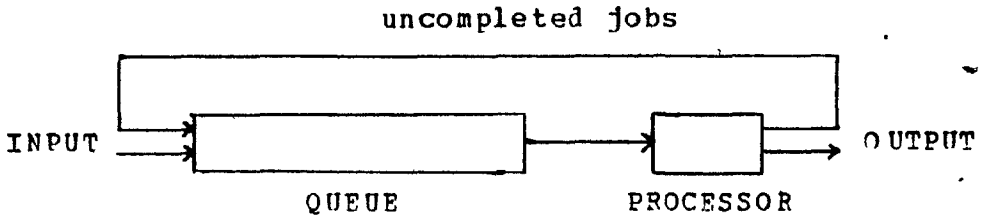

Figure 1.2

Another version where this model is extended by a second queue to buffer the incoming jobs is given by Blatney et. al. [6,p.412]. According to a selection algorithm, jobs are chosen from this auxiliary store to be placed at the end of the queue of the previous example which now serves as intermediate storage area for uncompleted jobs. Further levels of detail can easily be added to the model, but as is shown in the above mentioned paper, a deep analysis of the model can already become quite complex.

For the first two examples, Kleinrock [7] has derived equations for the expected total time that a job spends in the system under the above stated assumptions.

A more complex model which analyzes the system throughput as a function of input/output and processing overlap is described by Hellerman and Smith [8]. Several simplifying assumptions had to be adopted to keep the mathematical framework within practical limits. This shows already a severe drawback from the technique of analytical models. For complex systems, the relation between different variables may even not be expressable by mathematical functions.

Analytic models are most useful for the evaluation of subsystems of limited size (e.g. drum paging efficiency

[9]). As they need a certain degree of detail for their descriptive parameters they are in general used for evaluation of existing systems. At earlier stages of system development where this information is not available, the simulation technique is more appropriate.

1.3.2 Logical Models

While for analytical models, the system structure has to be translated into mathematical terms the logical model reflects this structure more directly. There are several different ways to express a system structure.

The directed graph model is very similar to a flowchart of some software. System states are represented by nodes and transitions from one state into another are shown as directed arcs. More details on this technique can be found in the literature [10,11].

Another interesting approach to logical modelling is in the use of Markovian models. Foley [13] has described such a model, using a discrete time semi-Markov chain where the time interval between successive state transitions is fixed. The chain is described by a transition probability matrix whose elements are the probabilities of entering a certain state from the current state of the system. The transition probabilities can either represent expected or estimated values, or they can be obtained from measurements on the executing target system.

Simulation which is the most frequently used technique among logical models will be discussed in more detail in the following section.

The first step in simulation is to define a dynamic model of the investigated computer system. The model consists of classes of entities, the attributes of these classes, a set of activities, and a set of events. A job that is to be executed by the system represents, for example, one class of entities. It can be described by attributes such as memory requirement, I/O activities, execution time etc. Activities are, for example, arithmetic computations or waiting for I/O completion. A job that enters the system causes an event; I/O interrupts can also be classified as an event. Events have no duration but in general they induce some activity.

The basic concept of simulation is time. Although simulation time is usually not equal to the real system time, the simulator controls events and activities with regard to its own time scale. Thus the simulator follows the sequence of events which indicate changes of the system state over time. Models for simulation of digital computer systems increment time in discrete intervals whose length correspond to the simulation time between consecutive events.

To simulate a system, a set of different experiments is conducted on the model. In other words, a sequence of different workloads is presented to the input of the model and results about the system performance on this input are obtained.

Another way to drive the model is to measure certain parameters such as job arrival time, distribution of page faults etc. in a real system and to use this data as input to the model. During simulation, other performance

parameters can be varied to find their optimal values. In this case we speak about trace driven modelling.

If required, the simulation results of a series of experiments can be formulated as mathematical equations describing the relationship between system variables and performance parameters. This abstraction can then be used to carry out further experiments more economically. This indicates that there is no absolute separation between analytic models and simulation models. For example, some subcomponents of a system can also be described analytically within a simulator. Of course, it is much easier to modify some aspects of the system in a simulation model just by replacing attribute definitions than by modifying the complex equations of the other model. In simulation no attempt is made to isolate relations between variables; observations are merely carried out on how they change their values with time.

In practice, the task of creating a simulation model is simplified by special purpose simulation languages such as GPSS (General Purpose Simulation System) or CSS (Computer System Simulator) [14]. Those languages provide facilities for varying parameters, for different experiments.

Especially at the planning stage of computer system development the simulation model represents the most convenient technique for performance evaluation. It is not only the most flexible of all models but it also can be constructed to any level of detail. Concurrency of activities can easily be modelled whereas analytic models are not able to express this feature which is becoming more

and more important in modern computing systems.

However all modelling techniques have common disadvantages. The obtained results can only be as good as the model. Analytical models are often oversimplified to reduce their complexity whereas simulation models are very expensive. Simplifications and assumptions in less known areas of the system can lead to undetected distortion of results. The validity of a model can be examined by comparing the results with those obtained on a real system but of course this is only possible if this system already exists. As the performance of a system is a function of the input, the experiments have to be designed at least as carefully as the model.

1.3.3 Performance Evaluation Programs

The techniques described in this paragraph apply to installed operational systems. They are mainly used to compare performance of different systems or to verify expected improvements of system behaviour. Executing the same assembly run on two different machines, we may observe different execution times just by looking at our watch. But even by timing the runs with a clock with a resolution of one microsecond would not tell us why machine A is faster than machine B. If we did the same test with another sample job we could obtain the opposite result, computer B being the faster one. In order to achieve more objective results, artificial workloads consisting of special programs or sequences of programs have been developed to evaluate the performance of computing systems.

The simplest technique in the class of performance evaluation programs is the instruction mix. The frequency of a machine instruction is derived from a dynamic trace under real workload. In the instruction mix, each instruction execution time is then weighted by a coefficient based on the frequency measurements. A table of instruction weights for a scientific and for a commercial mix are given in [15]. Knowing the instruction execution times of the target CPU a value representing the CPU performance can easily be calculated. The result depends on the dynamic trace which determines the relative importance of each instruction with regard to the overall performance.

The kernel technique uses another attempt to define a representative workload. It is composed of one or more assembler routines, each of which solves a specific computational problem, e.g. matrix inversion or evaluation of a polynomial. The set of problems that form a kernel is considered to be representative of the predominant application of the system.

The kernel technique has the advantage in that it is more CPU independent than the instruction mix as it tests performance on a task level rather than on the instruction level. However an objective comparison of different CPU's depends on the objective programming. The advantages and disadvantages of the last two techniques are discussed further in [2].

So far, the I/O characteristics of a system have not been evaluated. Benchmark problems include this feature in their workload definition. A popular benchmark problem for

commercial installations is the file updating program [16]. A processor reads a master file from disk or tape, updates it according to a detail file and produces a report file containing a record of each transaction performed.

Many attempts have been made to standardize benchmarks, but in general, they still reflect a special application. This technique may be valuable to gain insight into the expected performance of a system mainly used for a particular application environment but it cannot give a global measure of system performance.

Synthetic jobs as proposed by Buchholz [17] are more flexible than benchmarks. In those programs, written in high level languages, various resources may be parametrically changed to fit the characteristics of a real workload. This is done by several tasks which are executed cyclically. As every task requests for another resource (memory, CPU-time, I/O channels, etc.), the loop parameters can be adjusted to the system size and the workload to be simulated.

The techniques described so far evaluate system performance under the condition that the test program is the only one executed in the system. In this case the execution time of the job is used for evaluation or comparison. For timesharing systems we would rather want to measure the response time under an artificial workload. This means several users have to be simulated submitting different tasks concurrently to the system. A description of an internal driver simulating interactive users for the MULTICS time sharing system is outlined in [18]. Each user is

simulated by a process which can be considered an interactive benchmark consisting of a fixed sequence of commands and fixed think times.

Criticizing performance evaluation programs, one can say they rather define performance than evaluate it. Yet, if well understood, they can give some information about performance with regard to the yielded application. If they are used along with performance measurement techniques (which will be discussed in the next chapter), the obtained results can help to improve system performance under a well defined workload.

CHAPTER 2

Performance Measurement Techniques

Performance measurement or monitoring can be defined as the process of obtaining useful information on the performance of software and software controlled computer hardware. The earliest measurement tools were developed around 1956. They were able to supply information such as job execution time or core utilization which was obtained by core resident software monitors. In 1958 IBM developed the first laboratory version of an external hardware device for performance measurement. Tables A.1 and A.2 give a historical overview of some of the monitoring tools and their capabilities that have been implemented so far.

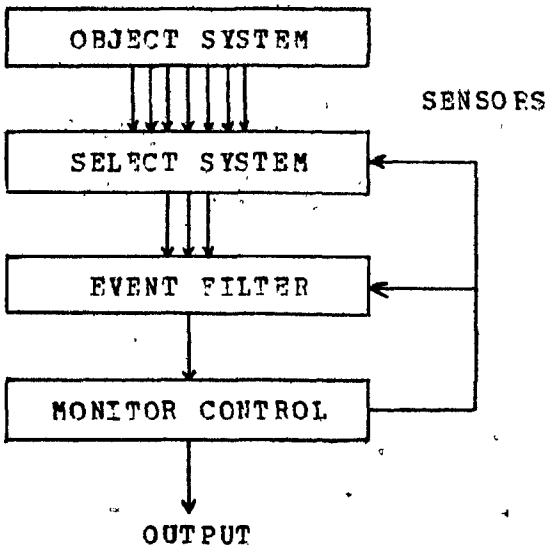

Figure 2.1: General model of a system monitor

The basic structure of a monitoring facility can be

described by the model in figure 2.1, regardless of hardware or software being used to extract the information. The computer system which is to be measured is called the object or host system. Sensors collect signals which indicate changes in the state of this system. These changes are also called events. The sensors can be implemented by software or hardware. Their implementation will be discussed in more detail in the following chapters. A hardware sensor for example can indicate if the wait light on the computer console is on or off. On the other hand, a software sensor can give an interrupt to the monitor whenever execution of a program passes over a predefined point in the code. The select system chooses desired observation points at times when their states are of interest. The measurements then proceed through the next stage where the flow of events is filtered so as to reject signals that are irrelevant for the particular monitoring session. Both selecting systems are supervised by the monitor control which receives the preprocessed signals and produces some sort of output.

Early monitoring devices in general recorded the collected data on magnetic tape. Programs were written to process these tapes after a monitoring session, reducing the large amount of data and creating statistical reports. The problem of interpreting monitor results is considered in [31]. Later monitors reduce and process the incoming data in real time besides producing a detailed record on mass storage. Graphic displays provide a convenient way of visualizing immediately system activities as they take place.

2.1 Objectives for Monitoring

Monitoring is used to obtain exact information about the various activities in a computing system while it is executing some tasks. One of these tasks is the operating system itself which normally is the most complex program of a given system. In general computer manufacturers offer a range of operating systems for different applications. However the detailed specifications which depend on the individual computing environment have to be entered in the form of parameters to the system. Many of these values, e.g. the maximum number of connected terminals or the number of tape and disk drives can be determined right at the beginning. However for tuning the operating system to achieve optimal execution a system monitor is almost indispensible.

2.1.1 Measurement of System Activities

Before optimizing system or user software we have to get a picture of the real system activities during execution. Measurements can give a utilization report for the different modules of an operating system. Based on this information, more frequently used modules can be made core resident while others can be stored on disk or drum. The statistical analysis of a disk usage report could show a high probability that module B is accessed immediately after module A on the same device. In this case seek times can be reduced by locating both files in the sequence A,B close together on the same device.

Many times device bottlenecks have slowed down a whole computer system. It can be very painful to pinpoint the source of troubles if no monitoring facilities are available. Using a monitor, an unexpected overload of an input/output channel can easily be detected just by sampling the channel activities during peak periods. This can also lead to predicting future facility requirements of the computing centre.

System measurement may also help to improve the coding of software. Cantrell and Ellison [19] talk about "performance bugs" by which they mean errors in evaluation or judgement of performance optimization. If the performance of some software is important it should be performance debugged by measurement and analysis. Inefficient parts of the program can be detected and recoded.

Another way to use monitors which have already been proved to be useful is in connection with simulation to produce what is called "trace driven modelling" [15]. A very detailed profile of the job stream, down to an almost microscopic level is observed and supplied as input to the simulation model.

Most operating systems are documented by numerous manuals and even if their descriptions are accurate (which is not always the case), system understanding is better facilitated by monitoring devices that show what really happens in the computer. A performance monitor with graphical output can help teach different aspects of operating systems. Such a device could also be used as feedback to the operators enabling them to make decisions.

The main objective of monitoring in this thesis is to offer a tool for operating system research and development. The monitor can be used to verify the efficiency of algorithms for paging or task scheduling. For the MULTICS operating system [20] this application has proved to be very useful [18]. Performance measurement can also be used for general software research such as program correctness proofs, automatic flowcharting, tracing, and debugging (see for example the SNUPER COMPUTER project [21]). Some of these possibilities are discussed further in chapter 6.

2.1.2 Prevention, Detection and Recovery from Failures

As mentioned in the last chapter, availability is an important factor in computer system performance. The conventional way to increase availability is to schedule certain maintenance periods during which the system is not accessible to users.

Permanent monitoring of the activity of system devices, including the CPU can help reduce maintenance time and yet insure high reliability of the system. Most operating systems are already able to capture fault conditions and produce a corresponding message. Monitoring facilities can go a step further, producing a precise report on all error conditions to show the maintenance personal where to concentrate their efforts. Further improvements are obtained when the monitor analyzes fault conditions and executes some of the diagnostic routines on the devices that reported failures. When this is done during idle time of the system, the time for maintenance can be reduced even

further.

In every computing system there are some resources which are vital to its operations. If the disk drive which contains operating system modules, part of main memory or even the CPU itself is working improperly then the system is "down". At installations that control important processes such as air traffic control or centrals for electric power switching, backup equipment for the most important resources is indispensible. In these systems a monitor could quickly detect an error condition and initiate the necessary actions to replace the defective device by its backup. In computer networks a node causing problems can be located and messages can automatically be routed through another path avoiding this node.

With the growing complexity of operating systems and multiprocessor hardware the reliability of computer installations becomes an increasing problem. Many times users are frustrated by system crashes which are due to implementation errors. For the system manager it is often a very difficult task to reconstruct and analyze the reasons for the malfunction. Dynamic verification and consistency checks by a measurement device could help analyze and even prevent such events. Plausibility checks can, for example, detect unreasonable overusage of resources which may be a hint for failure in the near future. This situation is typical for deadlocks which occur when the resource requirements of different tasks cannot be resolved. One approach for dynamic verification of operating system decisions is described by Fabry in [22].

2.2 Software Monitors

The earliest monitoring techniques developed used programs in core to monitor what the system was doing. These monitors are an integral part of the host system, i.e. the system that is to be analyzed. Thus, the monitor is both logically and physically a part of this system. The primary attribute of an internal software measurement technique is that it has access to and can selectively record system data stored in memory or CPU registers (depending on the computer architecture, device registers and buffers can be examined too).

When activated, the monitor interferes with the system operation to record and analyze values of interest. This leads to some considerations for the development of software monitors:

- The monitor must not alter any system variables except the ones that belong to the monitor itself.
- Resources used by the monitor, e.g. CPU time, memory space, I/O channels have to be well defined.
- The presence of the monitor must be logically invisible to the system.

Normally a software monitor is written in assembly language. The reasons for this are twofold: first, the execution time can be kept at a minimum and second, the access routines to system registers and memory locations have to be written in a low level language.

A survey of existing software monitors is given in appendix A, table A.1.

2.2.1 Sampling Technique

The basic idea of the sampling technique is to interrupt the normal flow of execution at periodic intervals where control of the system is passed to the measurement routine. The intervals can be determined by a clock or a combination of events in the system that meet some criterion. The best choice for the sampling rate would be a random time schedule which is known to be statistically independent of any natural execution pattern in the program to be analyzed. This avoids synchronization effects between the monitor and the sampled events which otherwise could amplify or zero out the recording of periodic events.

To obtain a compute time profile of a program, that program is loaded and run in its normal fashion without any modifications and without any need for prior knowledge of any of its characteristics. At the time of a sample, execution is interrupted and values of interest are recorded by the monitor. Some of the typical information gathered at this point is given in the following list:

CPU active

Location of next instruction to be executed

Instruction code

Location of data referenced by the instruction

Processor status word (PSW)

Status of channels, control units, and devices

I/O originators.

This information, recorded on tape along with the sample time is sometimes called a "system's snapshot". At the

completion of the run this tape can, for example, be sorted by interrupt location address and the resulting frequency distribution printed. The accuracy of the sampling technique is based on the central limit theorem [23]. The central limit theorem states that the accuracy achieved improves as the number of random samples increases. If 8000 samples are taken, the probability that we will be within 2% of the actual value is 99.99%. After approximately 26000 samples, there is no substantial improvement in accuracy.

2.2.2 Intercept Technique

Another approach to software monitoring is to place intercept points (or hooks) at strategic locations in system software and application programs. When the monitor is enabled and execution passes over such a point, control is passed to the measurement program which records interesting values just as in the sampling mode before passing control back to normal execution. Contrary to the method described in the last chapter the intercept technique is very operating system and machine dependent. This is mainly reflected in the way how the hooks can be placed. In systems with many different vectored interrupts where each of them is responsible for a small group of peripheral devices, the interrupts can be intercepted to obtain information about I/O activity. If additionally the operating system and user programs exchange information by means of some sort of supervisor calls which cause interrupts, there is a convenient way of monitoring this type of activity with the intercept method. In some systems

an entire class of interrupts may be matched into a unique address. Generally within the system architecture there is further information which identifies the specific nature of the interrupt.

To create intercept points for events other than interrupts, the source code has to be modified by insertion of special intercept code at interesting points of the program. This is for example used to monitor if certain locations such as branch entry points or subroutines are executed. Details of the implementation of sampling and intercept monitors are described in chapter 4.

With the intercept technique the requirement for software monitors to be logically invisible to the system when the monitor is not active is more difficult to meet. While there are no problems to deactivate hooks which capture interrupts just by resetting the original addresses in the interrupt vectors, this can cause more problems for hocks that were placed into the code. As changing the source code to remove those points is impractical, a way has to be found to change the hook command into a NOP (no operation) command. In bigger systems like the IBM 370 series, this can be done by the EXECUTE instruction which has the ability to define a hook instruction either to be an interrupt creating one or to be just a NOP. Microprogrammable machines can support the same feature in a similar way just by redefining a micro instruction. For computers from Digital Equipment (PDP11 family) the problem has to be solved in a different way which will be shown in chapter 4.

Another technique which is for example used at the McGill

computing centre should be mentioned. The software monitor (SLACMON, [51]) is invoked whenever the CPU is idle in order to use up all "spare" time. During those periods, a counter is incremented at a fixed rate. The counter value gives then a measure of the CPU load.

2.2.3 Comparison of Sampling and Intercept Methods

In a commercial computing environment where only the overall system behaviour has to be controlled, the sampling monitor would be a sufficient tool. It is relatively easy to handle and the degree of detail can be adjusted simply by changing the sample frequency. On a CPU with an average instruction time of one micro second and a sample rate of one milli second, statistically relevant information can be obtained after only 10 seconds with a maximum of 1000 instructions between two successive samples. In practice this means that a system profile can be updated in 10 second intervals. However the choice of the sampling frequency is not a trivial one and depends upon the yielded information. Another problem with sampling monitors is their low degree of flexibility. The set of values collected per sample during one session is mostly invariable. The selection will have to be general enough to give all the required information but it has to be detailed enough to minimize redundancy in this information. The number of values recorded per sample cannot be too large to keep the ratio of recording time to sampling period resonably low and thus limit the overhead.

For applications where specific problems in a system have

to be analyzed and where each occurrence of certain events has to be recorded, the intercept monitor is the more appropriate tool. At initialisation a set of events can be defined along with an individual description of the information to be recorded for each of the events. This feature increases the flexibility of that monitor type and decreases the amount of redundant data recorded.

Assuming we want to monitor disk arm movement to find an optimal arrangement of files on that disk we can specify that only values like physical disk address, seek time and originator identification be recorded. If, in the same run, paging activity is to be monitored, the events indicating page faults etc. can produce a completely different set of values. This distinction cannot be made in sampling monitors where we would have to record the union of both sets for every event which of course increases the impact that the monitor imposes on the system.

In general, the main advantage of the intercept technique is the ability to concentrate on a clearly defined range of system activities and to generate a reproducible report.

The monitor overhead in the sampling technique is readily defined by the sampling rate and the number of values recorded per sample. This is not so easy in the case of the intercept technique. Actually the overhead is a function of the event occurrence. Additionally, an exhaustive overall system profile based on a statistical evaluation cannot be obtained.

As long as events can refer to interrupts, the installation of an intercept monitor is no problem: the corresponding

interrupt vectors just have to be modified to point to the monitor routine. But the installation, activation, and deactivation of hooks in the software code can cause some headaches. As mentioned in chapter 2.2.2 this depends on the system layout. A quite unproblematic solution for computers in the PDP 11 family is shown in chapter 4.

All software monitors require host system resources such as CPU time, core memory, buffer space, I/O channel time, and mass storage. This of course is a disadvantage as the monitor's operation may distort the normal system activities and thus may lead to a situation which is known as the Heissenberg uncertainty. Therefore the impact of the monitor imposed on the measured system has to be investigated carefully before the results of a monitoring session can be analyzed.

2.3 Hardware Monitors

2.3.1 Architecture of Hardware Monitors

In the previous section we have seen that software monitors use resources of the host system. Among other reasons this was one of the drawbacks that led to the development of hardware measurement techniques. Table A.2 shows some of the hardware monitors developed during the last 18 years. A good description of the history of hardware measurement techniques is given in [24].

Hardware monitors collect electrical signals that indicate the state of the host system. Probes are attached to test

points or pins on the back panel of the computer system. These high impedance probes are completely passive to the host system. In other words, they do not affect the performance of the computer in any way. Hardware monitors usually operate in sampling mode rather than in the event driven fashion. In the second case an event is caused by the occurrence of an electrical signal at a probe point.

Once signals are presented to the monitor, they are generally made available at some form of logic plugboard. This is used to accomplish first level data reduction on the signals to determine states of coexistence and mutual exclusiveness. If, for example, the overlap of CPU and I/O channel activity has to be measured, we would "OR" the active signals of all channels and combine this output via a logical "AND" with the CPU active signal. More examples for the use of logical data reduction in hardware monitors are listed in [25].

The resultant signals are directed to several counters or recorded together with a time stamp on a storage media. Some of the more recent hardware monitors perform some real time evaluation of the incoming signals and update the result periodically on a display unit.

The method by which the signals are evaluated and recorded constitutes the major difference between early monitors and those available today. It also defines the type of measurement which the monitor can perform and thus is a means of classifying hardware monitors into major subgroups.

2.3.2 Summary Devices

The earliest hardware monitors developed by IBM were summary type monitors (see appendix A, table A.2). They recorded signals either by timing their duration or counting the number of times they occurred. The counter contents was periodically read out and written on tape. Thus, the summary type monitor could give information about the percentage of time a signal was active over the monitoring period. When counting the signals it gave the frequency of occurrence for the measured events.

It is important to note that any dynamic properties of the system activities were lost in the accumulated information. However for computing systems in the early 1960's this monitor type was good enough to give a clue to what was happening in the computer. At many installations this technique is still used to produce overall system utilization reports.

2.3.3 Dynamic Monitors

With the increasing complexity of computer systems the need for more detailed activity reports became imperative. Information was required for input to various simulation programs whose accuracy depended on the actual distribution of I/O and computing activities. The summary type monitor would indicate the CPU being much more utilized than expected but the cause of the inefficiency remained unknown. One of the early attempts to overcome this problem by using a dynamic monitor is given in [26]. A dynamic trace of the

event "Jump instruction executed" was used to follow the execution flow and to detect loops.

Instead of merely accumulating the events, dynamic monitors provide a trace in time of events as they occur. The major problem in implementing these devices is that the computer is generally able to change its state faster than the monitor can record it. There are mainly two solutions to the problem: either the monitor's resolution is kept at a frequency at which the storage facilities are capable of registering the events or as in [27] the monitor is given the ability to stop the host system at certain times to follow up with the recording activity. However, the interruption of the host computer is against the principle of noninterference as stated above for the basic hardware monitors. Furthermore it can hardly be tolerated in commercial installations. Different buffering techniques have been implemented to increase the resolution of dynamic monitors without losing too much information.

2.3.4 Modes of Operation

Modern hardware monitors combine dynamic as well as summarizing features to provide information about different aspects of the host system activities. Corresponding to the measurements made by the early monitors there are basically three modes of operation:

1. Event mapping (E-MAP) mode: In this mode the counters used by the early summary monitors are replaced by words in the monitor's random access memory. The address of a memory word is given by the code of the event. The

contents of each memory cell represents either the accumulated count of the events or the total time a particular event is active. In the first case, the content of the according memory cell is incremented by one every time the event occurs. In the second case, the monitored signals are sampled at regular intervals as specified by an internal clock.

2. Time mapping (T-MAP) mode: While an input signal is active, a counter is incremented at a rate determined by an internal clock. When the specified signal becomes inactive a memory location whose address is given by the counter value is incremented by one. The monitor's memory then represents a frequency distribution for the duration of the observed signal. In practise this could be used to obtain the seek time distribution of a disk unit.

3. Store mode (S-MODE): The store mode corresponds to the trace measurements described in chapter 2.3.3. The event codes are simply written in consecutive memory locations as they occur or at the frequency determined by the internal sampling clock. The memory is divided into buffer zones which are periodically dumped on tape to free the storage space. The choice of one of the monitor modes depends on the information requirements. Several examples how these features of hardware monitors can be used are presented by Epstein [28].

2.4 Hybrid Monitors

While hardware monitors can virtually measure any significant physical machine event, they can capture logical events only in some cases. Protect key activity on IBM 360 systems for example is strictly a logical function of the operating system, but the register that contains the code for the protect key of the contolling task is in a fixed machine location and can therefore be monitored.

A number of extremely important system characteristics are never reflected directly in the hardware. Thus there is no efficient way for a hardware monitor to determine how much memory is occupied by user programs, how many jobs are currently in the job queue, the reasons why many jobs may wait for resource allocation before being initiated or how long a particular executive routine may reside in core.

The most evident advantage of hardware monitors is that they do not affect the host system in its performance. They are capable of sensing a wide variety of hardware and even some software events but they are limited in their ability to detect the stimulus for a set of responses. This cause and effect relationship can be better established by software monitors which take advantage of the fact that they reside within the host system. They have access to virtually any parameter which is controlled by the operating system software. Especially in multiprogramming systems, the latter technique can be used to investigate the system's task table and correlate the location of an event with the identification of the originating program. Physical machine

events, however, are beyond the reach of software monitors. In table 4.3 three monitoring techniques are compared with regard to the most important criteria.

Fortunately, the capabilities of hardware and software monitors complement each other and thus suggest a solution to this problem. The resulting type of systems monitor is called a hybrid monitor. The underlying premise of the hybrid monitor is that a hardware monitoring device is not invisible to the operating system, but instead, it is treated as an "intelligent" peripheral device.

This concept has been implemented at the MIT Lincoln Laboratory [29] where sensors and event counters are allocated to user programs under the control of software. The user can define which activities are to be sensed and the hardware device interrupts the user program as soon as it recognizes one of these events. The device has been used to derive an online histogram of the subroutine utilization, to investigate virtual memory performance of a single program and other similar tasks.

The distinquishing characteristic of hybrid monitors is that software can be used to determine activities which are to be measured by the hardware device. This implies that the commands to a hardware monitor are no longer given by buttons on the front pannel but they are dynamically supplied and altered by a software program. In table 3 the different monitoring techniques are compared.

For example, suppose the memory activity of a certain program is to be monitored. Soft hooks can be established in the supervisor's dispatching module to detect when the

program is about to be executed. At this point the software monitor can take control and transmit to the hardware the base address and size of the program. Since memory activity is to be monitored it also directs the hardware to select the memory address register as the source for the measurements.

One of the first hybrid monitors for systems with virtual memory is the DSP-1 [30] which is currently used at the McGill computer centre. It reduces the monitor overhead in the above mentioned example in the following way: A map of memory areas that have to be monitored is established by software and sent to the device which then continously samples the content of the program counter (PC) and compares this value to the ones in the map. As soon as the PC enters one of the specified memory areas, the monitor starts to record predefined system variables.

As we have seen, the monitoring devices have become more sophisticated with increasing complexity of computer systems. While the first monitors purely recorded everything they were able to measure, the more advanced techniques concentrate on the problem of screening the sensed signals to eliminate redundant or unwanted information prior to the recording. Plugboard logic and event filters were improved but at the end it became obvious that many problems could be solved if a second processor were used to perform the preliminary data reduction. This has led to the last generation of performance measurement tools which will be the subject of discussion in the

following chapters.

CHAPTER 3

Monitoring with External Processors

3.1 A General Instrumentation System

The DSP-1 mentioned in the last chapter is an elaborate tool for performance measurements. The sensory system which had to be programmed for different logical combinations on bulky plugboards in earlier systems is now controlled by commands on a console. However, the most important feature of the DSP-1 is the possibility to transmit those commands in remote mode from the host system, i.e. software residing in the monitored computer can define the preliminary data reduction and other parameters that control the monitor's operation.

As it has been shown, hybrid monitors combine the advantages of hard and software monitors but they are not flexible enough to change the monitoring mode dynamically as a function of the incoming events or to perform more complex real time data reduction. Due to the storage requirements of the map and store modes, memories were added to monitor devices. Their internal control structure required an increasing amount of hardware to supervise the data flow. The DSP-1 monitor incorporates already a special purpose processor to take care of these functions.

In this chapter one further step will be taken and a more flexible instrumentation system will be discussed which includes its own processor and software residing in the

monitor's memory. This model is, at the same time, the basis on which a performance monitor was developed and which will be the subject of the remaining chapters.

3.1.1 The General Model

Logically the monitor consists of three parts as shown in figure 3.1. The software monitor, operating in sampling or event driven mode, resides within the host system where it extracts information which is available in the host memory. The hardware monitor collects electrical signals with its sensory system and performs first level data reduction. Both units send the preprocessed information to the control or supervisor system. On the other hand, the control system has the ability to send data back to either of the monitors, interrupt their operation via control lines and also interrupt the host system itself. This last point is usually not found in commercial monitor devices. However, it has proven to be very useful for applications in research and development. Finally, after having processed the data, the control unit communicates the result over the I/O bus to some peripherals.

In the following three paragraphs we will take a closer look at each of the components. A functional drawing of the configuration is given in appendix A, figure A.1.

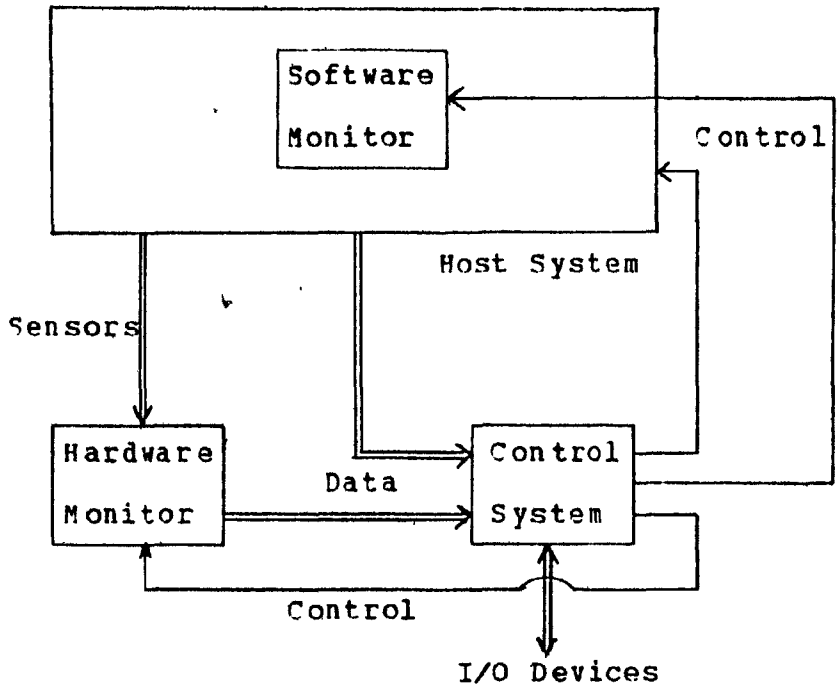

Figure 3.1: General Monitor Model

3.1.2 Software Monitor

This part of the performance measurement system resides in the host memory. For normal applications, the software monitor runs in the intercept mode because the data gathered in sampling mode can be extracted more efficiently by the hardware sensors.

As early as in 1967, as part of the "Snuper Computer" project [21], Estrin developed an idea for hybrid monitors which has been included in this system. Instead of inserting softhooks at strategically important points of programs which are to be monitored, the software is analyzed at compile time and the relative program counter value for those points is determined. The technique used by Estrin is

very similar to those used for program correctness proofs [35] (see also Chapter 5). At load time the absolute PC values of the event locations are determined and passed on to the sensory system which reports an event as soon as one of those locations appears in the PC during execution.

Of course, the method is not easy to implement as it requires changes in the operating system software, namely the compilers and loaders. It has the advantage that the system degradation is placed into utility tasks and the program to be observed can run without any interference. It is also a typical example for a situation where the distinction between operating system and software monitor begins to disappear.

When the monitor, running in intercept mode, has recognized an event it will collect relevant data and send it together with the event identification to the control CPU. This transmission can be done over a standard I/O channel of the host system. The same channel is used in the other direction to send information about the operating mode to the monitor. This has to include interrupt vector locations, a breakdown of events to be captured per interrupt if the vectors are used for multiple functions, and details about the required data per event.

3.1.3 The Sensory System

High impendance probes capture signals that indicate the state of the host CPU and its devices. In practice, a set of probe points is amplified in so called concentrators before the signals arrive at the sensory system. More about

hardware probe points can be found in the literature [32,33]. The signals then pass through logic circuits where the first level data reduction takes place. The combinatorial logic can take many forms. Generally speaking some logical relationship between the signals is determined. Instead of using a plugboard to specify which signals have to be compared and which logical function is to be used, the combinations are defined by the control processor. Additionally this remote technique makes quick changes during a monitoring session possible. The resulting signals are then presented at the input of buffer registers, counters or comparators. In the first case the data is ready to be transmitted to the control CPU or directly stored into memory by means of a DMA channel. Otherwise signals are either counted directly or used to open a gate to increment the counter with a fixed frequency during the signal duration. These two techniques correspond to the STORE or E-MAP mode and the the T-MAP mode respectively.

Although the control CPU maintains its own time base it can be useful to send the host time to the monitor. Provided both systems increment time at the same rate, only an initial synchronization is necessary.

Apart from checking the input signals, the use of comparators leads to a new data reduction technique called sequencing. The first monitored value is compared to the contents of a series of registers. The index of the first matching register is remembered. In case there is no match at all, the value is ignored, otherwise the next value is compared with the register following the remembered one and

if the test is successful again, the index of the currently matching register is stored. This can be carried out a certain number of times depending on the size of the sequencer or infinitely if the comparator series is logically closed to form a loop. In a linear arrangement the result of this procedure indicates if a predefined sequence of values occured in the host system or not. In the case of a closed sequence, a counter can keep track of how many times a loop was executed.

This idea can be carried further to analyze more complex sequences with alternative branches by arranging the comparator registers in the form of a finite state automaton. For any state there is a predefined set of events which can move the automaton in one of the successor states. If the received signal does not belong to the set or if the automaton enters a trap state the sequence of signals is rejected as not belonging to the language (in terms of the theory of formal languages) which was defined by the automaton.

Finally the circuit can have a "learning" ability if the first sequence of n values defines the comparators and is checked against the next n values. In the case of a match, a loop is detected and the comparator values can be transmitted to the control processor to identify the loop or else the second n values are used to redefine the comparator values.

However these attractive features require complex hardware and for economical reasons their implementation will have to wait for further innovations in hardware technology.

3.1.4 The Control Processor

The choice of the control CPU is no easy task. The CPU should have the following properties:

1. Execution speed similar to the host system
2. Fast I/O channels
3. A minimum of 32K words of memory

In general the control system is of a magnitude smaller than the one to be measured. This means that, for example, an IBM 360 installation should be monitored by a mini computer or if a mini computer is to be supervised, it would require a micrc computer in the control system. This, however, creates difficulties regarding the first point stated above. In most cases the speed of a CPU decreases with the size. The reason for this lies in the architecture of the central processor control. Large scale systems have many functional units working in parallel. Medium and especially small scale CPU's have a sequential microprogrammed controller which makes them easier to implement but slower in execution. This problem is inherent in all processor controlled monitors. If the control processor is not especially designed for this application, the solution can only be found in the sensory system itself and in alternating or rotating data buffers.

The main bottleneck of monitors lies in the transmission of data between the sensory system and the control processor. This is why most hardware monitors perform some preliminary data reduction with combinatorial logic. This measure is still insufficient for systems which produce events at an

exessive rate. For this reason it is desirable to have a set of high speed registers in the sensory system which can be used as buffers when data bursts arrive at the input. If this is still insufficient there is only one way out of it, namely to interrupt the host system until the monitor has caught up. The interrupt should be directed to the software monitor which can inhibit any processor activity until it gets the signal to release the CPU over the control/host communication channel. At installations where this cannot be tolerated, at least a data overrun indication is recorded along with the data.

A second speed problem can occur at the output channel when data is recorded on mass storage. However, the situation becomes only serious when the monitor is running in store mode where every event is recorded without any data reducing transformation. The monitor speed would then be limited by the speed of the recording device. In all other modes this problem is not very likely to occur as first and second level data reduction as well as buffering techniques should decrease the data rate so that it can be handled easily.

The memory requirement stated in the third point is due to several considerations. First, memory has to hold the decision and evaluation software. Second, it has to provide enough space for buffer zones and third, it has to have room for the E-Map, T-MAP or STORE modes. Some of the information that reaches the monitor does not have to be processed immediately. In this case the operation speed can be accelerated when the processor and its I/O channels are bypassed and the data is stored via direct memory access

(DMA).

Apart from managing the data flow from the monitor units the control processor has several other tasks. It has to perform second level data reduction and on-line evaluation and present the results mostly in statistical form to the user. CRT displays or line printers can be used as output devices. At the same time a more detailed record of system events has to be buffered in the control memory to be dumped periodically on a mass storage device, e.g. a magnetic tape. Finally the processor has to accept commands from a keyboard. This enables the user to set the monitoring mode, the display mode, and other parameters to specify the operations of the whole performance measurement system. A desirable feature would be to have not only a manufacturer supplied set of software routines but a "monitoring operating system" which incorporates the basic tasks of data acquisition. In the CPM-X monitor [34], the most frequently used routines for data reduction are programmed in firmware, which again helps to increase the processing speed of the CPU. The user should have the possibility to write his own evaluation software on top of the utility routines. A considerable amount of flexibility would be added to the system and programs to track down special problems or to modify the evaluation techniques could be written according to individual requirements.

3.2 Application of the System

The interactive nature of this system offers a range of new applications. The most difficult part of a monitoring session is to set up the probe points, define the logic combinations in the sensory system and to specify the events to be intercepted by the software monitor. Finally the secondary data reduction procedure in the control processor has to be chosen. Experience has shown that the first setup never exactly gives the desired information. Several test and correction cycles have to be carried out before the result is satisfactory. With the monitoring system developed in this chapter, the adjustments can be done interactively from the monitor's keyboard.

Let us consider a typical example. During system profile measurements a significant operating system overhead is observed in one of the utility programs. Without relocating probe points or inserting new hooks in the software, the monitor can supply the identification of the troublesome routine as soon as it is loaded from disk to core by associating the physical disk address with the name of the accessed module, assuming that the system device directory is available for the monitor. Knowing the size of the identified routine a sampling run can be set up to observe PC activity for the corresponding core segment. The start of the sampling session can be defined as the event of loading the routine the next time into core. The resulting execution profile will then show which parts of the routine are responsable for the low performance. Commands for this

tracing procedure can be given in a simple language (possibly similar to BASIC) and the complete process can be supervised from the monitor's console.

The centralized control over the measurement system also enables non specialists to handle the monitor. Operators can ask for continuous feedback for their decisions at the host system console and can immediately see if the system load is optimal. The real time display showing the different system activities is a valuable tool for teaching purposes. The pedagogical value of a dynamic display is obviously better than static explanations on the blackboard.

Finally the interactive monitor is an efficient tool for system development and research. In the experimental computing environment the host system can be interrupted or slowed down. While events have to be otherwise captured "on the fly" and the results are displayed in summary form or evaluated offline, in this application a detailed online analysis is possible. The monitor can be regarded as a sophisticated debugging routine. The hardware probes allow an instruction trace much faster than any conventional trace routine. If the monitor has the ability to alter the state of the host system, e.g. changing the PC, stack pointers etc., all the requirements for a powerful debugging tool are met.

CHAPTER 4

HIMOS - Hybrid Interactive Monitor System

4.1 Purpose of HIMOS

One of the initial objectives of the design of a monitoring system for the mini computer installation in the School of Computer Science at McGill University was to provide a method to increase the availability of the computers. It was planned to develop a technique for automatic loading and execution of diagnostic programs as soon as a system failure was detected. During the analysis of the problem the idea was abandoned for several reasons. First it turned out that monitoring was a prerequisite to accomplish the original goal. Although there was a multitude of literature on computer system instrumentation, the subject seemed promising for further research. Second it was found that problems other than availability, yet related to monitoring, were more pressing. At the same time the project offered the possibility to test the real time capability of the mini computer used to control the monitoring process.

The resulting objective was to design an interactive hybrid monitor, controlled by a dedicated processor as outlined in figure A.1. Because the monitored system was underutilized most of the time, the improvement of performance characteristics was not one of the prime considerations although with the increasing workload, this may become an

important point in the future. But, as mentioned before, monitors can be used in many other applications. During the last year mainly three directions emerged for research on the mini computer installation:

1. Multiprocessor systems: This subject is specially attractive as there are already two mainframes available in the department. In the long run it is planned to extend the system by several micro computers to build a network for research purposes which could be supervised by the monitor.
2. Operating systems: Apart from the system software required for point one, there are projects on paging algorithms and task scheduling routines in progress. Recently two new time sharing systems (UNIX [36] and concurrent PASCAL [37]) were acquired and the monitor should be of help in understanding and extending those systems which are not very well documented.
3. Programming languages: This field offers a wide range of monitor applications. It is planned that one of the projects uses the monitor to produce flowcharts dynamically according to the real execution. This topic is closely related to the problem of program correctness proofs for which the monitor offers some practical possibilities (see ch. 5).

These three points can only give a rough overview of the possible applications. One side effect of the monitoring system for example showed up during the development of the monitor itself. The currently used operating system, RT-11 (supplied by Digital Equipment [38]), employs several

techniques which were not described in the accompanying manuals. Already the very first version of the monitor detected several unknown scheduling techniques and the knowledge of their existence induced new implementation ideas.

Finally the monitor will assist in teaching several hardware and software courses. For a course in introduction to computer architecture, hardware probes can be connected to the host system and dynamic changes of the machine state can be demonstrated under reduced execution speed on the graphic display. For students in software oriented courses, the real time creation of the execution profile for a sample program will aid in understanding the relation between static and dynamic program structures. The computer will no longer appear as the "mystic black box" and terms like program counter or interrupt will become more realistic once they are visualized.

4.2 Computing Environment

4.2.1 System Configuration

The computing facilities in the School of Computer Science are subject to continuous improvements in soft and hardware. Therefore references to the system configuration are as of March 1976.

The complete computing equipment was manufactured by Digital Equipment Corporation (D.E.C.). There are mainly two independent systems. One is based on a PDP 11/20 processor with 28 - K words of core memory, one 2.5 Megabyte

disk drive, two 288 Kilobyte DEC-tape drives and a Tektronix 4012 graphic display which serves as a console device. This installation is used as the control system for HIMOS.

The host system is built around a PDP 11/45 CPU with 40 K words of core memory, a memory management unit, one floating point processor, one 2.5 Megabyte disk drive and two system consoles consisting of a 300 baud hardcopy terminal (DEC-Writer) and a 1200 baud video display (Beehive).

The two systems are physically located close together. They are connected via two communication channels. One consists of two interconnected serial asynchronous interfaces operating at approximately 10 Kilobaud; the second link operates by means of two parallel asynchronous 16 bit interfaces at an average transmission rate of 40,000 words per second. A third parallel interface establishes the communication between the PDP 11/20 and the hardware monitor. All interfaces were originally designed by DEC but were modified to fit the special needs for high reliability and increased data transfer rates.

The complete system configuration is shown in the appendix, figure A.2.

4.2.2 PDP 11 Family

In order to fully understand some of the monitoring techniques applied in HIMOS, it is necessary to have a basic knowledge of the PDP 11 system architecture. This paragraph will only highlight the features which are relevant to monitor implementation, especially with regard to the PDP 11/45 structure as this computer was chosen for the host

system. Further details about the PDP 11 structure, the corresponding processor handbooks should be consulted ([39] and [40]). A detailed description of peripherals and interfaces can be found in [41].

The PDP 11/45 is a 16 bit parallel processor which can run in kernel, supervisor or user mode. It can perform DMA data transfer as well as CPU controlled interrupt or flag driven I/O. The interrupt system has a 4 level priority scheme and the CPU can be on one of 8 different priority levels. The 16 general purpose registers (GPR's) are functionally divided into two sets of 6 registers each, 3 stackpointers (SP), each for one of the three different operating modes and the program counter (PC). The address space ranges from 0 to 32 K words and can be extended up to 128 K words under memory management. The upper 4 K of address space are reserved for I/O devices and the lower 256 words are used for interrupt vectors. The CPU interprets 78 instructions which can be used in 12 different addressing modes.

One of the most important features of the PDP 11 design is the UNIBUS concept: All system components and peripherals are connected to a single bidirectional asynchronous bus, called the UNIBUS. Since all system elements, including the CPU, communicate with each other in identical fashion via the UNIBUS, no distinction can be made between memory access or input/output operations. Although this technique facilitates programming it caused some problems for the monitor implementation.

The HIMOS intercept technique is based on the interrupt mechanism of PDP 11 computers. Interrupts can be caused by

peripheral devices or the CPU itself. A hardware interrupt occurs when a device wishes to indicate that a specific condition has occured. The relative priority of the request is determined by the processor's priority and the level at which the request is made. If the device's priority is higher than the current processor level the following actions take place:

1. The processor relinquishes control of the bus passing it over to the requesting device.
2. The device sends a unique memory address as base of the interrupt vector. The first word of the vector contains a pointer to the interrupt service routine, while the second location holds the new processor status word (PS).
3. The CPU stores the current PS and PC in temporary registers.
4. The new PC and PS are taken from the interrupt vector. The old PS and PC are pushed in that order onto the current stack and the service routine is initiated.

Upon the RTI command (return from interrupt) these actions are executed in the reverse order.

The following commands in the PDP 11 instruction set can cause an interrupt sequence:

BPT, IOT (Breakpoint and I/O traps):

The first command is used for debugging techniques where breakpoints can be created by substituting the original instruction by the BPT code. IOT can be used to schedule I/O actions.

EMT, TRAP (Emulator trap and trap):

Those two instructions are used for several dispatching techniques. The high byte contains the operation code while the low byte contains a user defined constant in the range from 0 to 377 (octal). In RT-11 the EMT instruction is used as supervisor call.

PIRQ (Programmed interrupt request):

This instruction offers another way for task dispatching. The request is treated in a similar way as device interrupts because the priority level can be specified.

Finally there are 3 important vector locations reserved for processor fault conditions. As soon as one of the conditions occurs, the processor traps to the corresponding vector.

Trap to 4:

Most novice programmers are confronted with this error condition. It indicates the processor "time out" signal. The reasons can be access to nonexisting memory or device locations and odd address in a word instruction.

Trap to 10:

The processor has encountered an illegal operation code. This situation is typical for attemps to execute data instead of program code.

Trap to 24:

Whenever the power drops beyond a certain limit the processor traps to this location to initialize the power fail routine. When power is restored a trap to

the same location will start the power-up routine.

4.2.3 The RT-11 Operating System

Originally it was intended to monitor the PDP 11/45 under the UNIX time sharing system. However, for technical reasons the UNIX system generation was delayed. At the beginning of 1976 the implementation of HIMOS was started under RT-11 (Version 02B-SJ) which is, at this time, the only available operating system.

RT-11 is a single user operating system for computers of the PDP 11 series. On the PDP 11/45 and 11/70 the CPU is permanently kept in kernel mode while executing RT-11 or user tasks scheduled by the system. It requires a minimum of 8 K words of core and can handle a maximum of 32 K words address space. It is designed for program development and real time applications. It supports PAL-11 and MACRO, the PDP 11 assembler languages, FORTRAN IV and BASIC. Several utility programs are available such as the text editor (EDIT), and on line debugging aid (ODT), a file handler (PIP), several device handlers and a batch utility. It can accept device interrupts and service the two most important error traps (trap to 4 and 10).

In the following some of the aspects of RT-11 which are important for monitoring will be discussed. Further information can be obtained from the manuals [38, 42].

When RT-11 is booted it determines the physically available core (the maximum for RT-11 is 28 K words) and loads itself on top of the memory. The resident system part needs approximately 4 K words. The trap vectors and the most

important device vectors are then initialized. When device handlers are brought into core, RT-11 locates them just below the system. Normally, user programs are loaded at address 1000 (this and following addresses and memory values are given in octal if not stated otherwise). The resulting memory layout is given in figure 4.1.

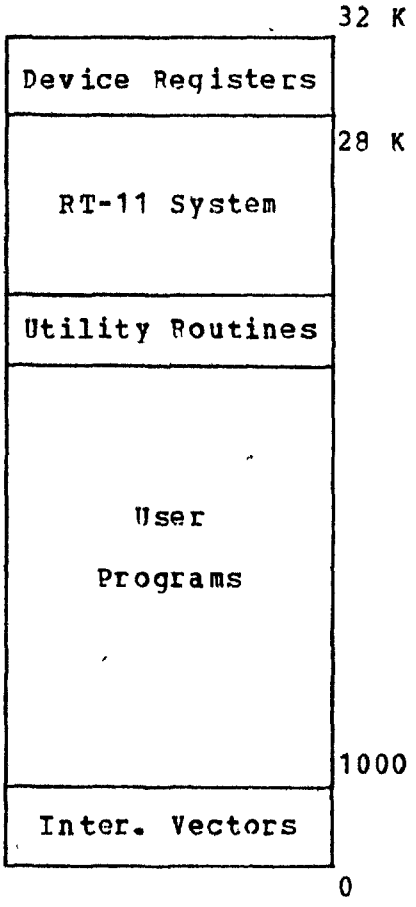

Figure 4.1: Memory Layout under RT-11

An executible program image is called a SAV-module. It always represents core locations from zero to the high limit of the code. This enables a programmer to define certain vector values statically in his program. To make sure that no vital system vectors are erased when a user program is

loaded, the system resets some vectors after a program load. The vectors to be restored are indicated in a protection table within the system. This technique is obviously not the most elegant way of system protection. On the other hand, RT-11 cannot take advantage of a hardware memory protection as this is not a standard item and is only supplied with the memory management unit. Therefore RT-11 can only protect itself at program load time and also against keyboard commands from the console (like Examine or Deposit). This is not the case during run time. A user program has access to any existing address within 32 K. The subject of memory protection has considerably influenced the implementation of HIMOS.

The communication between the operating system and a user program as well as internal RT-11 scheduling is performed via EMT-calls. The use of the EMT instruction is therefore exclusively dedicated to system actions. A large macro library contains all system utility routines, for example, READ and WRITE requests. The macros use EMT-calls to transmit a command via software interrupts to RT-11. For example the EXIT macro call at the dynamic end of a program is translated into an "EMT 350" instruction where the number 350 serves as a function code for RT-11.

Disk and tape storage is blocked into records of 256 (decimal) words. On RK05 cartridge disks this corresponds to one sector. At the beginning of a tape or disk a file directory is located. It contains starting address and lengths of the files on the device. Files are stored sequentially but after several file manipulations there may

be gaps between individual files.

RT-11 has no file protection. Each user can access any information on a device. Only the directory is protected, provided system macros for file manipulation are used.

4.3 Concept of HIMOS

The HIMOS system was specially developed for the system configuration described above. However it can be used to monitor any other computer within the PDP 11 family, provided a second CPU for the control system is available. System dependency is introduced by the software monitor which is designed to operate under RT-11. The hardware monitor part is more flexible although some of its logic has to meet special requirements of the PDP 11/45 architecture.

The PDP 11/45 was chosen to be the host system which is monitored under control of the 11/20. This choice is due to the fact that the bigger 11/45 has more components of interest such as the floating point processor and the memory management unit. Originally it was planned to execute the control software on the recently announced LSI-11 micro computer (manufactured by DEC) but it was not available on time. However, the control program can easily be transferred at any time since the instruction set is compatible.

HIMOS was implemented in three steps. A first version of the software monitor allowed some initial experiments to become familiar with various problems. This version ran like the early software monitors on the host system which was used at the same time to analyze events and to output

the results. Second, the hardware monitor was designed and third the control system was implemented on the 11/20. This approach led to a modular design of the instrumentation system. Each monitor can operate on its own. Only the supervisor CPU is necessary to receive information and to control the operating modes.

4.3.1 Software Monitor Functions

The software monitor (S-MON) is core resident in the host system where it is protected by RT-11. When a specified event occurs it is captured by S-MON which accumulates pertinent information in a buffer before transmitting the complete event record to the control system (CS). The event record is headed by the event code (EC) and can virtually contain an unlimited number of additional words indicating the state of the host system at the time of the event. In a commercial environment S-MON would release the CPU after having gathered the information and would transfer the record concurrently with the continued execution of the interrupted task. In HIMOS however, the record is transmitted and a response from the control system has to be received before S-MON releases the CPU. Although this technique increases the monitor overhead it is necessary to allow interactive response from the supervisor. If a special event is detected in the control system, the monitor mode may be changed, the hardware monitor initialized etc, before the host system changes its state. This feature opens a wide field of applications for HIMOS which are not possible in most other monitor

implementations. Any hard or software event may be dynamically declared as a special event. The list of these EC's is kept in the control system.

From its design the software monitor uses exclusively the intercept technique. As will be shown later, even the sampling mode is implemented using the same technique. At initialization time the vectors for all interrupts to be recorded are replaced by pointers into the monitor program. The original vector values are saved in S-MON. The choice of the vectors can be interpreted as the first level of data reduction (as a matter of fact it is data elimination). At the same time a list of items has to be specified which are to be recorded upon the occurrence of an event. Every individual event can have its own set containing items of interest. This gives the second level of data reduction.

At this point it is helpful to follow an event through the software monitor (see fig. 4.2). Let us assume "trap to 10" is specified as an event. As soon as the CPU encounters an illegal instruction code, the execution flow is interrupted and control is passed to the location pointed at by memory address 10. As a result the PC will contain the entry point of S-MON. The event code is computed and used in the main dispatcher as an index to call the corresponding event handler (EH). There, the system state, according to a specification list, is stored into a buffer and the complete record is sent to the supervisor system. Assuming that the event analysis did not indicate a special event, the supervisor simply sends an acknowledgement and S-MON will transfer control back to RT-11 as indicated by the original

contents of the interrupt vector.

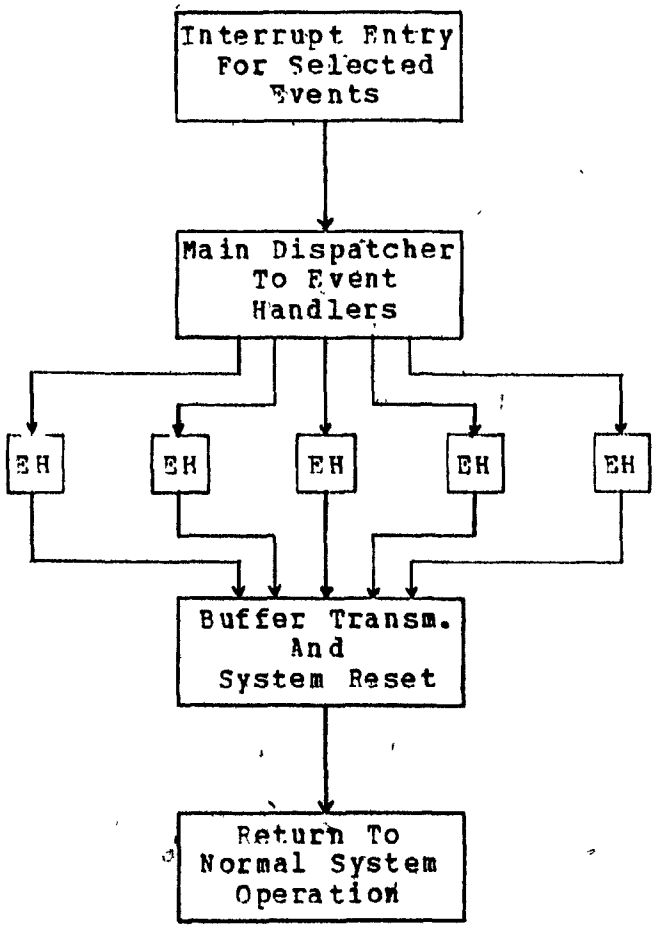

Figure 4.2: Functional Diagram of S-MON

The procedure is completely transparent to the user and to RT-11, as long as the event is acknowledged by the control system. If a special event was detected the control CPU has the possibility to change the system state in the host in which case HIMOS interferes activly with the host performance.

As mentioned before, the sampling mode is handled in a similar way: the 11/45 clock interrupt is intercepted and the event is reported to the supervisor. To transfer

control back to the point where a task was interrupted the necessary information is taken from the system stack. This solution offers the possibility to combine interrupt and sampling techniques during the same monitor session. For example device and CPU activity can be monitored while every second a sample interrupt is created. This can be used to monitor if the system is still in working order or if, for example, the CPU was stopped by a HALT instruction.

4.3.2 Hardware Monitor Design

The design of a hardware device to monitor PDP 11 computers requires some special considerations. Mainly the way how I/O activity is handled causes practical problems. Any I/O operation is performed on the UNIBUS and except for interrupts it cannot be distinguished from a regular memory access. Further difficulties are due to the fact that there are not many meaningful probe points available at the CPU backplane. To monitor, for example, the PS or PC requires complex and dangerous manipulation on the CPU boards. It is desirable that the computer design for next generation systems take that problem into account and bring out sufficient probe points for easy access. However some of the important signals in PDP 11 CPU's are brought to a plug which is used for maintenance purposes. This has simplified access to a subset of important probe points.

A general design of the hardware monitor (H-MON) was given to the Department of Electrical Engeneering at McGill where it should have been implemented as part of a course project. Due to problems with the delivery of components the

implementation has not been possible so far. Based on a detailed design together with the engeneering drawings, the function of the device will be discussed in a later chapter. Possible applications of the hardware monitor will be considered although results of practical experiments are not available.

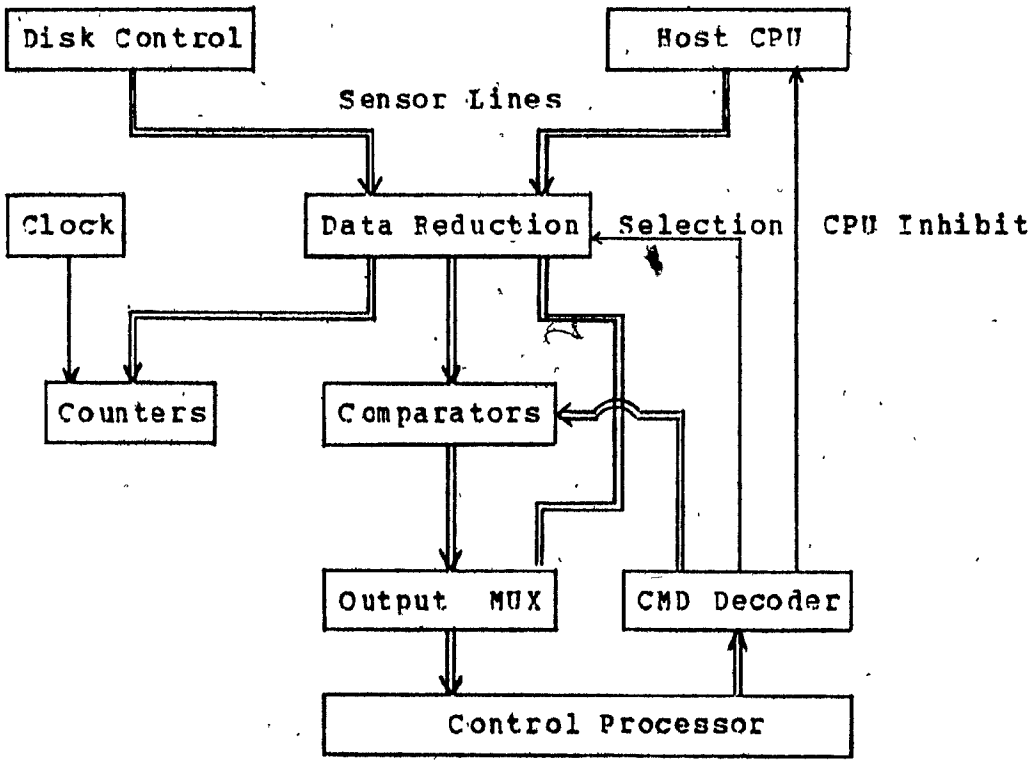

Figure 4.3: Hardware Monitor

According to the initial design, the hardware monitor has several groups of sensors which can be connected to probe points of the host CPU and the disk controller (see fig.4.3). The signals are then supplied to a data reduction logic which is controlled by the supervisor system. Hardware counters can accumulate signals at high speed or they can be used to time events with the internal frequency

generator. The device operates in different modes which can be changed dynamically by the control system. Results are transmitted over a modified parallel interface for further processing.

4.3.3 Monitor Control and Output

The supervisor system, implemented on the PDP 11/20, has complete control over all monitoring actions. Its command interpreter accepts a large set of instructions which can be grouped into three parts:

a. Display and recording mode

b. S-MON control

c. H-MON control.

(Any references to software features for the control of the hardware monitor imply that the relevant task entries exist in the supervisor system, but the corresponding code has been replaced by a NOP instruction.) To facilitate the definition of a monitoring session, command (CMD) files for each of the three groups can be created for repeated use.

The control system (CS) can interrupt and redefine the operation of S-MON and H-MON at any time either as a function of received data or according to a keyboard command. It analyses event records and stores them by means of an alternating buffering mechanism. Buffers can periodically be dumped on tape or disk. At the same time the CS displays the events in two different ways. For detailed information the complete record for every event is written out in numeric form. As this happens in real time and because the transmission to the terminal operates at

2,400 baud, the host activity has to be slowed down according to the event density and the number of items to be listed per event. In the graphic display mode where the occurrence of an event updates a histogram in real time, the monitor overhead is practically limited by the record transmission between the host and the CS.

Further tasks of the CS are to maintain the system time relative to the monitor initialization and to start and stop the monitor session in a remote way.

It may be instructive to follow up the example of an event occurrence given in section 4.3.1 with regard to the supervisor system (see fig. 4.4). As soon as the event record is received at the S-MON interface it passes through the event filter. The event code (EC) is compared to the list of special events. If the list is empty or if the EC does not match, the filter will send an acknowledgement to S-MON to release the host CPU. The system control will then direct the record to the output handler if real time output is specified. Otherwise the CS operation is finished. A buffer overflow is automatically resolved in the receiver program by switching the buffers and the filled buffer is recorded on mass storage at a later time when the CS is idle. If numerical or graphical display was specified the output handler will perform the necessary data transformation.

In case the event filter finds the EC in its list, the special event sequence is invoked. The display will give a numeric output of the event record (no matter in which mode it was running) and indicate that this is a special event.

During that time the host is suspended and S-MON is in receiving mode, waiting for further instructions, the operator can react in different ways. He may release the host immediately and thus terminate the special event sequence. He also can change the H-MON or S-MON mode according to new requirements for the succeeding analysis, or he decides to terminate the monitor session.

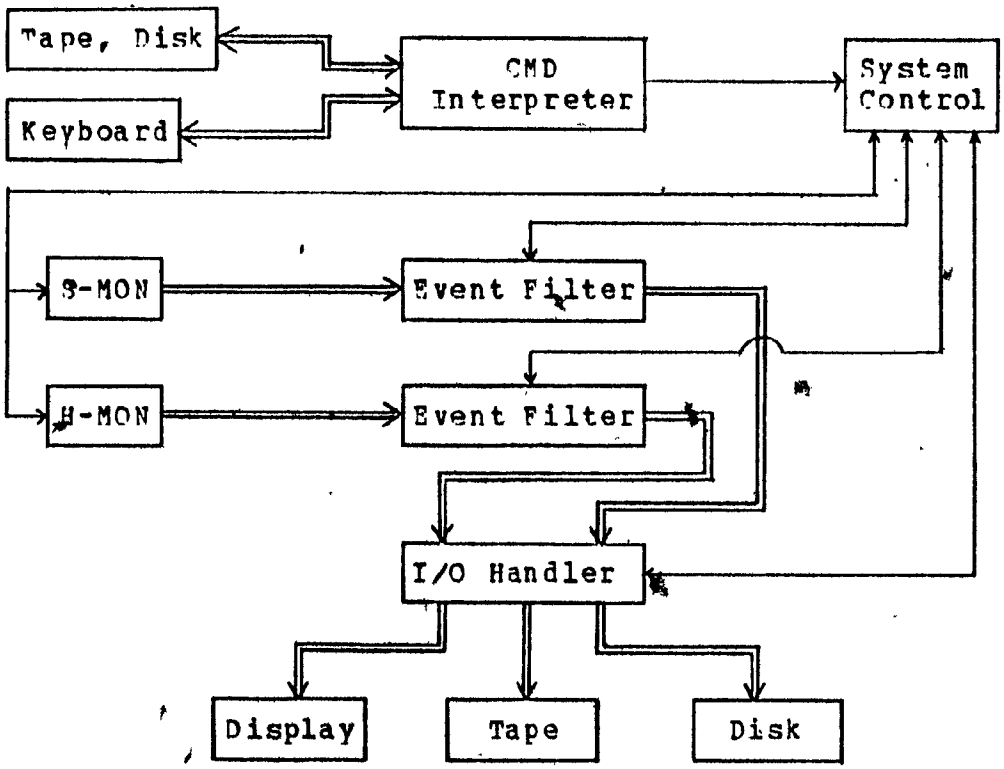

Figure 4.4: Monitor Control System

Other possibilities are left open for future extensions. As the receiving program in S-MON is a flexible, command code driven dispatching routine, other commands to S-MON like "examine location X" or "change value of X to Y" may be added.

4.4 Monitor Implementation

The two chapters on the soft and hardware implementation of HIMOS will give a comprehensive description. Further details on the software can be found in appendix B which contains the complete source listings. Appendix C gives a documentation of the hardware monitor and the communication link.

4.4.1 Software Monitor (S-MON)

S-MON is coded in PDP 11 MACRO assembler and consists of one load module of approximately 2 K words. In order to make S-MON core resident and protect it against user program as well as RT-11 access, the operating system bootstrap had to be modified. The idea is to load the executive not completely on top of the existing memory space but to leave some room for S-MON right above RT-11. In the current configuration RT-11 is located below 24 K. As a result RT-11 assumes that 24 K is the physical limit of available core and will not access higher locations. Furthermore it will abort attempts to modify the top area from the system console. A description of how to change the bootstrap can be found in the software support manual [42].

Another system modification was necessary to preserve the communication channel for the control system. As mentioned earlier, RT-11 loads every executable program starting from physical location 0 and resets afterwards important vector locations between 0 and 1,000. The restoring is performed in accordance with the system protection map. This map is

dynamically changed by S-MON as soon as the monitor is started. The following vectors are protected:

trap vector, loc. 34

clock vector, loc. 104

communication link vector, loc. 450.

User programs are usually terminated by transferring control back to the operating system with the EXIT macro call (EMT 350). If register 0 is cleared at this moment, RT-11 will execute a RESET instruction which disables all interrupts before reenabling the vital interrupts for the system. This however would harm the monitor operations because first, S-MON cannot receive supervisor commands any more and second, if it is in sampling mode, the clock interrupts would be turned off and the timer registers would be reset to 0. For these reasons, S-MON changes the RESET instruction in RT-11 to a NOP. This modification does not distort the performance of RT-11 but it insures full protection for the monitor activities.

4.4.1.1. Event and Data Selection

Software events are defined as interrupts which are intercepted by S-MON. A total of 32 events can be specified which is enough to cover all possible interrupts in the PDP 11/45. At initialization time, S-MON exchanges the content of an interrupt vector with a new PC and PS, and saves the original vector in a monitor table (VECSAV). To identify an interrupt, an event code (EC) is assigned to it. A convenient place for the identification was found in the condition code which resides in the 4 least significant bits

of the PS. As we have 32 possible EC's and only 16 can be specified using the condition code, two different interrupt entries were created in S-MON. This means that the new PC for events in group 0 points to ENTRY0 in S-MON and the PC for group 1 points to ENTRY1 (fig. 4.5). Each group can accept a maximum of 16 different interrupts. The EC is then defined as group index times 16 plus the interrupt index given in PS. This gives a total range of 0 - 37 in octal. Table A.4 in appendix A lists the event codes with the corresponding interrupt and vector locations.

Two mask words (MASK0, MASK1) are used to specify which of the 32 interrupts are to be captured. For example, if bit 3 in MASK1 is set, event 23 will be activated. To define a monitoring session, the mask words can be specified either at the host console when S-MON is started or interactively from the control system. If the command sequence at the beginning of S-MON is skipped, a default configuration will be assumed in which traps to location 4 and 10 (EC 1 and 2) are intercepted.

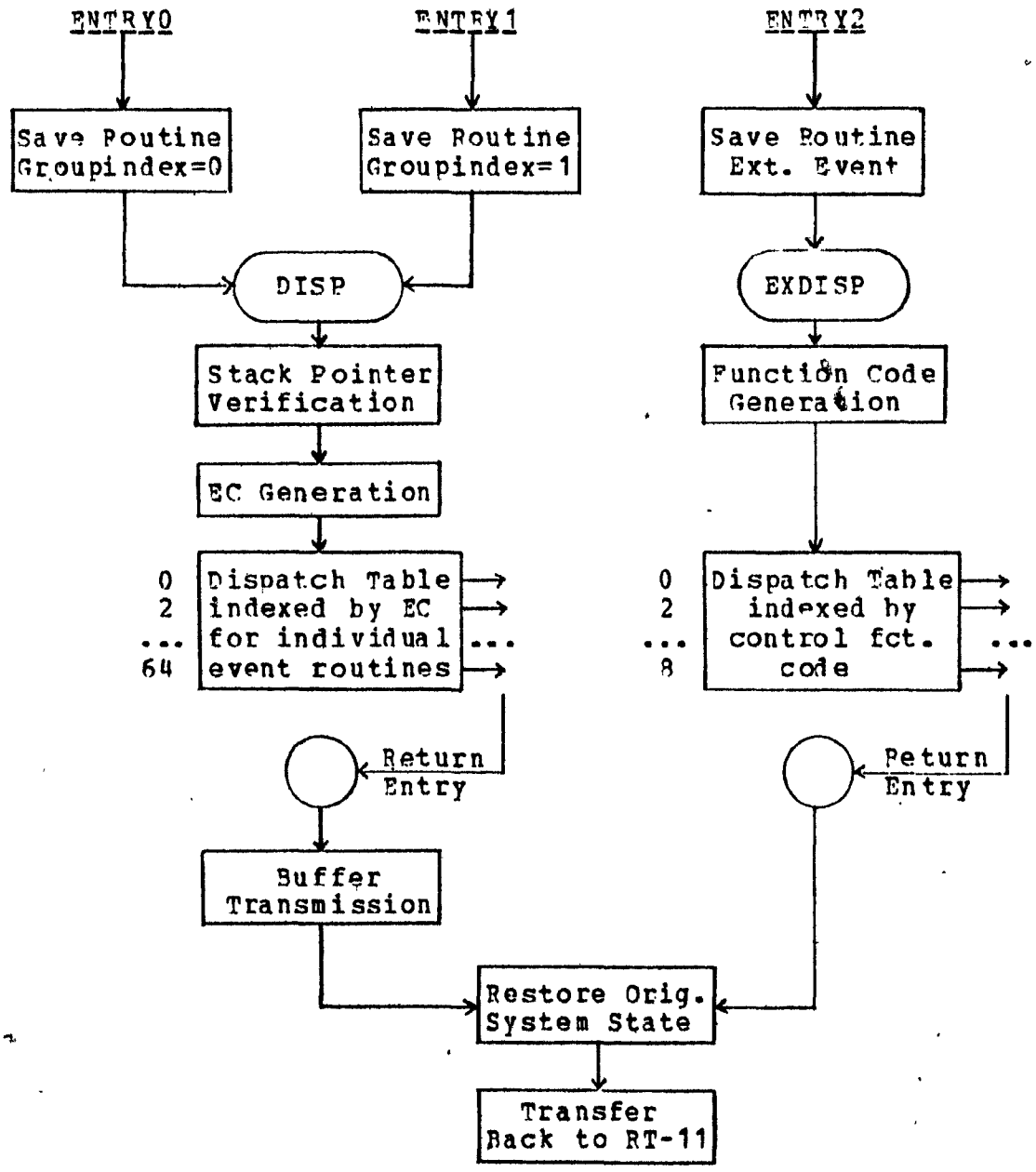

Figure 4.5: Core Resident Part of S-MON.

When an event arrives, the monitor creates a so called event record which contains information about the system state at the time of the interrupt. For every event, different system variables may be of interest and the monitor overhead would be increased considerably if all possible variables were recorded each time. Therefore S-MON

offers the possibility to define individual event records to meet the information requirements for every event. The desired record for each event is defined in the submask (SM) words. The minimum record for all events contains the three most important system parameters, i.e. the PC, PS, and SP. Most events can report additional values such as register content, buffer contents and RT-11 parameters. The exact assignment for the submasks is given in table A.5. For example, SM0 is responsible for the records of event 1 to 5. The low byte covers T4, T10, and the first two values of BPT, the high byte gives the last BPT bit and specifies the records for IOT and power fail. A value is included in the record when the corresponding bit is set.

More information can be reported for the EMT's. As this is a one word instruction [PC-2], the contents of the program counter before it was incremented will be the EMT code along with the function code which serves to distinguish 256 different EMT calls. For example, "104013" shows that an EMT 13 was executed. Because the EMT is used as a supervisor call in RT-11, the error code (ERC) can give valuable information.

The submasks SM2 to SM6 correspond to events in group 1 which are device interrupts. Depending on the device, register and buffer content can be included in the event report. The matching register addresses are contained in the device table of S-MON which is indexed by the EC.

The most important device is the RK-11 disk controller. SM6 is fully dedicated to define the event record for the RK-11 interrupt. It contains the basic processor

information plus the content of 5 disk registers (the register assignment is given in [41]). R3 which originally contained the word count was redefined. This can be justified because normally the word count is 0 when an interrupt occurs. It now contains the block number of the accessed file on the disk which can be obtained by transforming the physical disk address in R5. In the new representation, the disk address can immediately be used to refer to the directory dump where the relevant file name can be found.

Some events like EMT's and disk interrupts occur very frequently and it is desirable to perform preliminary data reduction on the host system before reporting the events to the supervisor. This eliminates unnecessary transmission time and decreases the input load at the control system. The submasks SM1 through SM6 contain one additional bit (bit 0 in each byte) for every event to indicate a preselection. If the bit is set, S-MON will examine one of the values in the event record and compare it to the content of a list of values, the so called select words. If the EC is found the event will be reported, otherwise it is ignored. As shown in table A.5, select words 0 to 3 can hold up to 7 EMT function codes (a function code can be specified in one byte). The end of the table has to be marked by a 0 entry. Select words 4 - 7 are used for TRAP codes. If no trap codes are specified the list can be used for another 7 EMT codes if necessary. The selection for disk events is kept as flexible as possible. Each item in the list specifies first the register number and second the value to be

compared. Thus, the criterion to report a disk event can, for example, be the disk error word or the physical address. TRAP or EMT codes can be extended over select words 11 - 20 if no disk selection is specified. This gives a maximum of 31 EMT codes or 23 TRAP codes which can be listed.

Every event record is headed by the corresponding event code. If specified, the content of the high speed clock buffer can be appended as the last transmitted value. This is important when program segments are to be timed.

All tables in the S-MON data structure are indexed by an even multiple of the EC. In conjunction with the masking technique, this has considerably helped to keep the execution time within reasonable limits.

4.4.1.2 S-MON ROUTINES

When the software monitor is started, it will print an identification on the system console and enter the command interpreter. This routine is responsible for the event and record definition. The execution is controlled by 4 dispatch tables which makes the program fast and easy to understand. The command interpreter (CMDI) is used to define the two mask and the seven submask words and to enter values for the select words. When the index for a mask word is entered, the mask word as it was defined previously is printed and modifications can be specified in the following line. For select words, CMDI distinguishes between SM0-10 and SM11-20. The first group is updated in byte form, i.e. for every word index two new byte values are requested. The second group is word oriented. For every index the two

error checks (e.g. CRC or LRC) could be used.

The speed of the parallel link is only limited by the UNIBUS speed, interrupt latency, and time requirements for the interrupt handler. In HIMOS the transmission rate is approximately 40,000 words/sec. This allows to transmit an average event record of 20 words in 0.5 ms.

HIMOS uses a simple protocol to transmit event records and specification tables over the communication link. The host system terminates an event record by the transmission of EOT and then waits to receive an acknowledgement (ACK) from the control station. When the host tables are updated interactively from the supervisor, the host sends the old table content, terminated by ETX. After the update, the control system issues an enquiry (ENQ) signal before transmitting the new table which is again terminated by ETX. All protocol and function codes are verified in both stations. A wrong function code will dispatch an error routine to generate a corresponding message. An erroneous protocol will cause a retransmission of the last record.

H-MON uses a parallel interface, similar to the ones described above, to transmit data to the control system. For this application, a DMA-channel would provide much better results but the corresponding interfaces were not available.

4.5 Control System Implementation

4.5.1 Program Control Structure

The control system (CS) program consists of 3 object modules: LOG, LOGIO, and LOGGRF (for listings, see appendix B.2). LOG is the main program and the other two modules contain basic I/O routines and subroutines for graphic display on the Tektronix terminal, respectively. To execute the CS program, the three object decks are linked together and loaded at starting address 1000.

After initialization of system parameters and interrupt vectors (S-MON, H-MON, clock), LOG enters the main dispatcher at WAITLP. This is the busy wait state in which the system waits for interrupts and schedules various tasks. During execution of this loop, the program examines the content of a dispatch word (DISPWD) and if one or more bits are set, the corresponding tasks are called (see fig. 4.6). Every time an event record is received, up to 3 bits in the dispatch word can be set in the event record processor (FILTER):

bit 1: record buffer

bit 2: print event record on terminal

bit 3: update graphic output.

The corresponding programs are discussed in chapter 4.5.4. Each of these routines will clear its own dispatch bit before returning control to the wait loop. The S-MON interrupt service routine (RCVREC) sets bit 0 in DISPWD to

schedule FILTER.

The second possibility to leave the loop is given by the reception of a command from the keyboard. This event transfers control to the system command interpreter (SYSCMD) which will analyze the command and schedule the according subroutine. Timer interrupts are directed to the line clock handler (LCLK) where the system time is updated.

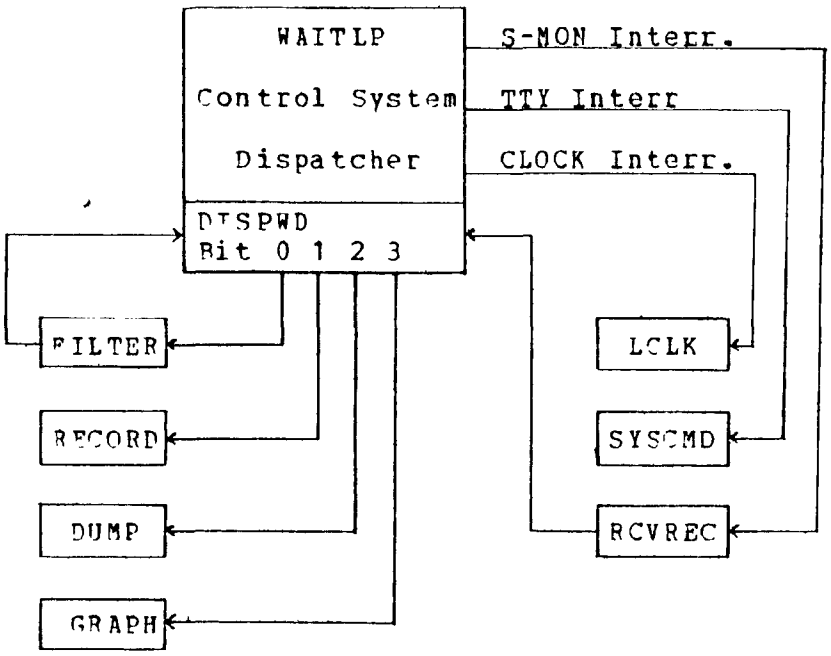

Figure 4.6: Control System Dispatcher

The centralized dispatching technique has helped to keep the program structure modular and easy to follow up. This is especially important in a system where different interrupts can occur at unpredictable times. Further program extensions can be introduced via unused bits in DISPWD without affecting the control structure.

LOG handles three input/output files:

Command File (SPC)

Table File (TAB)

Logging File (LOG).

Default extensions for file names are given in brackets. The command file contains all necessary specifications to define the supervisor system mode. It is created by the system command interpreter which accepts specifications either from the keyboard or from a previously generated command file. The table file is used in a similar way and keeps information regarding the host system mode, i.e. which interrupts are to be intercepted and what information is to be reported for every event. Both files help to simplify the use of HIMOS. Different monitoring sessions can be predefined to standardize frequently used operating modes. To invoke one of those sessions, the operator has only to know a small part of the full instruction set.

The logging file contains the event records as they were received from S-MON. A program (LOGPRT) was written to print this file in a readable form on line printer or data terminals. For performance evaluation which implies a statistical analysis of this file, other programs can be written according to the individual requirements. All HIMOS files can be stored on tape or disk units of the control system.

4.5.2 Command Interpreter and Operating Modes

The control system has three operating modes which specify the form of the event record output. For a detailed online analysis, the records can be displayed in numerical

form at the terminal. The record is headed by the EC and all following values are preceded by a two letter identification, explained in table A.6. Alternatively, the events can be presented in graphical form. A histogram is drawn dynamically in which a maximum of 21 different EC's can be entered along the X-axis. The Y-axis shows the number of occurrences for each event. The initial maximum on the Y-axis is set to 56. Every time one of the columns becomes too big, the picture is rescaled by factor of two. Concurrently with one of the display modes, the system can record incoming events in a logging file on tape or disk. Other display and storing features are explained below along with the corresponding control commands.

The operating mode of the control system can be defined by commands from the keyboard. The command interpreter (SYSCMD) accepts strings of virtually unlimited length but only the first character is significant to specify a command. New input is accepted in the system command mode which is indicated by an asterisk as the response character on the terminal. Commands that are given during other system activity, e.g. event analysis or output on the display, are stored for later processing. The corresponding delay is normally not significant because of the centralized dispatching technique which ensures that pending terminal input is served immediately after the last scheduled task. The first character of an input string is compared to a list of valid commands and the list index is used to dispatch the according task. If the character is not found, an error message is printed, followed by another asterisk on the next

line. In the following, the meaning of all valid system commands as they are listed in appendix A, table A.7 will be discussed. If necessary, further explanations for the corresponding operation mode are given.

W: This command suppresses graphical or numerical terminal output. Recording and counting of event information will be continued according to previous mode specifications. This feature is useful to give the operator some time to observe results on the screen before they are overwritten.

P: Graphic and numeric terminal output are resumed if they were previously suspended by the "Wait" (W) command. Output will continue with the next record after P is given.

E: Once this command is given, event reception is temporarily stopped and the user can enter a list of event codes, called special events. After termination of this input mode, CS will continue to receive events and compare every new EC to the list of special events. If the EC matches, a message announcing a special event along with the complete event record is printed. The "Wait" command is ignored in this case. Simultaneously, the host system is stopped until a "Continue" (C) is entered.

C: If the host was suspended due to a special event, this command will release execution. If the host was not disabled, an error message will be printed.

B: If graphic output was specified, the command will reset the counters of the graphic routine and a coordinate system with the initial scale for histograms will be drawn

on the Tektronix screen. It is recommended to issue this command before the graphic output mode is entered.

A: In graphic mode the screen will be erased and the histogram is drawn again with all columns as they were given before. This feature is necessary because the display cannot be erased partially. Although three quarters of the screen are reserved for command and output text, it can happen that the picture is overwritten.

D: After erasing the screen, a summary count of all events received so far in this session is displayed on the terminal.

S: The command interpreter for the host table update routine is invoked. The command structure is the same as it was described for S-MON. Any host system activity (except S-MON) is interrupted during this procedure until the new event configuration is established.

H: The hardware monitor control word can be defined and a new H-MON mode started. This command has not been implemented so far and does not cause any actions.

L: This command is used to define the logging mode for hard and software events by means of a subcommand interpreter (CSIL). A self explanatory message will ask for further specifications. The operator has the following three possibilities:

OLD: An existing specification file can be used to define the logging mode. The corresponding file name has to be entered and, if desired, the file can be altered.

NEW: A new mode is defined from the terminal. Consequently the operator can specify if the session is to be

recorded and/or displayed in numerical or graphical form. For the recording mode an output file name has to be entered.

UPDATE: The current mode is displayed and for each item as discussed in the last case, a new specification can be entered. The possibility to open a new output file is given.

After the command sequence for one of the three possibilities, the mode specification can be saved on a new command file or in the case of "OLD" it can be rewritten in the unchanged or altered form on the original file. Upon termination of CSIL the new logging mode is active.

While processing the commands, SYSCMD suspends any logging action and in some cases it blocks host activities by disabling its S-MON interface. Although this solution interferes with the operation of the host, it was chosen to insure a well defined state of HIMOS during and after modifications of its operation mode. For commercial applications another method could have been used (e.g. resetting the host to the unmonitored state before processing any commands), but for research and development environment, the technique implemented has proven to be sufficient and necessary when the system state has to be preserved until a new mode is defined.

4.5.3 Event Record Processing

Upon reception of the first interrupt from S-MON which announces the transmission of a new event record, control is passed to the receiving routine (RCVREC). Two alternating buffers of 256 words each are available to store incoming data. A flag indicates the current input buffer and a pair of pointers delimit the last received record within one buffer. As soon as one buffer is filled, RCVREC sets bit 1 in DISPWD to schedule the buffer recording routine (RECORD) if recording mode was specified. Record and buffer output can be processed concurrently with the reception of new records.

After receiving one record RCVREC sets bit 0 in DISPWD to schedule the FILTER and returns control to the main dispatcher. The wait loop will then continue to examine the dispatch word bit by bit in a round robin fashion and invoke the corresponding tasks. If, for example, it was interrupted before processing bit 2, it will first invoke the terminal output routines to process the previous event record before calling FILTER to analyse the just received record. This technique insures that no information is lost. If more than two records arrive at the input before the previous one was processed, reception is disabled until the CS has caught up. This only happens for high event density or when one of the slow terminal output modes is specified. More than two records cannot be buffered prior to the analysis because the system would lose its capability to react in real time upon the occurrence of special events.

If the "Wait" (W) command was given, FILTER will skip the terminal output dispatcher. Otherwise bit 2 or 3 are set according to the mode specification. In case special events were defined, the event table is searched and if the EC was found, a message is printed and the numeric output routine (DUMP) is scheduled via bit 2. The acknowledgement signal to release the host is only transmitted if the event was not found in the table. Otherwise the host keeps waiting for this signal until the "Continue" command (C) is entered from the keyboard.

Finally the current system time is appended to the event record and the count table for the summary report (command D) is updated before FILTER returns control to the wait loop. To reduce the monitor overhead, only one real data reduction is performed in FILTER, namely the special event feature which is especially useful when terminal output was suppressed via the "Wait" command. In this case only the special events are reported in real time while there is still the possibility to record a detailed report of the session on tape or disk for later analysis.

The two terminal output routines DUMP and GRAPH find the record in one of the two buffers according to the pointers which were set as record delimiters in RCVREC. DUMP transforms the record into a readable form and prefixes every value by a two letter identification. Each record is printed on a new line.

Subroutines for the graphic output are found in the object module LOGGRF. The program maintains its own table (ECTAB) to count received events. Every EC which is not yet entered

in the table will display the new identification along the X-axis, print the initial column increment, and open a new table entry. Upon overflow of one of the event columns, all counts in ECTAB are divided by two and the Y-axis of the display is rescaled. The column increment per event occurrence depends on the current scaling factor.

If no detailed information about host activities is required, the graph mode provides a convenient summary report. The histogram which is updated in real time shows not only what the host is doing but also in what sequence the events happen. It is mainly used to get a first impression of the monitored process and it helps to define the monitoring session for a more detailed analysis.

CHAPTER 5

Application of HIMOS

5.1 Execution Analysis under RT-11

One of the design goals of HIMOS was to develop a technique for real time and off line analysis of program execution. There are chiefly two approaches to this problem labelled the active and the passive methods. The latter one consists in the application of standard features provided by the monitor. System events such as I/O activity, executive calls, error traps, etc. can be intercepted. To obtain statistical information, the PC and other important system variables can be sampled. In both cases the target program does not need to be changed.

The active method requires certain changes in the observed program. Check points or hooks are inserted to the source code in order to emit a hook identification to the monitor when they are executed. (This method is similar to the break point technique used in debugging routines). For PDP 11 assembler programs, the TRAP instruction is used to identify a softhook. TRAP works in the same way as the EMT instruction; only it uses a different vector location. The high byte contains the operation code while the low byte holds the user defined hook identification. When, for example, TRAP 123 is executed, an interrupt to location 34 is generated and control is transferred to S-MON which will record event 7 with identification 123.

The TRAP instruction was chosen for several reasons to implement software hooks. It offers the programmer a quick and easy way to identify strategical parts of his program with a one word instruction. Once the hooks are inserted in a given routine, they can be disabled just by replacing the S-MON address in location 34 by an address containing a RTI instruction. Substituting the TRAP code by a NOP instruction eliminates the hook completely without affecting the program performance. Another low level hook technique should be mentioned in case the TRAP instruction is used for other purposes. Interrupts to S-MON can be created by an additional interface. The hook identification can then be transmitted by moving the desired code into the output buffer which is fed back in a closed loop to the interface input buffer from where S-MON can take the information.

To monitor execution of FORTRAN programs, two subroutines (INEMIT, EMIT) were written and can be linked to the main program. Both subroutines are listed in appendix B.3. EMIT finds the hook identification as the first argument of the call in the FORTRAN linkage table and uses it to form the corresponding TRAP instruction. INEMIT has to be called for initialization purposes before EMIT is invoked the first time. This is necessary because FORTRAN alters the TRAP vector for its runtime error message handler. EMIT calls should therefore only be used in fully debugged programs. Where this is not possible, the software hooks have to be implemented by means of the above mentioned alternative.

Software hooks are a valuable tool for many applications. They can be used to time execution over critical parts of a

program as will be shown in chapter 5.2.1. When they are placed on top of the executable code, they can emit a task identification for accounting purposes. If, additionally, event 6 (EMT) with subselection for EMT 350 (EXIT) is monitored, the time between the EMT call and EC 6 gives the total execution time of a job.

Under the RT-11 operating system, S-MON offers different possibilities to measure "productive" CPU time. The busy wait loop of RT-11 can be recognized by alternating calls of EMT 340 and EMT 360. Another approach is given by sampling the PC activity. With regard to CPU activity the memory can be grouped into several regions:

```
1000 - X       : User program execution
   X - Y       : I/O handlers
122222 - 124154 : RT-11 EMT handler
                  (if booted for 24K of core)
```

The values of X and Y depend on the individual program and can be specified via system macro calls. During execution the value of Y can be found in location 50 where it is stored by RT-11 at load time.

In any case, the definition of "productive" CPU time is relative. A true wait state as it is known in other systems does not exist under RT-11. Similar problems arise when I/O time is to be determined. Sampling the I/O handler space in core is one possibility. In chapter 5.2.2 another approach is taken. Disk access time is obtained as the difference between EMT 375 and a disk interrupt. EMT 375 invokes the I/O dispatcher in RT-11 which initiates a disk action. When the disk controller has found the addressed track and

transferred the complete record as it was specified in the word count, it gives an interrupt (EC 30). It was found that the average delay between those two events is in the range of 35 - 45 ms depending on the occupied disk space. This value is in accordance with RK-11 specifications. Undoubtedly the hardware monitor will be very useful to complement these results for more detailed investigations.

Generally, several steps are necessary to monitor a given process. An initial approach is to display global execution characteristics in graphical form on the terminal. Based on this first impression, more detailed measurements can be specified and event records can be directed in numerical form to the terminal or to mass storage for off line evaluation. Figure 5.5 shows a histogram obtained during the execution of a test batch stream. Event 5 (power fail) was specified as a special event. It occurred at the end of the batch stream when the 11/45 power was turned off.

5.2 Results of two Monitoring Sessions

The following two paragraphs will give exemples of monitoring sessions using the active and the passive methods of execution analysis. In both cases, the machine level representation of the programs is unknown and any observed PC values cannot be matched readily to a functional entity of the code.

5.2.1 FORTRAN Program Execution

A program (SAMGEN) was written to produce J random numbers in the range from 0.00 to 99.99. When the routine is entered, the value of J is requested from the terminal. In the following, the program executes J times a loop in which a random number is generated and immediately written onto disk. The random numbers are obtained from the FORTRAN function "RAN". It should be noted that the program was written to give an example for the possibilities of HIMOS and not to optimize a given problem. A listing of SAMGEN can be found in appendix B.4.

The program was executed twice, each time producing 20 samples. For the first run, SAMGEN was linked with a FORTRAN library which replaced all floating point instructions by software routines. The second time the floating point processor was used which improved the execution time but did not change other characteristics of the execution profile.

In figure A.7 the monitor output for a SAMGEN execution using the floating point processor is listed. The HIMOS command sequence to define this session on the control system is shown in figure A.8. When the command "RUN SAMGEN" is given, RT-11 executes several EMT calls to load the program into core (EMT 374, 375). The first TRAP (EC 7) identifies the EMIT call with argument 0 which can be found in the low byte of CP (content of PC). A delay of approximately 2 seconds follows, during which a message is printed and J read from the terminal. EMIT (1) and EMIT (2)

are produced before and after the first random number generation. During the following 6 EMT's the output file is opened and at EMIT (3) the first random number is written on disk. These three emitter calls appear another 10 times during the loop execution before EMIT (4) indicates the end of the program. Six EMT calls are issued to close the output file and enter it into the disk directory before EMT 350 indicates the RT-11 EXIT call.

For the first run (run A) an execution time of 234 ms (excluding terminal I/O and file handling) was calculated. The second run using FPP (run B) took 193 ms. This difference is due to the fast performance of the floating point processor but it also reflects a certain delay for the disk output. The latter factor was excluded by averaging the time intervals between EMIT (1) and EMIT (2). After subtraction of 2.36 ms per event for HIMOS overhead, the final result shows a random number generation time of 0.270 ms for run A and 0.100 ms for run B. The difference of 170 micro seconds per random number generation can be attributed to the processing time improvement for floating point operations.

5.2.2 FORTRAN Compiler Analysis

In the following example, a short FORTRAN program was compiled under several HIMOS modes to obtain insight into the RT-11 FORTRAN compiler characteristics. The test program HISTO is listed in appendix B.4. It can be used to read a file of sample values from disk and to print a distribution of these values in the form of a horizontal

histogram on the terminal. At the same time, a second file containing the values in ascending order is created.

RT-11 supports a multipass FORTRAN IV compiler. It consists of a root segment and several overlays which are dynamically loaded in core. The core resident part comprises of the root and one overlay, the system stack, device handlers, symbol table, and an internal representation of the source program. During the monitoring sessions it was discovered that the compiler uses no scratch files, only overlays are swapped between core and disk. A high level description of the compiler can be found in RT-11 manuals [52, 53].

Because no source listings of the compiler were available, the analysis was done in a top down fashion. In the first step a graphic performance profile was produced on the Tektronix screen (fig. A.6). The histogram shows five different events:

EC 24 : Timer interrupts

EC 20 : Terminal input

EC 21 : Terminal output

EC 30 : Disk interrupts

EC 6 : RT-11 calls for file handling.

EC 21 is more than twice as high as EC 20, because every typed character is echoed together with one additional filler character. The graph gave an impression of the amount of events and their relation in time.

For the second monitor session it was decided to select only disk interrupts and to record the protocol on disk. Figure A.9 shows the corresponding HIMOS output. The first

11 events report directory access for the source file HISTO.FOR and the output file HISTO.OBJ, both located in the 4th directory segment (block 16). At system time (ST) 103.18, five blocks of compiler overlay are read in before at ST 103.24 the source program is loaded from disk address (R3) 6531. (All file references were confirmed by comparison with the directory listing). After having read two more overlays, the compiler writes the first block of object code at ST 103.50 on disk. The content of R4 (bus address register) shows that the compiler uses always the same buffer at location 10520 to store one block of produced object code. Compilation is continued by alternating overlay input and object output until ST 106.06 when the last FORTRAN overlay is loaded into core. It is probably used to generate completion messages. Accesses to disk address 116 and 143 show that RT-11 utility routines for terminal I/O are brought into core. Eight directory accesses starting at ST 106.21 close the source and object files.

Already this simple monitoring session shows several performance bugs in RT-11 and the compiler. Directory access time can be reduced when the search is started within the last directory segment which is still in core. The probability of finding a file entry close to the last accessed one is higher than to find it in the beginning of the directory. A similar technique is found in memory paging algorithms where it is known as the "least recently used" method (see for example [37]). Further I/O time could be saved if the object code buffer size in the FORTRAN

compiler is increased. Corresponding changes in RT-11 and the compiler would reduce the compilation time at least by 12% in this example. The fact that all disk interrupts occured while the PC was pointing to RT-11 indicates that the compilation process is I/O bound. The program transfers control back to RT-11 for the next I/O request before the previous one is terminated. RT-11 executes a wait loop between locations 127130 and 127124 until the next interrupt arrives.

To obtain more details on the relationship between I/O and computing activity, the compilation was sampled at different frequencies. Figure A.10 shows the HIMOS output for 40 ms samples and fig. A.11 lists the corresponding commands for this session. The PC values show that less than one fifth of execution time is spent in the FORTRAN compiler. Another sampling session with 10 ms intervals confirmed this result. As expected, the compiler is only active immediately after disk interrupts.

The overhead imposed by HIMOS during the sampling sessions amounted to 5% of the total compilation time. It is interesting to see that the monitor overhead did not increase significantly with a higher sampling rate. This can be attributed to the fact that most samples were taken during I/O wait periods.

A last experiment was done to check if compilation time could be reduced by locating the FORTRAN compiler and the object file on the disk closer together. The disk space between the two files was reduced by 1172 blocks which corresponds to 45 tracks. As a result, the compilation was

400 ms quicker. Because the disk arm moved 12 times from one file to the other, the average saving per arm movement is approximately 33 ms.

5.3 Further Possibilities for HIMOS Assisted Research

Apart from the possibilities demonstrated in the last chapter, there are mainly three research topics which have influenced the implementation of HIMOS. The following paragraphs will point out how the monitoring system can be used to assist this research.

During the past few years, several techniques have been developed to prove the correctness of a given high level language program. One method, called the "deductive theory" was presented by Hoare and Lauer [35]. The meaning of a language is defined by axioms together with rules of interference. Theorems are derived from the axioms by means of these rules. The basic axiom of the theory is written as P{Q}R where P and R are assertions about the memory state before and after execution of the statement (or program) Q. P states the necessary entry conditions for Q, and R is calculated from P and Q using the theorems of the deductive theory. If Q terminates and if P was true before execution, then R will be true when execution is complete. Given a linear sequence Q of statements without branches or entry points, logical expressions for P and R can be established. When the sequence is executed on the host machine, memory values can be monitored to compute the expression for P. When R is true upon exit from Q, the sequence has been executed correctly according to the programmer's intentions.

If, for example, a programmed loop (L) is observed, two software hooks are sufficient to verify the correctness of execution no matter how many statements are involved. The first hook is located in front of L and causes the verification of P in the control system. If P is not true, the process should be aborted because the termination of the loop cannot be guaranteed. The second hook is inserted immediately after the exit of L. At this point, the evaluation of R will show if the loop was executed correctly. Memory values that are necessary to compute the assertions can be obtained at interrupt time either through the hardware monitor directly from memory or by means of the software monitor from general purpose registers.

This method has the advantage of verifying correctness of programs during execution without producing an excessive amount of data as the trace technique does. It also reduces the necessary preparation time for the calculation of assertions which is required for purely theoretical correctness proofs.

The prevention and detection of deadlocks constitutes another promising application field for monitors. Although HIMOS was implemented for RT-11 which is a single user system under which this problem cannot be encountered, it is felt that a short discussion of this topic is justified by the importance of the problem. A deadlock is a situation in which parts of, or even all system activities are blocked due to an erroneous scheduling decision. A fatal deadlock situation may, for example, arise when the spooling space becomes completely filled with input records for jobs

waiting to execute and with output records for jobs not finished executing. In many systems there is no way to recover the spooling space which is occupied by a partially executed job. Several ad hoc methods have been employed to reduce the probability for deadlocks without giving a guarantee for their complete elimination. Other approaches, mostly based on graph models (see for example [54]), attempt to formalize the problem and to construct detection and recovery algorithms. Hybrid monitors offer a compromise in the detection and prevention of deadlocks. The software part can observe the task arrangement in waiting queues of the system as well as possible relations between those tasks. This may lead to the detection of situations in which tasks are waiting for each other without being able to resolve their requirements. The workload of critical resources can be observed by the hardware monitor. If, for example, the utilization of spooling space becomes too high and the monitor control system obtains at the same time the information that the waiting queue for this resource is growing, an imminent deadlock is indicated. In such situations a warning can be displayed at the operator's console or the monitor can supply a corresponding message directly to the operating system.

Finally, HIMOS can give practical assistance for efficiency analysis of paging algorithms. The memory management unit in the PDP 11/45 divides the full address range of 128 K words into 48 pages of 32 to 4096 words each. When a location outside of the current page limits is accessed a trap to address 250 is generated. When this trap vector is

intercepted by S-MON, the occurrence of page faults can be monitored. In sampling mode, S-MON can examine the content of the page address registers to record the current memory partitions. This information can be complemented with a memory utilization report from H-MON when it is sampling the PC activity.

The examples outlined in this and the previous chapter are far from giving a complete survey of applications for HIMOS. Individual requirements and practical experience with the system should motivate exploration of further possibilities of the monitor.

CHAPTER 6

Evaluation and Conclusion

6.1 Evaluation of HIMOS

The monitor has proven to be a reliable tool for performance measurements. Once acquainted with the system, the user can dispense with a large set of control commands. Several command files provide the possibility to predefine various monitoring sessions which can be evoked without profound knowledge of system internal details. All monitor results reflect exactly the host computer activities and HIMOS reports are reproduceable at any time. The system's greatest successes have been the following:

a. Its centralized control allows the specification of all functions from the terminal at the supervisor system. Erroneous command input or ambiguous instructions (such as two different output modes on the same terminal) are detected and comprehensive error messages are generated. System internal plausibility checks recognize software and hardware errors.

b. Real time data filtering reduces the amount of recorded and displayed information. Only pertinent data is retained for further analysis.

c. Its interactivness enables the user to redefine a monitor session according to new requirements while host activities are suspended at a defined state. Host execution can be slowed down to follow up event occurrences in real

time on the display.

d. Three different output modes of which two can be handled concurrently, satisfy all needs for online and offline analysis. The graphic output produces a dynamic picture of system activities. A modular implementation facilitates future extensions for other integrated evaluation features.

e. Most failures of the host CPU and any peripherals attached to it can be reported. Analysis of such events is assisted by the accompanying status report.

f. All necessary information for various applications is supplied, namely:

- Performance improvement
- System profile, activity log
- Problem oriented sessions (debugging)
- Activity reports for simulation model input.

Yet, HIMOS also has some deficiencies, specifically:

a. The command handler for host system mode specifications accepts a very abbreviated form of input. Some familiarisation is required to define the host tables.

b. The monitor overhead is relatively high. Creation and transmission of one event record takes from 1.43 ms to 3.14 ms, depending on the length of the record. During this time the host computer is exclusively dedicated to monitor activities. This time requirement is explained by the following 3 reasons:

1. To maintain the capacity of interactive response to special events, monitor actions cannot overlap with

host operations to serve user or RT-11 tasks.

2. Event records are encoded into ASCII prior to their transmission.
3. Event records are analyzed in the control system to detect special events before the host system is released for normal execution.

c. Because S-MON was implemented under RT-11 its possibilities are limited to applications within a single user system.

d. The operating system has to be slightly modified to allow undisturbed operation of S-MON. With the exception of the FORTRAN run time error message handler, the changes do not affect any system properties.

The hardware monitor will most probably be implemented by the end of this year. Because of time and equipment limitations, many desirable features were excluded in the final design. Due to the interrupt latency of the PDP 11/20 CPU, the maximum sampling frequency had to be limited to 10 KHz. A DMA link between H-MON and the control system would offer a variety of further applications. As the control software includes already all necessary routine entries to handle the communication to and from H-MON, its adaptation to HIMOS will not create major problems.

6.2 Conclusion

Measuring computer performance during program execution can be compared with taking the blood pressure of a man under effort. Monitoring can actually be compared to

medicine. The only major difference lies in the complexity of the observed systems. In both cases the performance is evaluated and important points for improvement or reasons for failure can be found.

The necessity to monitor activities in large computer systems has been generally recognized. The objectives are not only to increase cost efficiency of an installation but also to ensure reliability and to obtain hints for further efforts in research and development.

Mini computers have conquered an important market place during the past decade. Among other applications they are especially well suited for real time applications in process control. In this environment high reliability is an essential factor and monitors are a valuable tool to insure this objective.

During the implementation of HIMOS, several deficiences in the design of computer hardware and software became evident. It is hoped that future hardware design will consider probe point requirements for hardware monitors. Especially with regard to the increasing trend of miniaturization and large scale integration there is a danger that many important signals become buried in a single chip. Monitor facilities should be a part of the standard equipment of CPU and peripheral hardware. One step in this direction has, for example, been taken by DEC. The KL10 processor produced by this company incorporates a PDP 11/40 CPU which is partly used to monitor hardware signals.

The same arguments can be cited for the development of operating system software. Information from monitoring

devices is useful to assist operating system decisions. Especially when software reaches the magnitude of OS/360, additional observation tools are indispensable.

Consequently two different approaches can be stated for system development with respect to monitors. The monitor can form an integral part of the system's hard and software. This approach was, for example, taken during the development of "Multics". However, this solution seems to have some dangers. It is quite possible that weak points in the system are only confirmed by corresponding deficiencies in the monitor, especially if the same team is responsible for implementing both parts.

A modular approach using distinct systems seems therefore more promising. It is quite possible that the standard configuration of future computer installations will dedicate one processor exclusively to control and measure the remaining system components. This method was adopted for the implementation of HIMOS. It is believed that HIMOS will constitute a valuable tool for future projects on the mini computer installation at McGill.

"The purpose of measurement is insight, not numbers."

(R.W. Hamming)

APPENDIX A

Figures and Tables

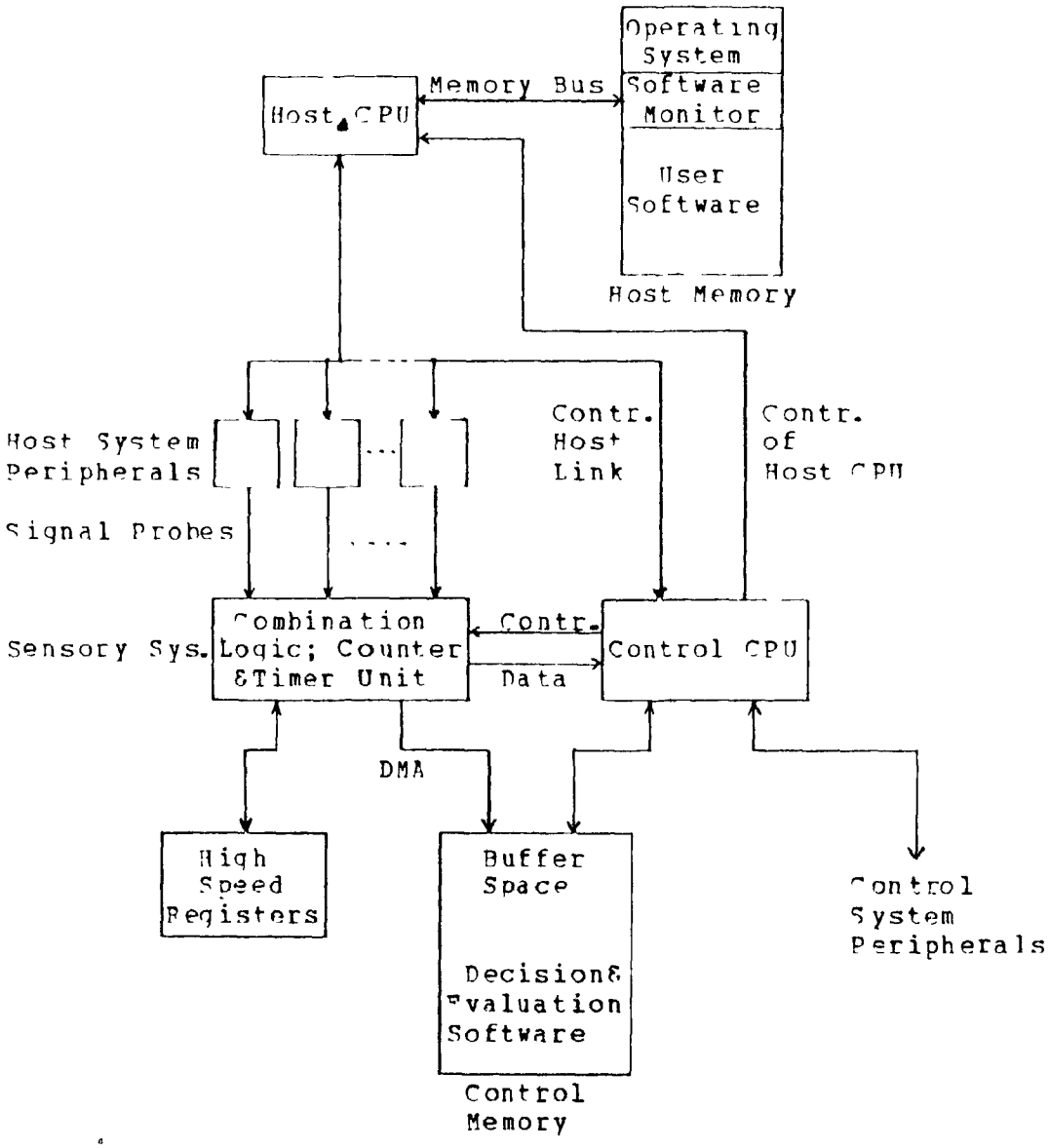

Figure A.1: A General Instrumentation System

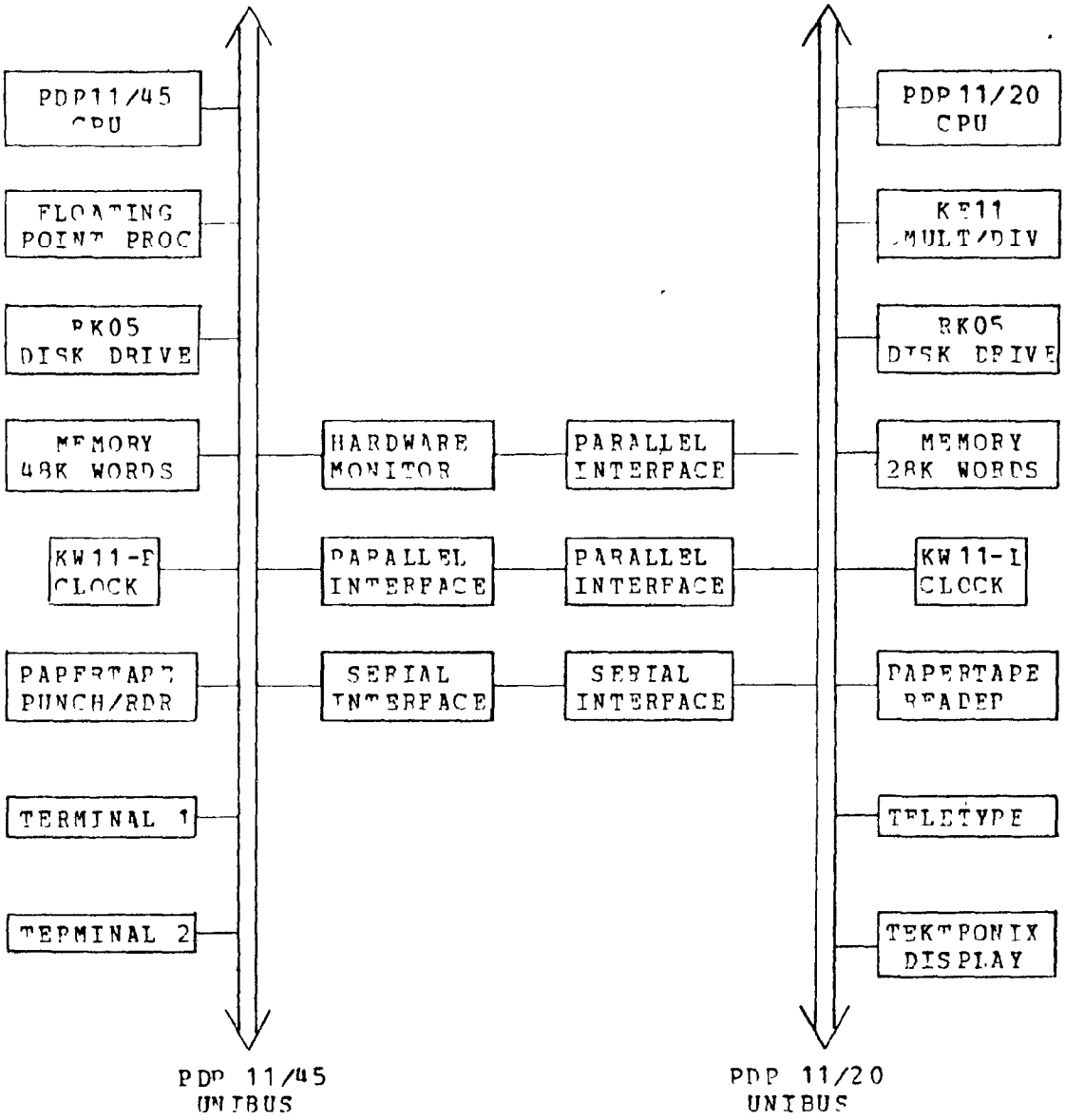

Figure A.2: System Configuration

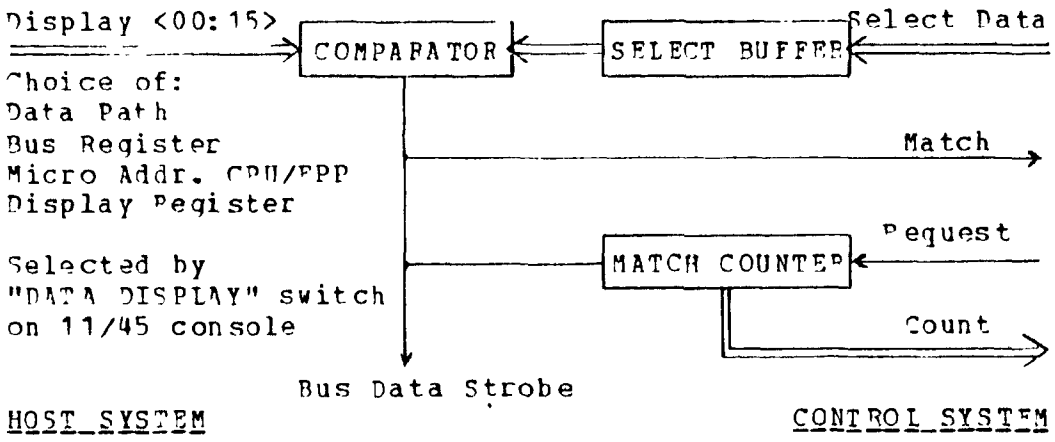

Figure A.3: Display Comparator

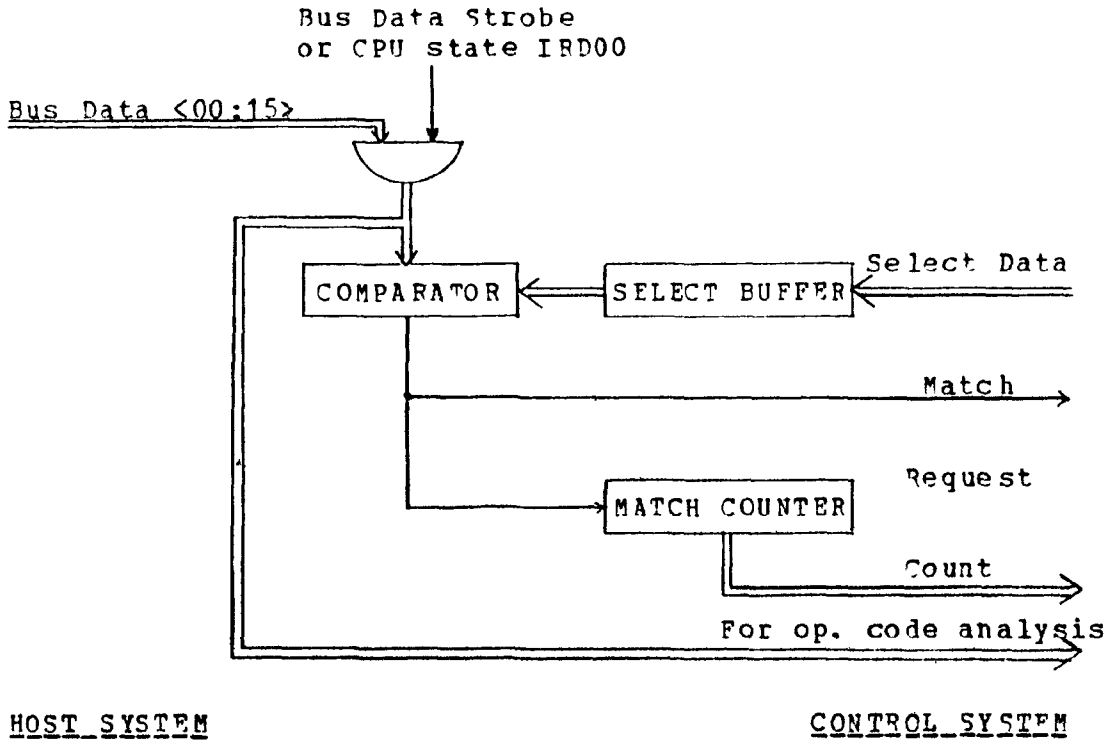

Figure A.4: Operation Code Analysis on UNIBUS Data Lines.

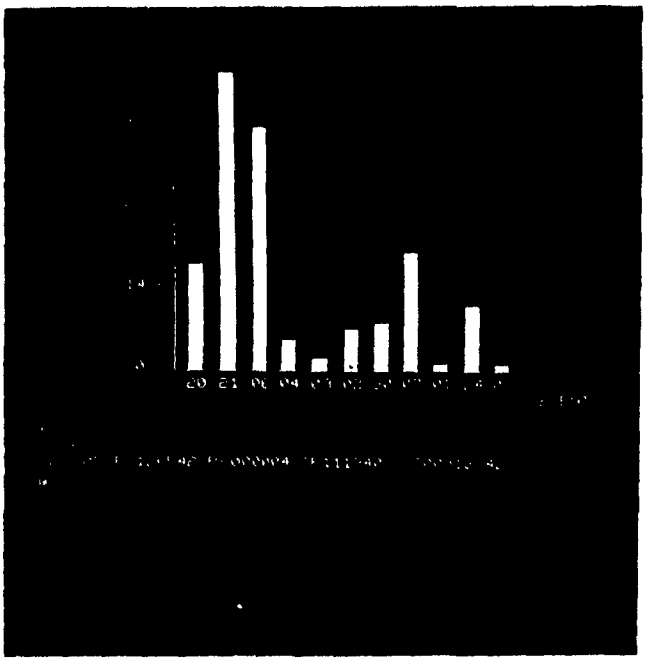

Figure A.5: Performance Profile of a Test Batch Run

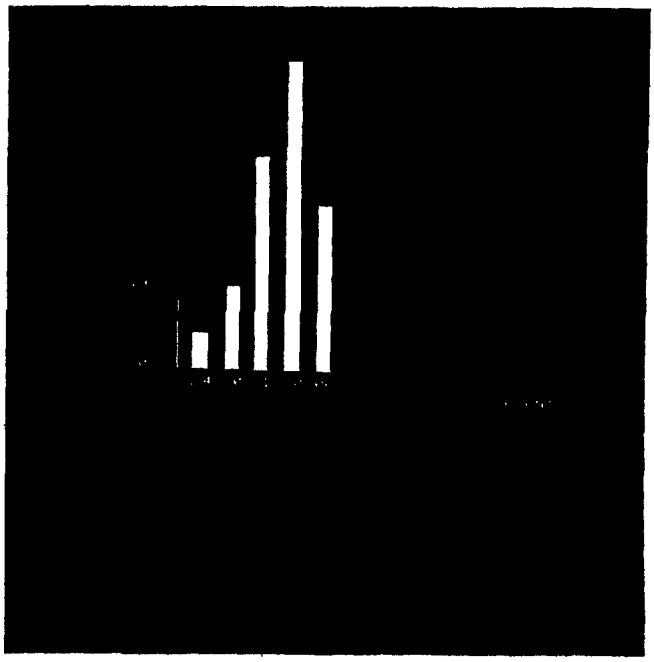

Figure A.6: Performance Profile of a FORTRAN Compilation

```
HIMOS LOG 14-7-76, 15:43,  SAMGEN RUN UNDER FPP . EC6,7

000006 PC106004 CP104374 EW000000 HT152345  ST00275.11
000006 PC115546 CP104375 EW000000 HT152407  ST00275.12
000006 PC115546 CP104375 EW000000 HT154216  ST00275.17
000006 PC115546 CP104375 EW000000 HT155135  ST00275.20
000006 PC115546 CP104375 EW000000 HT156042  ST00275.23
000006 PC115546 CP104375 EW000000 HT156250  ST00275 23
000006 PC013354 CP104375 EW000000 HT162405  ST00275 36
000006 PC013420 CP104375 EW000000 HT162437  ST00275 36
000007 PC001506 CP104400 HT162516  ST00275 37
000007 PC001506 CP104401 HT030077  ST00277.32
000007 PC001506 CP104402 HT030130  ST00277 32
000006 PC005160 CP104375 EW000000 HT034073  ST00277 44
000006 PC005160 CP104375 EW000000 HT035200  ST00277.47
000006 PC005160 CP104375 EW000000 HT036120  ST00277 50
000006 PC005160 CP104375 EW000000 HT037012  ST00277.53
000006 PC005160 CP104375 EW000000 HT037241  ST00277 54
000006 PC005160 CP104375 EW000000 HT037727  ST00277 56
000006 PC005160 CP104375 EW000000 HT040152  ST00277 56
000006 PC005160 CP104375 EW000000 HT040706  ST00277.59
000007 PC001506 CP104403 HT043213  ST00278 06
000007 PC001506 CP104401 HT043242  ST00278 06
000007 PC001506 CP104402 HT043275  ST00278 06
000007 PC001506 CP104403 HT043345  ST00278.06
000007 PC001506 CP104401 HT043374  ST00278 07
000007 PC001506 CP104402 HT043427  ST00278.07
000007 PC001506 CP104403 HT043477  ST00278 07
000007 PC001506 CP104401 HT043526  ST00278.07
000007 PC001506 CP104402 HT043561  ST00278.07
000007 PC001506 CP104403 HT043631  ST00278 07
000007 PC001506 CP104401 HT043660  ST00278.08
000007 PC001506 CP104402 HT043713  ST00278 08
000007 PC001506 CP104403 HT043771  ST00278 08
000007 PC001506 CP104401 HT044020  ST00278.08
000007 PC001506 CP104402 HT044052  ST00278.08
000007 PC001506 CP104403 HT044123  ST00278.09
000007 PC001506 CP104401 HT044152  ST00278.09
000007 PC001506 CP104402 HT044204  ST00278.09
000007 PC001506 CP104403 HT044261  ST00278.09
000007 PC001506 CP104401 HT044311  ST00278.09
000007 PC001506 CP104402 HT044342  ST00278.09
000007 PC001506 CP104403 HT044375  ST00278.10
000007 PC001506 CP104401 HT044447  ST00278.10
000007 PC001506 CP104402 HT044507  ST00278.10
000007 PC001506 CP104403 HT044563  ST00278.10
000007 PC001506 CP104401 HT044613  ST00278 10
000007 PC001506 CP104402 HT044645  ST00278.11
000007 PC001506 CP104403 HT044716  ST00278.11
000007 PC001506 CP104401 HT044745  ST00278.11
000007 PC001506 CP104402 HT044777  ST00278.11
000007 PC001506 CP104403 HT045053  ST00278.11
000007 PC001506 CP104401 HT045107  ST00278.12
000007 PC001506 CP104402 HT045171  ST00278.12
000007 PC001506 CP104403 HT045230  ST00278.12
```

Figure A.7: Event Report for "SAMGEN" Execution under FPP (1)

```
HIMOS LOG 14-7-76, 15:43,  SAMGEN RUN UNDER FPP . EC6,7

000007 PC001506 CP104401 HT045260  ST00278 12
000007 PC001506 CP104402 HT045312  ST00278.12
000007 PC001506 CP104403 HT045367  ST00278 13
000007 PC001506 CP104401 HT045417  ST00278 13
000007 PC001506 CP104402 HT045451  ST00278.13
000007 PC001506 CP104403 HT045525  ST00278.13
000007 PC001506 CP104401 HT045555  ST00278 13
000007 PC001506 CP104402 HT045607  ST00278.13
000007 PC001506 CP104403 HT045664  ST00278.14
000007 PC001506 CP104401 HT045714  ST00278.14
000007 PC001506 CP104402 HT046035  ST00278.14
000007 PC001506 CP104403 HT046112  ST00278.15
000007 PC001506 CP104401 HT046142  ST00278 15
000007 PC001506 CP104402 HT046174  ST00278.15
000007 PC001506 CP104403 HT046250  ST00278.15
000007 PC001506 CP104401 HT046300  ST00278.15
000007 PC001506 CP104402 HT046332  ST00278.16
000007 PC001506 CP104403 HT046407  ST00278 16
000007 PC001506 CP104401 HT046437  ST00278 16
000007 PC001506 CP104402 HT046471  ST00278.16
000007 PC001506 CP104403 HT046546  ST00278.17
000007 PC001506 CP104401 HT046713  ST00278.17
000007 PC001506 CP104402 HT046745  ST00278.17
000007 PC001506 CP104403 HT047021  ST00278.17
000007 PC001506 CP104404 HT047051  ST00278.18
000006 PC005160 CP104375 EW000000 HT055734  ST00278.39
000006 PC005160 CP104375 EW000000 HT056614  ST00278 41
000006 PC005160 CP104375 EW000000 HT057044  ST00278.42
000006 PC005160 CP104375 EW000000 HT057764  ST00278.45
000006 PC005160 CP104375 EW000000 HT060670  ST00278.48
000006 PC005160 CP104375 EW000000 HT061110  ST00278.48
000006 PC006430 CP104350 EW000000 HT063437  ST00278.56
```

Figure A.7: Event Report for "SAMGEN" Execution under FPP (2)

```
*L
LOG MODE DEFINITION
H-LOG OR S-LOG SPECS (H,S) : S
OLD, NEW OR UPDATE (O,N,U) : N
LOG RECORDING (Y,N)    : YES
NUMERIC DISPLAY (Y,N) : N
GRAPHIC DISPLAY (Y,N) : N
SAVE FILE (Y,N) : Y
 OUTPUT FILE FOR SPECS
*SES1=
 OUTPUT FILE FOR LOG RECORDING
*LOG5=

*S
HOST TABLE UPDATE
SPEC FILE-OLD OR UPDATE (O,U) : U

MASK UPDATE (Y,N)? : Y
GIVE INDEX OR T
#0
76543210765432l0
0000000000001110
0000000011001110
#T

SUBMASK UPDATE (Y,N)? :  Y
GIVE INDEX OR T
#1
76543210765432l0
0001001000110010
0001001000110011
#T

SELECT WORD UPDATE (Y,N)? :  Y
GIVE INDEX OR T
#0
350 :
351 : 343
#1
340 : 374
341 : 375
#T

TIMER FREQUENCY (>3: NO TIME LOG): 1
TIMER COUNT       : 0
SAVE FILE (Y,N) : NO
*D

EVENT COUNT:

 01:000000 02:000000 03:000000 04:000000 05:000000
 06:000027 07:000076 10:000000 11:000000 12:000000
 13:000000 14:000000 15:000000 16:000000 17:000000
 20:000000 21:000000 22:000000 23:000000 24:000000
 25:000000 26:000000 27:000000 30:000000 31:000000
 32:000000 33:000000 34:000000 35:000000 36:000000
```

Figure A.8: Command Sequence for HIMOS Session of Fig. A.7

```
HIMOS LOG 12-7-76,16 12, FORTRAN COMPIL  HISTO FOR, EC30

000030 PC127130 R3 000010 R4 113366 HT155506  ST00102 52
000030 PC127130 R3 000012 R4 113366 HT156426  ST00102.55
000030 PC127130 R3 000014 R4 113366 HT157345  ST00102.58
000030 PC127130 R3 000020 R4 113366 HT157552  ST00102 59
000030 PC127124 R3 000016 R4 113366 HT160265  ST00103 01
000030 PC127130 R3 000016 R4 113366 HT161103  ST00103 03
000030 PC127124 R3 000010 R4 113366 HT161411  ST00103.04
000030 PC127124 R3 000012 R4 113366 HT162331  ST00103 07
000030 PC127130 R3 000014 R4 113366 HT163251  ST00103 10
000030 PC127130 R3 000020 R4 113366 HT163455  ST00103 11
000030 PC127124 R3 000016 R4 113366 HT164170  ST00103 13
000030 PC127130 R3 001007 R4 006400 HT165664  ST00103 18
000030 PC127130 R3 006531 R4 012654 HT167626  ST00103 24
000030 PC127130 R3 001014 R4 006632 HT174304  ST00103 38
000030 PC127130 R3 001021 R4 007010 HT176205  ST00103 42
000030 PC127130 R3 007267 R4 010520 HT000354  ST00103 50
000030 PC127124 R3 001026 R4 007152 HT002356  ST00103 56
000030 PC127130 R3 007270 R4 010520 HT005136  ST00104 05
000030 PC127124 R3 001033 R4 006344 HT007345  ST00104 12
000030 PC127130 R3 007271 R4 010520 HT011717  ST00104 19
000030 PC127130 R3 001040 R4 006502 HT014332  ST00104 27
000030 PC127124 R3 001045 R4 006604 HT016233  ST00104 33
000030 PC127130 R3 001052 R4 007144 HT020133  ST00104 38
000030 PC127124 R3 001060 R4 007360 HT021257  ST00104 42
000030 PC127130 R3 001063 R4 004316 HT023055  ST00104 47
000030 PC127130 R3 001066 R4 004646 HT024036  ST00104 50
000030 PC127130 R3 001073 R4 006406 HT026554  ST00104 59
000030 PC127124 R3 001101 R4 007520 HT030515  ST00105 04
000030 PC127130 R3 001105 R4 005624 HT031537  ST00105 08
000030 PC127130 R3 007272 R4 010520 HT033645  ST00105 14
000030 PC127124 R3 001112 R4 007204 HT036526  ST00105 23
000030 PC127130 R3 007273 R4 010520 HT042676  ST00105 36
000030 PC127124 R3 001120 R4 007276 HT045207  ST00105 42
000030 PC127130 R3 007274 R4 010520 HT047460  ST00105 50
000030 PC127130 R3 007275 R4 010520 HT050336  ST00105 53
000030 PC127130 R3 007276 R4 010520 HT051215  ST00105 56
000030 PC127130 R3 007277 R4 010520 HT052073  ST00105 58
000030 PC127130 R3 001126 R4 007330 HT054507  ST00106 06
000030 PC127124 R3 001002 R4 006500 HT055324  ST00106 08
000030 PC127130 R3 000116 R4 121366 HT057370  ST00106 15
000030 PC127130 R3 000143 R4 121366 HT060657  ST00106 19
000030 PC127130 R3 000016 R4 113366 HT061434  ST00106 21
000030 PC127130 R3 000010 R4 113366 HT061743  ST00106 22
000030 PC127124 R3 000012 R4 113366 HT062662  ST00106 25
000030 PC127130 R3 000014 R4 113366 HT063602  ST00106.28
000030 PC127130 R3 000020 R4 113366 HT064007  ST00106.29
000030 PC127130 R3 000016 R4 113366 HT064522  ST00106.30
000030 PC127130 R3 000016 R4 113366 HT065337  ST00106 33
000030 PC127130 R3 000116 R4 121366 HT067176  ST00106 38
```

Figure A.9: Disk Activity Report for a FORTRAN Compilation

```
HIMOS LOG 12-7-76,18 03, FORTRAN COMPIL  EC30,24(40MS

000024 PC126532  ST00188 39
000030 PC127130 R3 000010  ST00194 01
000030 PC127130 R3 000012  ST00194 04
000024 PC127130  ST00194 05
000030 PC127124 R3 000014  ST00194 07
000024 PC127124  ST00194 08
000030 PC127124 R3 000020  ST00194 08
000030 PC127130 R3 000016  ST00194 10
000024 PC116574  ST00194 10
000030 PC127130 R3 000016  ST00194 12
000024 PC127130  ST00194 12
000030 PC127130 R3 000010  ST00194 13
000024 PC127130  ST00194 15
000030 PC127130 R3 000012  ST00194 16
000024 PC127124  ST00194 17
000030 PC127124 R3 000014  ST00194 19
000024 PC127130  ST00194 20
000030 PC127130 R3 000020  ST00194 20
000030 PC127130 R3 000016  ST00194 22
000024 PC127124  ST00194 22
000024 PC127130  ST00194 24
000030 PC127130 R3 001007  ST00194 27
000024 PC126572  ST00194 27
000024 PC127130  ST00194.29
000024 PC127130  ST00194 32
000030 PC127130 R3 006531  ST00194 33
000024 PC005616  ST00194 34
000024 PC006266  ST00194 36
000024 PC006260  ST00194 39
000024 PC127124  ST00194 41
000024 PC127130  ST00194 44
000024 PC127130  ST00194 46
000030 PC127130 R3 001014  ST00194 47
000024 PC127130  ST00194 48
000024 PC127130  ST00194 51
000030 PC127130 R3 001021  ST00194 53
000024 PC127130  ST00194 53
000024 PC127130  ST00194 56
000024 PC127130  ST00194 58
000030 PC127130 R3 007306  ST00195 00
000024 PC127124  ST00195.00
000024 PC127124  ST00195.03
000024 PC127130  ST00195 05
000024 PC127130  ST00195 08
000030 PC127130 R3 001026  ST00195.08
000024 PC127130  ST00195 10
000024 PC127124  ST00195 12
000030 PC127130 R3 007307  ST00195 14
000024 PC127124  ST00195.15
000024 PC127124  ST00195 17
000024 PC127130  ST00195 20
000030 PC127130 R3 001033  ST00195.21
```

Figure A.10: Sampling Session with 40 ms Periods (1)

```
HIMOS LOG 12-7-76,18 03. FORTRAN COMPIL  EC30,24(40MS)

000024 PC127130  ST00195 24
000024 PC127130  ST00195 27
000030 PC127124 R3 007310  ST00195 29
000024 PC127130  ST00195 29
000024 PC127124  ST00195 32
000024 PC127130  ST00195 34
000030 PC127130 R3 001040  ST00195 36
000024 PC006350  ST00195 36
000024 PC004422  ST00195 39
000024 PC127124  ST00195 41
000030 PC127130 R3 001045  ST00195.42
000024 PC127124  ST00195 44
000024 PC127130  ST00195.46
000030 PC127130 R3 001052  ST00195 48
000024 PC003556  ST00195.48
000024 PC127130  ST00195 51
000030 PC127130 R3 001060  ST00195 51
000024 PC002642  ST00195 53
000024 PC127124  ST00195 56
000030 PC127124 R3 001063  ST00195 56
000024 PC003426  ST00195.58
000030 PC127130 R3 001066  ST00195 59
000024 PC003720  ST00196 00
000024 PC004152  ST00196 03
000024 PC127130  ST00196.05
000030 PC127124 R3 001073  ST00196 08
000024 PC125612  ST00196 08
000024 PC127130  ST00196.10
000024 PC127130  ST00196.12
000030 PC127124 R3 001101  ST00196 14
000024 PC003356  ST00196 15
000030 PC127130 R3 001105  ST00196 17
000024 PC002426  ST00196 17
000024 PC127130  ST00196 20
000024 PC127124  ST00196 22
000030 PC127130 R3 007311  ST00196 24
000024 PC002676  ST00196 24
000024 PC127130  ST00196.27
000024 PC127130  ST00196.29
000024 PC127124  ST00196 32
000030 PC127124 R3 001112  ST00196 32
000024 PC004700  ST00196.34
000024 PC006204  ST00196 36
000024 PC127124  ST00196.39
000024 PC127124  ST00196.41
000024 PC127130  ST00196.44
000030 PC127130 R3 007312  ST00196.46
000024 PC127130  ST00196.46
000024 PC127130  ST00196.48
000024 PC127130  ST00196.51
000030 PC127130 R3 001120  ST00196.52
000024 PC005340  ST00196.53
000024 PC127130  ST00196.56
```

Figure A.10: Sampling Session with 40 ms Periods (2)

```
HIMOS LOG 12-7-76,18 03. FORTRAN COMPIL  EC30,24(40MS)


000030 PC127130 R3 007313  ST00197 00
000024 PC005340  ST00197.00
000030 PC127130 R3 007314  ST00197 03
000024 PC003506  ST00197.03
000030 PC127124 R3 007315  ST00197 05
000024 PC125612  ST00197 05
000024 PC127130  ST00197 08
000030 PC127124 P3 007316  ST00197.08
000024 PC127130  ST00197 10
000024 PC127124  ST00197 12
000024 PC127130  ST00197 15
000030 PC127130 R3 001126  ST00197 15
000024 PC127124  ST00197 17
000030 PC127124 R3 001002  ST00197 17
000024 PC127130  ST00197 20
000024 PC127130  ST00197 22
000030 PC127130 P3 000116  ST00197 24
000024 PC127130  ST00197 24
000024 PC127130  ST00197 27
000030 PC127130 P3 000143  ST00197 28
000024 PC127130  ST00197 29
000030 PC127124 R3 000016  ST00197 30
000030 PC127130 R3 000010  ST00197 31
000024 PC127130  ST00197.32
000030 PC127124 R3 000012  ST00197 34
000024 PC125612  ST00197 34
000024 PC127130  ST00197.36
000030 PC127124 R3 000014  ST00197 37
000030 PC127130 R3 000020  ST00197 38
000024 PC127130  ST00197.39
000030 PC127130 R3 000016  ST00197.40
000024 PC127124  ST00197.41
000030 PC127130 R3 000016  ST00197.42
000024 PC127124  ST00197 44
000024 PC127130  ST00197 46
000030 PC127130 R3 000116  ST00197 48
000024 PC122344  ST00197 48
000024 PC122344  ST00197.51
000024 PC124062  ST00197.53
000024 PC126534  ST00197.56
000024 PC122326  ST00197.58
000024 PC122440  ST00198.00
000024 PC122436  ST00198.03
000024 PC123366  ST00198.05
000024 PC122332  ST00198.08
000024 PC122444  ST00198.10
000024 PC124114  ST00198.12
000024 PC122326  ST00198.15
000024 PC126606  ST00198.17
000024 PC126614  ST00198.20
000024 PC122316  ST00198.22
000024 PC126604  ST00198.24
000024 PC126542  ST00198.27
```

Figure A.10: Sampling Session with 40 ms Periods (3)

```
*L
LOG MODE DEFINITION
H-LOG OR S-LOG SPECS (H,S) : S
OLD, NEW OR UPDATE (O,N,U) : UPD
NO LOG SPECIFIED
LOG RECORDING (Y,N)    : N? : Y
NUMERIC DISPLAY (Y,N) : N? : N
GRAPHIC DISPLAY (Y,N) : N? : N
SAVE FILE (Y,N) : N
CONT. OUTPUT ON OLD FILE (Y,N) : N
 OUTPUT FILE FOR LOG RECORDING
*LOG2=

*S
HOST TABLE UPDATE
SPEC FILE-OLD OR UPDATE (O,U) : U

MASK UPDATE (Y,N)? : Y
GIVE INDEX OR T
#1
7654321076543210
0000000000000000
0000000100010000
#T

SUBMASK UPDATE (Y,N)? :  N

SELECT WORD UPDATE (Y,N)? :  N

INTERR. FREQUENCY: 1              "=10KHZ"
TIMER COUNT       : 620           "DIV. BY 400 => 40MS SAMPLE RATE"
SAVE FILE (Y,N) : NO
*E
SP.EV. TABLE UPDATE:
GIVE EVENTS (# OR T)
#1
#2
#T
STOP ON HARD EVENT (Y,N) : N
*
SP.EV.:
000001 PC000007 PS000004 SP007124  ST00331.40
*C     "TO RELEASE HOST SYSTEM"
*
```

Figure A.11: Command Sequence for HIMOS Session of Fig. A.10

NAME	REF.	YEAR	PROPERTIES
SNUPER	[21]	1967	Intercept monitor (first phase)
GECOS II	[43]	1968	Event and sample driven; output on printer or tape.
CUE	[24]	1969	Sampling monitor; distributed by Bool and Babbage Inc.
SIPE	[44]	1969	Event driven; output on tape.
PPE	[24]	1970	Sampling monitor; distributed by Bool and Babbage Inc.
SPY	[45]	1970	Sampling monitor; output on paper tape or display; controlled by PDP-9.
MULTICS	[18]	1970	Event driven; graphic display; controlled by PDP-8.
SLACMON	[51]	1970	Sampling or event driven; monitors IBM/360 under OS/MVT.

Table A.1: Survey of Pure Software Monitors.

NAME	REF.	YEAR	PROPERTIES
--	[24]	1958	First hardware monitor by IBM for laboratory use.
POEM	[24]	1963	Program oriented event monitor
SNUPER	[21]	1967	Event driven, preset by host software (second phase).
SPAP	[27]	1967	Accumulating monitor, possibil. to interrupt host system.
SUM	[24]	1969	First commercially distributed hardware monitor.
NEUROTRON	[46]	1971	Minicomputer controlled; output on display and tape.
CPM-X	[34]	1972	Event driven; interaction with host software; minicomputer controlled.
DYNAPROBE	[47]	1972	Sampling and event driven; minicomputer controlled.
MICROSUM	[48]	1973	Accumulating monitor; real time display.
DSP1	[49]	1974	Accumulating and sampling; interaction with host software; output on display and tape.

Table A.2: Survey of Hardware and Hybrid Monitors

	HARDWARE	SOFTWARE	HYBRID
Host system degradation	none	medium to high	very low
Level of detail	medium	high	high
Operating sys. dependent	no	yes	less than soft. mon.
Training required	extensive	limited	extensive
Monitoring of op. sys. parameters	very diffic.	easy	easy
Software monit. capability	none	good	very good
Global system profile	good	medium	very good
Interrupt recognition	dep. on system	yes	yes
Device monitoring	yes	difficult	yes
Cost:			
Development	high	medium	high
Maintenance	high	low	high

Table A.3: Comparison of Monitoring Techniques.

FC	Vector	Event
Group 0:		
00	--	not used
01	04	Trap to 4
02	10	Trap to 10
03	14	BPT
04	20	IOT
05	24	Power fail
06	30	EMT
07	34	TRAP
10-17	--	not used
Group 1:		
20	300	Console Keyboard
21	304	Console Printer
22	70	Paper Tape Reader
23	74	Paper Tape Punch
24	104	KW-11P Clock
25-27	--	not used
30	220	RK 11 disk
31	240	PIRQ
32	244	FPP (floating point proc.)
33	250	MMU (memory management)
34-37	--	not used

Table A.4: Event Code Interpretation.

Event Masks:

bit	7	6	5	4	3	2	1	0	
high	-	-	-	-	-	-	-	-	GROUP0
low	TRAP	EMT	PWFL	IOT	BPT	T10	T4	-	
high	-	-	-	-	MMU	FPP	PIRQ	RK	GROUP1
low	-	-	-	KW	PP	PR	TP	TK	

Submasks:

bit	7	6	5	4	3	2	1	0	
					SM2				
high	-	SP	PS	PC	SP	PS	PC	SP	PWFL, IOT
low	PS	PC	SP	PS	PC	SP	PS	PC	BPT,T10,T4
					SM1				
high	-	-	-	[PC-2]	SP	PS	PC-2	SEL	TRAP
low	-	-	ERC	[PC-2]	SP	PS	PC-2	SEL	EMT
					SM2				
high	-	-	-	[REG]	SP	PS	PC	SEL	TP
low	-	-	[BUF]	[REG]	SP	PS	PC	SEL	TK
					SM3				
high	-	-	-	[REG]	SP	PS	PC	SEL	PP
low	-	-	[BUF]	[REG]	SP	PS	PC	SEL	PR
					SM4				
high	-	-	-	-	-	-	-	-	
low	-	-	[BUF]	[REG]	SP	PS	PC	SEL	KW-11 CLOCK
					SM5				
					- not used -				
					SM6				
high	-	-	-	-	-	-	R5	R4	RK 11 disk
low	R3	R2	R1	R0	SP	PS	PC	SEL	controller

Select Words

Selw. 0-3 : 7 EMT codes, one 0 as endmarker

Selw. 4-10: 7 TRAP codes, one 0 as endmarker

Selw.11-20: 4 pairs of the form: Reg. number
Reg. value

Table A.5: Event Specifications

ID	EXPLANATION
PC	Program Counter
PS	Processor Status word
SP	Stack Pointer (=R7)
CP	Content of Program counter (=op. code for 1 wrd instr.)
EW	Error Word (=content of #52) for EMT calls
SR	Device Status Register
BF	Device Buffer Content
Rn	Register n of disk controller
HT	Host Time (content of host clock buffer)
ST	Supervisor Time (in sec. relative to start of session)

Table A.6: Value Identification for Numeric Display

CMD	SUBPROG	EXPLANATION
W	WAITUP	Terminal output is disabled
P	PROC	Terminal output is enabled after W
E	SPEVUP	Special event table update mode
C	CONT	Continue after special event occurrence
B	INPICI	Initialize histogram, draw coordinates
A	DRPICI	Erase Screen and draw histogram as it was
D	CNTOUT	Erase screen and print event summary report
S	CSIS	CMD interpreter for S-MON mode definition
H	CSIH	CMD interpreter for H-MON mode def. (not impl.)
L	CSIL	CMD interpreter for logging mode definition
T	TERM	Terminate session and print summary report

Table A.7: Control System Commands

```
S-MON:: INTERACTIVE SOFTWARE MO RT-11 MACRO VM02-09    00111126 PAGE 1

                                        .TITLE S-MON:: INTERACTIVE SOFTWARE MONITOR
                                ;       ********************************************
                                ;*
                                ;*************************************************************
                                ;*
                                ;* THIS PROGRAM IS PART OF THE MONITOR SYSTEM "MIMOS".
                                ;* THE FIRST PART IS NONRESIDENT AND ONLY USED FOR INITIALISATION.
                                ;* THE SECOND PART IS RELOCATED ABOVE RT11 STARTING AT 134000
                                ;* FROM WHERE SYSTEM ACTIVITIES ARE MONITORED.
                                ;* S-MON IS A PURE INTERCEPT TYPE MONITOR WHICH CAN ANALYZE
                                ;* ANY VECTORED EVENT.
                                ;* PART ONE STARTS AT "START"
                                ;* PART TWO STARTS AT "STRES"
                                ;*
                                ;*************************************************************
                                ;
                                        .MCALL .PRINT,.REGDEF,.EXIT,.QUTBUF,.DEF,.SEND,.RECV,.DEF
                                        .MCALL .TTYIN,.TTYOUT
000000                                  .DEF
000000                                  .REGDEF
                                        .ASECT
000000                                  .=1000                  ;LOAD POINT
                                        .TITLE CONSTANT DEFINITIONS
                                ;       ***************************
                                ;
000001                                  M=1                     ;M+1 GROUPS
000004                                  MAXAM=M*4
000040                                  N=16.**(M+1)            ;N = TOTAL NUMBER OF TESTED VECTORS
000004                                  EOT=4
000006                                  ACK=6
000340                                  BR7=340
177776                                  PSW=177776
177564                                  TTYOUT=177564           ;TTY OUTPUT STATUS
172542                                  TBUF=172542             ;TIMER CNT BUFFER
172540                                  TSTAT=172540            ;TIMER STATUS REGISTER
000320                                  PLIV=320                ;VECTOR FOR PARALLEL LINK
164000                                  PLSR=164000 *           ;STATUS REG.  -"-
164002                                  PLOB=164002             ;OUTPUT BUF.  -"-
164004                                  PLIB=164004             ;INPUT BUF.   -"-
164000                                  IIS=PLSR                ;REDEFINITIONS FOR PAR. LINK
164000                                  IOS=PLSR
164004                                  IIB=PLIB
164002                                  IOB=PLOB
000320                                  IIV=PLIV
000324                                  IOV=PLIV+4
164000                                  COMCHI=IIS              ;STATUS REG. OF COMM. CHANNEL IN
164000                                  COMCHO=IOS
134000                                  BEGIN=134000            ;START ADDR. OF RES. ROUTINE
134000                                  NEWSTK=BEGIN            ;SP DURING MONITORING
137720                                  DIST=BEGIN-STRES        ;RELOCATION FACTOR
                                ;
                                ;RELOCATED VARIABLES
                                ;
136402                                  XBLANK=BLANK+DIST
137420                                  XAST=AST+DIST
136440                                  XCRLF=CRLF+DIST         ;DELIMETER FOR PRINT
140012                                  XREGR=REGG+DIST         ;SAVE LOC FOR R6
```

```
                                        XREGS=REGS+DIST         ;SAVE LOC FOR R5
                                        XTXTBF=TXTBUF+DIST
                                        XENTR0=ENTRY0+DIST      ;ENTRY ADDR, FOR GROUP 0
                                        XENTR1=ENTRY1+DIST      ;ENTRY ADDR, FOR GROUP 1
                                        XENTR2=ENTRY2+DIST      ;ENTRY ADDR, FOR SUPERVISOR
                                        XPSSAV=PSWSAV+DIST      ;TO SAVE I NUMBER
                                        XVECSV=VECSAV+DIST      ;SAVE TABLE FOR VECTORS
                                        XVECTB=VECTAB+DIST      ;VECTOR TABLE
                                        XMSK0=MASK0+DIST        ;FIRST MASK WORD
                                        XREPL=REPLACE+DIST      ;REPLACEMENT PROCEDURE
                                        XDIR=DIRECT+DIST        ;TO=ORIG, TO SAVE,1=SAV TO ORIG
                                        XQUES=QUES+DIST         ;?
                                        XMPC=MPC+DIST           ;!
                                        XMPS=MPS+DIST           ;"
                                        XMSP=MSP+DIST           ;#
                                        XMLPC=MLPC+DIST         ;$
                                        XM52=M52+DIST           ;%
                                        XS0060=SL0060+DIST      ;SELECT WORD
                                        XDE=DE+DIST             ;RELOC. HANDLER ADDRESSES
                                        XSG00=SG00+DIST
                                        XSG01=SG01+DIST
                                        XSG10=SG10+DIST
                                        XSG11=SG11+DIST
                                        XSG12=SG12+DIST
                                        XTABUP=TABUPD+DIST
                                        XHANG=HANG+DIST
                                        XDSPT0=DISPT0+DIST      ;MAIN DISPATCH TABLE
                                        XDSPT2=DISPT2+DIST      ;SUPERVISOR CMD DISPATCH
                                        XS1100=SL1100+DIST
                                        XMSK01=MSK101+DIST
                                        XDSTAT=DVSTAT+DIST
                                        XRTII=RTII+DIST
                                        XMM005=MSKPC5+DIST
                                        XENDRU=ENDRUN+DIST
                                        XENDCM=ENDCOM+DIST
                                        XCONT=CONT+DIST
                                        XTIMON=TIMEON+DIST
                                        XTINT1=TINT1+DIST
                                        XTINT2=TINT2+DIST
                                        XTREG1=TREG1+DIST
                                        XTREG2=TREG2+DIST
                                ;
                                ;
                                        .TITLE MACRO DEFINITIONS
                                        ;***********************
                                ;
                                ;
                                        .MACRO  .DCOUT
                                        ;JSR PC,DPRINT
                                        JSR PC,DCPUT
                                        .ENDM
                                ;
                                        .MACRO  .CHOUT
                                        ;JSR PC,PRINT
                                        JSR PC,CHPUT
                                        .ENDM
                                ;
```

```
                        .MACRO .SNDWRD          ;TRANSMIT ONE WORD TO SUPERV.
                        TST @#PLSR              ;WAS PRVIOUS WORD RECEIVED?
                        BMI .-4                 ;WAIT IF NOT
                        MOV R0,@#PLOB           ;SEND WORD IN R0
                        .ENDM

                        .MACRO .RCVWRD          ;RECEIVE ONE WORD FROM SUPERV.

                        TSTB @#PLSR             ;IS THERE A NEW WORD
                        BPL .-4                 ;WAIT IF NOT
                        MOV @#PLIB,R0           ;STORE WORD IN R0
                        .ENDM

                        .MACRO .BUFOUT          ;TRANSMISSION OF ONE EVENT RECORD
                        ;JSR PC,PRNTBF
                        JSR PC,SENDBF
                        BIT #2,R4               ;ENDMARKER SET?
                        .ENDM
```

```
                                        .TITLE I/O SUBROUTINES - NON RESIDENT
                                ;       ************************************
                                ;
                                ;*****
001000                          PRINT:  ;PRINT TEXT STARTING AT (R0)
                                ;*****
001000                                  .PRINT
001002  105737  177564          PRINT3: TSTB @#TTYOUT
001006  100375                          BPL PRINT3
001010  000207                          RTS PC                  ;RETURN FROM PRINT
                                ;
                                ;*****
001012                          OCRD:   ;READ OCTAL NUMBER; LINKAGE: PC
                                ;*****  ;R0=-3 IF T, -2 IF EMTY; RESULT IN R1
                                ;
001012  010346                          MOV R3,-(SP)
001014  005001                          CLR R1          ;RESULT BUFFER
001016  005003                          CLR R3                  ;SHIFT COUNTER
001020  005046                          CLR -(SP)               ;START MARKER ON TOP
001022                          OCRD1:  .TTYIN
001026  042700  000200                  BIC #200,R0
001032  122700  000012                  CMPB #12,R0
001036  001402                          BEQ OCRD3
001040  010046                          MOV R0,-(SP)    ;STACK NUMBER
001042  000767                          BR OCRD1
001044  005726                  OCRD3:  TST (SP)+
001046  005716                          TST (SP)                ;TOP OF STACK?
001050  001416                          BEQ OCRD5       ;YES
001052  022716  000124                  CMP #124,(SP)   ;T?
001056  001410                          BEQ OCRD4
001060  162716  000060                  SUB #60,(SP)    ;CONVERT
001064  011600                          MOV (SP),R0
001066  072003                          ASH R3,R0               ;SHIFT NB N TIMES LEFT
001072  050001                          BIS R0,R1
001074  062703  000003                  ADD #3,R3       ;INCR SHIFT CNT
001100  000762                          BR OCRD3
001102  012700  177777          OCRD4:  MOV #-1,R0              ;-3 IF T
001106  000757                          BR OCRD5
001110  005703                  OCRD5:  TST R3
001112  001002                          BNE OCRD6
001114  052700  177775                  BIS #177775,R0          ;-2 IF EMPTY
001120  005726                  OCRD6:  TST (SP)+
001122  012603                          MOV (SP)+,R3
001124  000207                          RTS PC          ;RETURN FROM OCIN
                                ;
                                ;*****
001126                          PRTBYT: ;PRINT BYTE OCTAL; LINKAGE: PC
                                ;*****  ;VALUE IN R0
                                ;
001126  010146                          MOV R1,-(SP)
001130  010246                          MOV R2,-(SP)
001132  042700  177400                  BIC #177400,R0          ;CLEAR UPPER BYTE
001136  012701  000003                  MOV #3,R1               ;PRINT 3 DIGITS
001142  000404                          BR TYP1
                                ;*****
001144                          PRTWRD: ;PRINT WORD OCTAL; LINKAGE: PC
                                ;*****  ;VALUE IN R0
```

```
                58
                59 001142  010146                  ;
                60 001144  010246                          MOV R1,-(SP)
                61 001146  012701  000006                  MOV R2,-(SP)
                62 001152  010102                          MOV #6,R1               ;PRINT 6 DIGITS
                63 001154  010246          TYP1:           MOV R1,R2               ;SAVE COUNT
                64 001156  042716  177770  TYP2:           MOV R0,-(SP)
                65 001162  000241                          BIC #177770,(SP)
                66 001164  006000                          CLC
                67 001166  006200                          ROR R0
                68 001170  006200                          ASR R0
                69 001172  005301                          ASR R0
                70 001174  001367                          DEC R1
                71 001176  012600          TYP3:           BNE TYP2
                72 001200  005302                          MOV (SP)+,R0
                73 001202  001405                          DEC R2                  ;LAST DIGIT?
                74 001204  005701                          BEQ TYP4                ;YES
                75 001206  001002                          TST R1                  ;ANY TYPED YET?
                76 001210  005720                          BLE TYP4                ;YES
                77 001212  001406                          TST R0                  ;NO,LEADING 0?
                78 001214  062700  000060                  BEQ TYP5                ;YES, SKIP
                79 001220                  TYP4:           ADD #60,R0
                80 001224  104001                          .TTOUT
                81 001226  005201                          WAIT
                82 001230  005702                          INC R1
                83 001232  001361          TYP5:           TST R2                  ;DONE?
                84 001234  012602                          BNE TYP3                ;NO
                85 001236  012601                          MOV (SP)+,R2
                86 001240  000207                          MOV (SP)+,R1
                87                                        RTS PC                  ;RETURN FROM PRTRYT/WRD
                88                         ;
                89 001242                  ;+++++
                90                         BINWD:  ;READ WORD BINARY; LINKAGE: PC
                91                         ;+++++  ;WORD ADDR: R1
                92 001242  010546                  ;
                93 001244  010246                          MOV R5,-(SP)
                94 001246  005002                          MOV R2,-(SP)
                95 001250  005003                          CLR R2
                96 001252                                 CLR R3
                97 001254  042780  000200  BINWD1:        .TTYIN
                98 001262  122740  000015                  BIC #200,R0
                99 001266  001410                          CMPB #15,R0
               100 001270  005245                          BEQ BINWD3
               101 001272  006302                          INC R5
               102 001274  122700  000060                  ASL R2
               103 001300  001364                          CMPB #60,R0
               104 001302  052702  000001                  BEQ BINWD1
               105 001306  000761                          BIS #1,R2
               106 001310  005705                          BR BINWD1
               107 001312  001401  BINWD3:        TST R5
               108 001314  010211                          BEQ BINWD4              ;EMPTY WORD
               109 001316                                 MOV R2,(1)              ;STORE NEW WORD
               110 001322  012602  BINWD4:        .TTYIN                          ;EMPTY RING BUFFER
               111 001324  012605                          MOV (SP)+,R2
               112 001326  000207                          MOV (SP)+,R5
               113                                       RTS PC                  ;RETURN FROM BINWD
               114                         ;
                                           ;+++++
```

```
                                ;
                                ;
001330                  BINPRT: ;PRINT WORD BINARY; LINKAGE: PC
                        ;*****  ;WORD ADDR: R1
                                ;
001330  010146                 MOV R1,-(SP)
001332  012700  002453          MOV #BITPOS,R0          ;PRINT BIT IDENTIFIER STRING
001336  004767  177436          JSR PC,PRINT
001342  011104                 MOV (1),R4              ;WORD TO BE PRINTED
001344  012701  002474          MOV #BITS,R1
001350  032704  100000  BINPT1: BIT #100000,R4         ;CHECK LEFTMOST BIT
001354  001403                 BEQ BINPT2
001356  112721  000061          MOVB #61,(1)+
001362  000402                 BR BINPT3
001364  112721  000060  BINPT2: MOVB #60,(1)+
001370  006304  BINPT3: ASL R4
001372  020127  002514          CMP R1,#BITS+16.
001376  001364                 BNE BINPT1
001400  012700  002474          MOV #BITS,R0
001404  004767  177370          JSR PC,PRINT
001410  012601                 MOV (SP)+,R1
001412  000207                 RTS PC                  ;RETURN FROMBINPRT
                                ;
001414  012706  001000  START:  MOV #1000,SP           ;*****ENTRY POINT *****
001420  005737  164004          TST @#177564            ;CLEAR INPUT BUFFER
                                ;
```

```
                        .TITLE SPECIFICATION OF S-MON MODE
                        ;*********************************
                        ;
                        ;*********************
                        ;COMMAND INTERPRETER *
                        ;*********************
                        ;
001424  005005          CMDI0:  CLR R5                  ;SET LOOP SWITCH
                        ;
001426                  CMDI0:  ;*** MASK UPDATE ***
                        ;
001426  012700  002515          MOV #MSKIT,R0
001432  004767  177542          JSR PC,PRINT
                        ;
001436                  CMDI1:  ;*** Y/N INTERPRETER ***
                        ;
001436                          .TTYIN                  ;READY OR NO WITH LEADING GARBAGE
001442  042700  000200          BIC #200,R0
001446  122700  000015          CMPB #15,R0
001452  001402                  BEQ CMDI2
001454  010001                  MOV R0,R1               ;SAVE CHAR
001456  000767                  BR CMDI1
001460                  CMDI2:  .TTYIN                  ;EMPTY RING BUFFER
001464  122701  000116  CMDI3:  CMPB #116,R1           ;N?
001470  001002                  BNE CMDI4               ;NO
001472  000175  003024          JMP @CMDT1(5)           ;DISPATCH
001476  122701  000131  CMDI4:  CMPB #131,R1           ;Y?
001502  001006                  BEQ CMDI5               ;YES
001504  012700  002671          MOV #CMDERR,R0
001510  004767  177264          JSR PC,PRINT            ;PRINT ERROR
001514  000175  003052          JMP @CMDT2(5)           ;DISPATCH
                        ;
001520                  CMDI5:  ;*** INDEX INPUT ***
                        ;
001520  012700  002545          MOV #IREQ,R0            ;REQUEST INDEX
001524  004767  177250          JSR PC,PRINT
001530  012703  002663  CMDI6:  MOV #CMDSS+2,R3
001534  004767  177246          JSR PC,PRINT
001540  004767  177246          JSR PC,GCHR             ;READ INDEX INTO R1
001544  005700                  TST R0                  ;TZ
001546  002402                  BLT CMDI7               ;YES
001550  000175  003046          JMP @CMDT4(5)           ;ERR-DISPATCH FOR VALUE OUTPUT
001554  000175  003040  CMDI7:  JMP @CMDT3(5)           ;ERR-DISPATCH FOR NEXT GROUP
                        ;
001560                  CMDI8:  ;*** MASK I/O ***
                        ;
001560  006301                  ASL R1                  ;INDEX*4
001562  006301                  ASL R1
001564  062701  006240          ADD #MASKS,R1           ;ADDR. OF MASK
001570  004767  177554          JSR PC,BINPRT           ;PRINT MASK
001574  004767  177442          JSR PC,BINRD            ;READ NEW MASK
001600  000751                  BR CMDI6                ;READ NEXT INDEX
                        ;
001602                  CMDI9:  ;*** SUBMASK I/O ***
                        ;
001602  006301                  ASL R1                  ;INDEX*2
001604  062701  006254          ADD #MSK005,R1          ;ADDR. OF SUBMASK
```

```
 58 001610 004767 177514          JSR PC,GINPRT          ;PRINT MASK
 59 001614 004767 177422          JSR PC,GINRD           ;READ NEW MASK
 60 001620 000743                 BR CMDI6               ;READ NEXT INDEX
 61                               ;
 62 001622                        CMDI10: ;*** SUBMASK UPDATE ***
 63                               ;
 64 001622 005725                 TST (5)+               ;INCR. LOOP SWITCH
 65 001624 012700 002565          CMDI11: MOV #SMSKIN,R0
 66 001630 004767 177144          JSR PC,PRINT
 67 001634 000167 177576          JMP CMDI1              ;READ Y/N
 68                               ;
 69 001640                        CMDI12: ;*** SELECT WORD UPDATE ***
 70                               ;
 71 001640 005725                 TST (5)+               ;INCR. LOOP SWITCH
 72 001642 012700 002621          CMDI13: MOV #SELWIN,R0
 73 001646 004767 177120          JSR PC,PRINT
 74 001652 000167 177560          JMP CMDI1              ;READ Y/N
 75                               ;
 76 001656                        CMDI14: ;*** SELECT WORD I/O ***
 77                               ;
 78 001656 006301                 ASL R1                 ;INDEX*2
 79 001660 010102                 MOV R1,R2              ;SAVE INDEX
 80 001662 022701 000020          CMP #20,R1             ;INDEX>10?
 81 001666 002424                 BLT CMDI16             ;YES=GOTO WORD I/O
 82 001670 012703 000002          MOV #2,R3              ;PROCESS 2 BYTES
 83                               ;
 84 001674                        CMDI15: ;*** BYTE I/O ***
 85                               ;
 86 001674 005000                 CLR R0
 87 001676 116200 006300          MOVB SL0060(2),R0      ;GET OLD BYTE VALUE
 88 001702 004767 177216          JSR PC,PRTBYT          ;PRINT IT
 89 001706 012700 002665          MOV #GAP,R0
 90 001712 004767 177062          JSR PC,PRINT           ;PRINT 2
 91 001716 004767 177070          JSR PC,GCHR            ;GET NEW VALUE
 92 001722 005700                 TST R0
 93 001724 100402                 BMI CMDI5A             ;0 NEW VALUE
 94 001726 110162 006300          MOVB R1,SL0060(2)      ;STORE IT
 95 001732 005202                 CMDI5A: INC R2
 96 001734 077321                 SOB R3,CMDI15          ;SECOND BYTE
 97 001736 000674                 BR CMDI6
 98                               ;
 99 001740                        CMDI16: ;*** WORD I/O ***
100                               ;
101 001740 016200 006300          MOV SL0060(2),R0       ;GET OLD WORD VALUE
102 001744 004767 177172          JSR PC,PRTWRD          ;PRINT IT
103 001750 012700 002665          MOV #GAP,R0
104 001754 004767 177020          JSR PC,PRINT
105 001760 004767 177026          JSR PC,GTWRD           ;GET NEW VALUE
106 001764 005700                 TST R0
107 001766 100660                 BMI CMDI6              ;NO NEW VALUE
108 001770 010162 006322          MOV R1,SL1100(2)       ;STORE IT
109 001774 000655                 BR CMDI6
110                               ;
111 001776                        CMDI17: ;*** TIMER CONTROL ***
112                               ;
113 002000 005067 000342          CLR TIMLG              ;0: NO TIME LOGGING
114 002004 005067 000334          CLR TIMLOP             ;0: NO TIME INTERRUPT
```

```
115 002026 005067 004334          CLR TMEG1
116 002032 032767 000020 004230   BIT #20,MASK1          ;TIME INTERRUPT?
117 002040 001420                 BEQ CMDI19            ;NO-ASK FOR TIMING
118                               ;
119                               ;       *** TIME INTERR, MODE ***
120                               ;
121 002042 012700 002703          MOV #TIMST1,R0        ;GET FREQUENCY
122 002046 004767 176706          JSR PC,PRINT
123 002052 004767 176754          JSR PC,GCHR           ;READ 0-3
124 002056 006301                 ASL R1
125 002060 052761 000010          BIS #10,P1            ;SET REF. MODE DOWN
126 002066 052167 004276          BIS R1,TMEG1
127 002072 005267 004266          INC TIMEON            ;SET, INTERR. AFTER EXEC
128 002076 012700 002777  CMDI18: MOV #TIMCNT,R0        ;GET COUNT
129 002102 004767 176714          JSR PC,PRINT
130 002106 004767 176722          JSR PC,GCHR
131 002112 010167 004250          MOV R1,TMEG2          ;SET CNT
132 002116 000421                 BR CMDI20             ;END OF INTRR, TIMING
133                               ;
134 002120                 CMDI19: ;*** TIME LOG MODE ***
135                               ;
136 002120 012700 002731          MOV #TIMST2,R0        ;GET FREQUENCY
137 002124 004767 176672          JSR PC,PRINT
138 002130 004767 176730          JSR PC,GCHR
139 002134 022701 000003          CMP #3,R1             ;>3?
140 002140 103410                 BHT CMDI20            ;YES-NO TIME LOG
141 002142 006101                 ASL R1
142 002144 052761 000031          BIS #31,R1            ;START REF. MODE UP
143 002152 052167 004214          BIS R1,TMEG1
144 002160 005267 004206          INC TIMLG             ;SET TIME LOG MODE
145 002164 000746                 BR CMDI18             ;LOG, GET CNT
146                               ;
147 002166                 CMDI20: ;*** TRAP INIT ***
148                               ;
149 002166 012737 140036 000034   MOV #XRTII,##34      ;YES-SET RTI ADDR. IN VECTOR
150 002174 005037 000036          CLR ##36
151                               ;
152 002200                 CMDI21: ;*** VECTOR PROTECTION ***
153                               ;
154 002200 013700 000054          MOV #*54,R0
155 002204 052760 177777 000526   BIS #177777,526(0)     ;TRAP VECT. PROTECTION
156 002212 052760 177777 000532   BIS #177777,532(0)     ;CLOCK VECT. PROTECTION
157 002220 052760 177777 000302   BIS #177777,302(0)     ;PROM, CH. VECT. PROTECTION
158 002226 012737 000240 146762   MOV #240,##146762      ;CHANGE SYS. RESET TO NOP
159                               ;
160                               ;
```

```
                                        .TITLE INITIALISATION PROGRAM
                                ;       ****************************
                                ;
002206  012701  134000                  MOV #134000,R1
002212  162701  003054                  SUB #STRES,R1
002216  012700  002424                  MOV #TXT1,R0
002222  004767  176552                  JSR PC,PRINT
002226  010100                          MOV R1,R0
002230  004767  176700                  JSR PC,PRTORD
                                ;
002234  005037  000024          POWFL:  CLR @#24                ;SET POW.FAIL VECT
                                ;
002240  012737  134046  000320          MOV #XFNTR2,@#IIV       ;SET SUPER INTERR. VECT.
002246  012737  000340  000322          MOV #BR7,@#IIV+2
002254  052737  000100  164000          BIS #100,@#IIS          ;ENABLE SUPER INTERR.
002262  005767  004054                  TST TIMEON              ;TIMF INTERR. MODE?
002266  001406                          BEQ INIT1               ;NO
002270  016737  004056  000104          MOV TINT1,@#104
002276  016737  004052  000106          MOV TINT2,@#106
002304  012767  157046  003730  INIT1:  MOV #XTXTBF,TXTPTR      ;SET TEXT POINTER
                                ;
002312  012700  003054          RELOC:  MOV #STRES,R0           ;RELOCATE PROGRAM AND DATA
002316  012701  134000                  MOV #BEGIN,R1           ;STARTING AT BEGIN
002322  012702  007114                  MOV #ENDMON,R2          ;UNTIL ENDMON
002326  012021          AGN2:   MOV (R0)+,(R1)+
002330  020002                  CMP R0,R2
002332  001375                  BNE AGN2
002334  005037  140030                  CLR @#XBGIN
002340  004767  002730                  JSR PC,EXCH             ;EXCHANGE TRACED VECTORS
002344  016737  000000  172542          MOV TREG2,@#TBUF
002352  016737  003770  172540          MOV TREG1,@#TSTAT
002360  042737  000100  164000          BIC #100,@#CONCHO
002366  052737  000100  164000          BIS #100,@#CONCHI       ;ENABLE COMM. CH.
002374  000240                          NOP
002376  000240                          NOP
002400  000240                          NOP
002402  012700  002402                  MOV #.,R0               ;SET R0=P/C FOR .EXIT
002406  005767  003730                  TST TIMEON              ;CLOCK?
002412  001403                          BEQ FIN                 ;NO TIME INTERRUPTS
002414  052737  000101  172540          BIS #101,@#TSTAT        ;START CLOCK INTERR. MODE
002422                          FIN:    .EXIT
                                ;
                                ;
                                ;
                                ;       NON RESIDENT DATA SECTION
                                ;
                                        .NLIST BEX
002424  117     106     106     TXT1:   .ASCII/OFFSET /
002434  200                             .BYTE 200
002435  015     012             TXT2:   .BYTE 15,12
002437  110     111     107             .ASCII/HIGH LIMIT /
002452  200                             .BYTE 200
002453  067     066     065     BITPOS: .ASCIZ /76543210/
002474  060     060     060     BITS:   .ASCIZ /000000000000000/
002515  015     012     115     MSKIN:  .ASCII <15><12>/MASK UPDATE (Y,N)? : /<200>
002545  107     111     126     INEG:   .ASCII /GIVE INDEX OR 1/<000>
002565  015     012     123     SMSKIN: .ASCII <15><12>/SUBMASK UPDATE (Y,N)? :  /<200>
```

```
                                                    .ASCII <15><12>/SELECT #CMD UPDATE (Y,N)? $  /<200>
                                                    .ASCII <15><12>/#/<200>
                                                    .ASCII / : /<200>
                                                    .ASCII /?ILL CMD?/
                                                    .ASCII <15><12>/INTERR. FREQUENCY$ /<200>
                                                    .ASCII <15><12>/TIMER FREQUENCY (>3: NO TIME LOG): /<200>
                                                    .ASCII /TIMER COUNT      : /<200>
                                                    .EVEN
                                                    CMDI10                  ;CMD DISP. TABLE 1:
                                                    CMDI12                  ;NEXT GROUP AFTER 'N' INPUT
                                                    CMDI17
                                                    CMDI0                   ;CMD DISP. TABLE 2:
                                                    CMDI11                  ;ERROR DISP. AFTER WRONG Y/N INPUT
                                                    CMDI13
                                                    CMDI10                  ;CMD DISP. TABLE 3:
                                                    CMDI12                  ;DISP. AFTER I AT INDEX INPUT
                                                    CMDI17
                                                    CMDI8                   ;CMD DISP. TABLE 4:
                                                    CMDI9                   ;I/O FOR NEW INDEX
                                                    CMDI14
                                                ;
                                                    .LIST BEX
                                                    .NLIST ME
                                                ;
```

```
                                .TITLE RESIDENT ROUTINES
                        ;       ************************
                        ;
003054  000240          STRES:  NOP                     ;START OF RESIDENT ROUTINE
                        ;
                        ;*****************************
                        ;INTERCEPT ENTRY FOR GROUP 0
                        ;*****************************
                        ;
003056  013737  177776  140026  ENTRY0: MOV @#PSW,@#XPSSAV    ;SAVE PSW WITH I-CODE
003064  010767  000020          MOV PC,RETADD           ;TO SIMULATE JSR
003070  000426                  BR SAVE
003072  012700  000000          MOV #0,R0               ;INDICATE GROUP INDEX
003076  000442                  BR DISP
                        ;
                        ;*****************************
                        ;INTERCEPT ENTRY FOR GROUP 1
                        ;*****************************
                        ;
003100  013737  177776  140026  ENTRY1: MOV @#PSW,@#XPSSAV    ;SAVE PSW WITH I-CODE
003106  010767  003776          MOV PC,RETADD           ;TO SIMULATE JSR
003112  000415                  BR SAVE
003114  012700  000001          MOV #1,R0               ;INDICATE GROUP INDEX
003120  000431                  BR DISP
                        ;
                        ;*****************************
                        ;SUPERVISOR INTERRUPT ENTRY*
                        ;*****************************
                        ;
003122  013737  177776  140026  ENTRY2: MOV @#PSW,@#XPSSAV   ;SAVE PSW
003130  042737  000100  164000          BIC #100,@#COMCHI       ;DISABLE SUP. INTERR.
003136  010767  003746          MOV PC,RETADD           ;TO SIMULATE JSR
003142  000401                  BR SAVE
003144  000520                  BR EXDISP               ;GOTO DISP. FOR EXT EV.
                                ;
                                ;
                        ;
003146  000240          SAVE:   NOP
003150  010567  003724          MOV R5,REGS             ;SAVE GPR'S
003154  012705  140012          MOV #XREGS,R5
003160  010025                  MOV R0,(R5)+
003162  010125                  MOV R1,(R5)+
003164  010225                  MOV R2,(R5)+
003166  010325                  MOV R3,(R5)+
003170  010425                  MOV R4,(R5)+
003172  062767  000004  003710  ADD #4,RETADD           ;GO BEYOND OR INSTR
003200  000177  003704          JMP @RETADD             ;SIMULATE RTS
                        ;
```

```
                        .TITLE EVENT HANDLER DISPATCH
                        ;*****************************
                        ;
                        ;
003204  000240          DISP:   NOP
003206  005004                  CLR R4                  ;FLAG REGISTER
                        ;
                        ;
003210  010601                  MOV SP,R1
003212  042701  100000          BIC #100000,R1          ;CLEAR SIGN BIT
003216  020127  000500          CMP R1,#500             ;EVENT STK <=500?
003222  100006                  BPL .SVSTK              ;NO-PROTECT VITAL VECTORS
003224  052704  000001          BIS #1,R4               ;FLAG SP MODIFICATION
003230  012701  001000          MOV #1000,R1            ;YES - SET IT TO 1000
003234  010106                  MOV R1,SP               ;SAVE IT AS OLDSTK
003236  005011                  CLR (R1)                ;SIMULATE EVENT PC=0
003240  010667  003642  SVSTK:  MOV SP,OLDSTK           ;SAVE STACK
003244  012706  134000          MOV #NEWSTK,SP          ;SET NEW STACK
                                )
                        ;
003250  072027  000004          ASH #4,R4               ;SHIFT GROUP NO. 3 4 LEFT
003254  042737  177760  140026  BIC #177760,@#XPSSAV   ;MASK I-NUMBER
003262  063700  140026          ADD @#XPSSAV,R4         ;COMPUTE EVENT CODE*16+J*I
003266  010005                  MOV R5,R5               ;SAVE R4 DURING T/O
003270  000240                  NOP
003272  010037  177570          MOV R0,@#177570         ;TEST
                        ;
                        ;
003276  006300                  ASL R0                  ;MULT EV CODE BY 2
003300  004770  137304          JSR PC,@XDSPTB(0)       ;CALL SUBGROUP HANDLER
003304                          .BUFOUT
                        ;
                        ;
003314  000240  RESTOR: NOP
003316  006305                  ASL R5
003320  006305                  ASL R5
003322  010500                  MOV R5,R4               ;R4= EVENT CODE
003324  016701  003556          MOV OLDSTK,R1           ;OLD STACK ADD
003330  162701  000004          SUB #4,R1               ;PUSH FOR ADDITIONAL VECTOR
003334  010167  003546          MOV R1,OLDSTK           ;SAVE STACK ADD FOR RTI
003340  016021  137412          MOV XVECSV(R4),(R1)+    ;PUT SAVED VECTOR ON STACK
003344  005720                  TST (R0)+
003346  016011  137412          MOV XVECSV(R4),(R1)
003352  004767  000350          JSR PC,UNSAVE
003356  016706  003524          MOV OLDSTK,SP           ;RESET SP
003362  000240                  NOP
003364  105767  002530          TSTB TRAP
003370  001405                  BEQ RESTR1
003372  105067  002522          CLRB TRAP
003376  012700  003376          MOV #.,R0               ;SET R0=PC FOR .EXIT
003402  104350                  EMT 350
003404  000002  RESTR1: RTI                   ;RETURN TO SYSTEM
                        ;
```

```
                                        .TITLE CONTROL COMMAND HANDLER
                                ;       ******************************
                                ;
                                ;*****
003406                          EXDISP: ;DISPATCHER FOR SUPERVISOR COMMANDS
                                ;*****
                                ;
003406  005004                          CLR R4
003410  010667  003472                  MOV SP,OLDSTK           ;SAVE STACK
003414  012706  134000                  MOV #NEWSTK,SP          ;SET NEW STACK
003420                                  .RCVXRD                 ;GET FUNCTION CODE INTO R0
003452  012737  000777  177570          MOV #777,@#177570       ;TEST
003440  004770  137400                  JSR PC,@XDSPT2(0)       ;DISPATCH SUPERVISOR COMMAND
003444  004767  000256                  JSR PC,UNSAVE
003450  016706  003432                  MOV OLDSTK,SP
003454  052737  000100  164000          BIS #100,@#COMCHI       ;ENABLE SUP. INTERR.
003462  000002                          RTI                     ;RETURN TO SYSTEM
                                ;
                                ;*****
003464                          HANG:   ;WAITLOOP FOR FURTHER SUP. CMDS
                                ;*****
                                ;
003464                                  .RCVXRD                 ;WAIT FOR A CMD
003470  004770  137400                  JSR PC,@XDSPT2(0)       ;DISPATCH NEXT CMD
003502  000207                  CONT:   RTS PC                  ;RETURN FROM HANG, REL. HOST
                                ;
                                ;*****
003504                          TABUPD: ;UPDATE OF MONITOR TABLES
                                ;*****
003504  012737  000007  177570          MOV #7,@#177570         ;TEST
003512  005237  140034                  INC @#XDIR              ;SET EXCHANGE BACK=FROS
003516  004767  001552                  JSR PC,EXCH             ;RESET ORIGINAL VECTORS
003522  012701  137170                  MOV #XMSKO,R1           ;TOP OF SPEC TABLE
003526  012702  137304                  MOV #XENDCM+2,R2                ;BOTTOM OF
003532  012100                  TABUP1: MOV (1)+,R0             ;SEND SPECS TO SUPER
003534                                  .SNDXRD
003546  020102                          CMP R1,R2               ;FINISHED?
003550  001370                          BNE TABUP1              ;NO
003552  012737  000070  177570          MOV #70,@#177570        ;TEST
003560                                  .RCVXRD                 ;WAIT FOR ACK
003572  022700  000006                  CMP #6,R0               ;WHEN THIS IS IN, THE NEW
003576  001401                          BEQ TABUP2              ;TABLE IS DEFINED AT SUPER
003600  000000                          HALT                    ;PROTOCOL-ERROR
003602  012701  137170          TABUP2: MOV #XMSKO,R1
003606  012702  137304                  MOV #XENDCM+2,R2
003612  012737  000700  177570          MOV #700,@#177570       ;TEST
003620                          TABUP3: .RCVXRD                 ;RECEIVE NEW TABLE
003632  010021                          MOV R0,(1)+             ;FROM SUPERVISOR
003634  020102                          CMP R1,R2               ;FINISHED?
003636  001370                          BNE TABUP3              ;NO
003640  012737  007000  177570          MOV #7000,@#177570      ;TEST
003646  022700  000003                  CMP #3,R0               ;WAS LAST EFX?
003652  001401                          BEQ TABUP4
003654  000000                          HALT                    ;PROTOCOL ERROR
003656  005037  140030          TABUP4: CLR @#XDIR              ;EXCHANGE VECTORS ACCORDING
003662  005737  157266                  TST @#XMITMOD           ;TIMP INTERR. MODE?
003666  001406                          BEQ TABUP5              ;NO
```

```
003670  013737  137276  000104          MOV @#XTINT1,@#104      ;YES-SET DUMMY VECTOR
003676  013737  137300  000106          MOV @#XTINT2,@#106
003704  004767  001364          TABUPS: JSR PC,EXCH             ;TO NEW TABLE
003710  013737  137274  172542          MOV @#XTREG2,@#IBUF     ;SET TIMER STATUS AND
003716  013737  137272  172540          MOV @#XTREG1,@#TSTAT
003724  000207                          RTS PC
                                ;
                                ;
                                ;
003726  000240                  UNSAVE: NOP
003730  012705  140012                  MOV #XREGd,R5           ;RESTORE GPR'S
003734  012500                          MOV (R5)+,R0
003736  012501                          MOV (R5)+,R1
003740  012502                          MOV (R5)+,R2
003742  012503                          MOV (R5)+,R3
003744  012504                          MOV (R5)+,R4
003746  016705  003126                  MOV REG5,R5
003752  000207                          RTS PC
                                ;
                                ;*****
003754                          ENDRUN: ;RESET SYSTEM; TERMINATE MONITORING SESSION
                                ;*****
                                ;
003754  012706  134000                  MOV #NEWSTK,SP
003760  012737  000001  140030          MOV #1,@#XDIR           ;SAVE TO ORIGINAL
003766  004767  001302                  JSR PC,EXCH             ;DO THE EXCHANGE
003772  005037  172540                  CLR @#TSTAT             ;CLR TIMER
003776  005037  164000                  CLR @#COMCHI            ;CLR COMMUNIC. CHANNEL
004002  004767  177720                  JSR PC,UNSAVE
004006  016706  003074                  MOV OLDSTK,SP
                                ;
004012  005000                          CLR R0                  ;INIT SYSTEM AT .EXIT
004014                                  .EXIT
                                        ;
```

```
                                        .TITLE EVENT HANDLERS
                                ;       **********************
                                ;
                                ;*****
004016                          DE:     ;DISPATCH ERROR, WRONG EV CODE
                                ;*****
004016  004767  001024                  JSR PC,ECOUT            ;SEND EVENT CODE
004022  012700  137030                  MOV #XQUES,R0           ;SEND QUESTION MARK
004026                                  .CHOUT
004032  000207                          RTS PC                  ;RETURN FROM DE
                                ;
                                ;*****
004034                          SG00:   ;EV1-EV5: TRAP 4,TRAP 10, BPT, IOT, PWPL
                                ;*****  ;SELECTION: PC,PS,SP
                                ;
004034  004767  001006                  JSR PC,ECOUT            ;SEND EVENT CODE
004040  016701  002210                  MOV MSK005,R1           ;GET SPEC MASK
004044  012702  000001                  MOV #1,R2               ;STARTING WITH EV 1
004050  020205                  SG001:  CMP R2,R5               ;LOOK FOR EV CODE
004052  001404                          BEQ SG002
004054  072127  177775                  ASH #-3,R1              ;SHIFT 3 RIGHT
004060  005202                          INC R2                  ;TO PLACE THE SPEC BITS FOR
004062  000772                          BR SG001                ;THE EVENT WHICH OCCURED
004064  032701  000001          SG002:  BIT #1,R1               ;PC?
004070  001403                          BEQ SG003               ;NO
004072  004467  000770                  JSR R4,PCOUT
004076  000000                          .WORD 0
004100  032701  000002          SG003:  BIT #2,R1               ;PS?
004104  001402                          BEQ SG004               ;NO
004106  004767  001000                  JSR PC,PSOUT
004112  032701  000004          SG004:  BIT #4,R1               ;SP?
004116  001404                          BEQ SG005               ;NO
004120  004767  001014                  JSR PC,SPOUT
004124  004767  001120                  JSR PC,TIMOUT
004130  020527  000003          SG005:  CMP R5,#3               ;WAS IT EV.1 OR 2?
004134  002002                          BGE SG006               ;NO
004136  105267  001756                  INCB TRAP               ;SET FLAG
004142  000207                  SG006:  RTS PC                  ;RETURN FROM SG00
                                ;
                                ;*****
004144                          SG01:   ;EV6-EV7 : EMT, TRAP - WITH SUB SELECTION
                                ;*****  ;SELECTION : PC-2, PS, SP, (PC-2), (SC)<-EMT ONLY
                                ;
004144  016701  002106                  MOV MSK007,R1           ;GET MASK
004150  012702  137224                  MOV #X5$060,R2          ;GET ADDR. OF 1ST SEL BYTE
004154  022705  000006                  CMP #6,R5               ;DETERMINE IF 6 OR 7
004160  001403                          BEQ SG011
004162  000301                          SWAB R1                 ;BRING EV7 MASK TO LOW BYTE
004164  062702  000010                  ADD #8.,R2              ;11 8 SEL BYTES PER EV 11
004170  032701  000001          SG011:  BIT #1,R1               ;SELECTION?
004174  001003                          BNE SG012               ;YES-GOTO SUBSELECTOR
004176  004767  000036                  JSR PC,SG01A            ;NO -GOTO REPORT PROCESSING
004202  000207                          RTS PC                  ;RETURN FROM SG01
                                ;
004204                          SG012:  ;SUBSELECTOR
                                ;
004204  017703  002676                  MOV @OLDSTK,R3          ;GET EMT/TRAP CODE
```

```
 58 004210  005743                      TST -(3)
 59 004212  011303                      MOV (3),R3
 60 004214  000240              SG013:  NOP
 61 004216  120312                      CMPB R3,(2)             ;EV INSTR.=SEL INSTR.?
 62 004220  001003                      BNE SG014               ;NO
 63 004222  004767  000012              JSR PC,SG01A            ;YES-GOTO REPORT PROCESSING
 64 004226  000207                      RTS PC                  ;RETURN FROM SG01
 65 004230  005202              SG014:  INC R2                  ;GET NEXT SEL BYTE
 66 004232  105712                      TSTB (2)                ;SELECTION?
 67 004234  001367                      BNE SG013               ;YES-GOTO SUBSELECTOR
 68 004236  000207                      RTS PC                  ;RETURN FROM SG01
 69                             ;
 70 004240                      SG01A:  ;*** REPORT PROCESSING FOR SG01 ***
 71                             ;
 72 004240  004767  000602              JSR PC,ECOUT            ;SEND EVENT CODE
 73 004244  032701  000002              BIT #2,R1               ;PC?
 74 004250  001403                      BEQ SG01A1              ;NO
 75 004252  004467  000610              JSR R4,PCOUT
 76 004256  000002                      .WORD 2                 ;FOR PC+2
 77 004260  032701  000004      SG01A1: BIT #4,R1               ;PS?
 78 004264  001402                      BEQ SG01A2              ;NO
 79 004266  004767  000620              JSR PC,PSOUT
 80 004272  032701  000010      SG01A2: BIT #10,R1              ;SP?
 81 004276  001402                      BEQ SG01A3              ;NO
 82 004300  004767  000634              JSR PC,SPOUT
 83 004304  032701  000020      SG01A3: BIT #20,R1              ;(PC-2)?
 84 004310  001403                      BEQ SG01A4              ;NO
 85 004312  004467  000644              JSR R4,CPCOUT
 86 004316  002002                      .WORD 2                 ;FOR PC-2
 87 004320  032701  000040      SG01A4: BIT #40,R1              ;(S2)?
 88 004324  001410                      BEQ SG01A5              ;NO
 89 004326  012700  137002              MOV #XMS2,R0            ;YES-SEND (S2)
 90 004332                              .CHOUT
 91 004336  012700  000052              MOV #*52,R0
 92 004342                              .GCOUT
 93 004346  004767  000676      SG01A5: JSR PC,TIMOUT
 94 004352  000207                      RTS PC                  ;RETURN FROM SG01A
 95                             ;
 96                             ;*****
 97 004354                      SG10:   ;EV20-EV27: TK, TP, PR, PP, K11-P,
 98                             ;*****  ;SELECTION: PC,PS, SP, (REG), (BUF).
 99                             ;
100 004354  010500                      MOV R5,R0               ;GET EC
101 004356  042700  177771              BIC #177771,R0          ;MASK INDEX 0,2,4,6
102 004362  016001  137204              MOV XMS101(0),R1        ;GET MASK WORD
103 004366  032705  000001              BIT #1,R5               ;EC ODD?
104 004372  001401                      BEQ SG100               ;NO
105 004374  000301                      SWAB R1                 ;YES
106 004376  010503             SG100:  MOV R5,R3               ;GET EC*2
107 004400  006303                      ASL R3
108 004402  004767  000440              JSR PC,ECOUT            ;SEND EVENT CODE
109 004406  032701  000002              BIT #2,R1               ;PC?
110 004412  001403                      BEQ SG101               ;NO
111 004414  004467  000446              JSR R4,PCOUT
112 004420  000000                      .WORD 0                 ;PC OF NEXT INSTR.
113 004422  032701  000004      SG101:  BIT #4,R1               ;PS?
114 004426  001402                      BEQ SG102               ;NO
```

```
     115 004430  004767  000456              JSR PC,PSOUT
     116 004434  032701  000010    SG102:    BIT #10,R1              ;SP?
     117 004440  001402                      BEQ SG103               ;NO
     118 004442  004767  000472              JSR PC,SPOUT
     119 004446  032701  000020    SG103:    BIT #20,R1              ;(REG)?
     120 004452  001406                      BEQ SG104               ;NO
     121 004454  017702  137712              MOV @#XDSTAT(3),R2      ;GET (REG)
     122 004460  004467  000544              JSR R4,REGOUT
     123 004464      046     200             .BYTE 46,200            ;IDENT: "&"
     124 004466  000000                      .WORD 0
     125 004470  032701  000040    SG104:    BIT #40,R1              ;(BUF)?
     126 004474  001412                      BEQ SG105               ;NO
     127 004476  016702  137712              MOV XDSTAT(3),R2        ;GET BUF. ADDR.
     128 004502  005722                      TST (2)+
     129 004504  011202                      MOV (2),R2
     130 004506  004467  000516              JSR R4,REGOUT
     131 004512      047     200             .BYTE 47,200            ;IDENT: "'"
     132 004514  000000                      .WORD 0
     133 004516  004767  000526              JSR PC,TIMOUT
     134 004522  000207            SG105:    RTS PC                  ;RETURN FROM SG10
     135                                     ;
     136                                     ;******
     137 004524                    SG11:     ;EV30: RK DISK WITH SUB SELECTOR
     138                           ;******   ;SEL: PC, PS, SP, REG0,...REG5,
     139                                     ;
     140 004524  016701  001540              MOV MSK110,R1           ;GET MASK
     141 004530  010146                      MOV R1,-(SP)
     142 004532  032701  000001              BIT #1,R1               ;SUB SELECTION?
     143 004536  001114                      BNE SG117               ;YES-GOTO SUB SELECTOR
     144 004540  012601            SG110:    MOV (SP)+,R1
     145 004542  004767  000300              JSR PC,ECOUT    ;SEND EVENT CODE
     146 004546  006201                      ASR R1
     147 004550  032701  000001              BIT #1,R1               ;PC?
     148 004554  001403                      BEQ SG111               ;NO
     149 004556  004467  000304              JSR R4,PCOUT
     150 004562  000000                      .WORD 0                 ;PC OF NEXT INSTRUCTION
     151 004564  006201            SG111:    ASR R1
     152 004566  032701  000001              BIT #1,R1               ;PS?
     153 004572  001402                      BEQ SG112               ;NO
     154 004574  004767  000312              JSR PC,PSOUT
     155 004600  006201            SG112:    ASR R1
     156 004602  032701  000001              BIT #1,R1               ;SP?
     157 004606  001402                      BEQ SG113               ;NO
     158 004610  004767  000324              JSR PC,SPOUT
     159 004614  012703  000060    SG113:    MOV #60,R3              ;CNT 0-5 IN ASCII
     160 004620  010546                      MOV R5,-(SP)
     161 004622  006305                      ASL R5
     162 004624  016505  137712              MOV XDSTAT(5),R5        ;ADDR. OF FIRST REG.
     163 004630  006201            SG114:    ASR R1
     164 004632  032701  000001              BIT #1,R1
     165 004636  001412                      BEQ SG116               ;NOT THIS REG.
     166 004640  117367  000015              MOVB R3,SG115+1         ;MOV REG.NR. IN OUT FIELD
     167 004644  022703  000063              CMP #63,R3              ;REG 3?
     168 004650  001416                      BEQ SG114A              ;YES-GO COMPUTE BLOCK NR.
     169 004652  011502                      MOV (5),R2
     170 004654  004467  000350    SG114C:   JSR R4,REGOUT
     171 004660      052     000   SG115:    .BYTE 52,0
```

```
172 004662      040     200                     .BYTE 40,200
173 004664  005725                      SG116:  TST (5)+                ;NEXT REGISTER
174 004666  005203                              INC R3
175 004670  022703  000006                      CMP #6,R3
176 004674  001365                              BNE SG114
177 004676  004767  020546                      JSR PC,TIMOUT
178 004702  012605                              MOV (SP)+,R5
179 004704  000207                              RTS PC                  ;RETURN FROM SG11
180                                     ;
181 004706                              SG114A: ;*** COMPUTE BLOCK NO. (LAST BLK+1 OF ACCESSED FILE)***
182                                     ;       ;   ((CYL-1)*2+SURF)*14+SECT+5
183                                     ;
184 004706  010146                              MOV R1,-(SP)
185 004710  010500                              MOV R5,R0               ;ADDR. OF REG.3
186 004712  022020                              CMP (0)+,(0)+           ;ADDR. OF REG.5 IN R0
187 004714  011000                              MOV (0),R0              ;GET DISK ADDR.
188 004716  010001                              MOV R0,R1
189 004720  072127  177773                      ASH #-5,R1              ;SHIFT CYL ADDR. RIGHT BOUNDED
190 004724  042701  177400                      BIC #177400,R1
191 004730  005301                              DEC R1                  ;CYL-1
192 004732  006301                              ASL R1                  ;*2
193 004734  032700  000020                      BIT #20,R0              ;SURFACE?
194 004740  001401                              BEQ SG114B              ;0
195 004742  005201                              INC R1                  ;ADD 1 TRACK
196 004744  070127  000014              SG114B: MUL #14,R1              ;*14
197 004750  042700  177760                      BIC #177760,R0          ;MASK SECTOR
198 004754  060001                              ADD R0,R1               ;ADD SECTOR
199 004756  062701  000030                      ADD #30,R1              ;30 FOR DIRECTORY OF 2 TRACKS
200 004762  010102                              MOV R1,R2               ;READY TO SEND STUFF OUT
201 004764  012601                              MOV (SP)+,R1
202 004766  000732                              BR SG114C               ;GO BACK FOR OUTPUT
203                                     ;       ;END OF BLOCK NUMBER CALCULATION
204                                     ;
205                                     ;
206 004770                              SG117:  ;***SUB SELECTOR FOR EV30***
207                                     ;
208 004770  010500                              MOV R5,R0
209 004772  006500                              ASL R0
210 004774  016003  157712                      MOV XDSTAT(0),R0        ;ADDR. OF FIRST REG.
211 005000  012701  157246                      MOV #XS1100,R1          ;ADDR.OF FIRST SEL. WORD
212 005004  012702  000604                      MOV #4,R2               ;CNT
213 005010  012103              SG118:  MOV (1)+,R3             ;GET REG NO.
214 005012  006303                              ASL R3
215 005014  060003                              ADD R0,R3               ;REG. ADDR.
216 005016  011304                              MOV (3),R4              ;REG. CONT.
217 005020  042124                              BIC (1)+,R4             ;DO AN "OR"
218 005022  021304                              CMP (3),R4
219 005024  001245                              BNE SG119               ;GOTO REPORT PROCESSING
220 005026  077213                              SOB R2,SG118            ;DO IF 4 TIMES
221 005030  012601                              MOV (SP)+,R1            ;NO EV REPORT PRODUCED-CORR. STACK
222 005032  000207                              RTS PC                  ;RETURN FROM SG11
223                                     ;
224                                     ;*****
225 005034                              SG12:   ;EV31-EV33: PIRQ,FPP,MMU
226                                     ;*****
227 005034  004767  000006                      JSR PC,ECOUT
228 005040  004767  000204                      JSR PC,TIMOUT
229 005044  000207                              RTS PC
230                                     ;
```

```
                        .TITLE BUFFER ROUTINES
                        ;**********************
                        ;
                        ;
                        ;*****
005046                  ECOUT:  ;OUTPUT OF EVENT CODE; LINKAGE: PC
                        ;*****
005046  012700  136440          MOV #XCRLF,R0
005052                          .CHOUT
005056  010500                  MOV R5,R0
005064                          .OCOUT                  ;LOG EVENT CODE
005064  000207                  RTS PC                  ;RETURN FROM ECOUT
                        ;
                        ;*****
005066                  PCOUT:  ;OUTPUT OF PC-K; K=1ST ARG.; LINKAGE: R4
                        ;*****
005066  012700  137032          MOV #XMPC,R0            ;SEND 1 FOR PC
005072                          .CHOUT
005076  017700  002004          MOV @OLDSTK,R0          ;GET EV-PC-K
005102  162400                  SUB (4)+,R0             ;USE 1ST ARG.
005104                          .OCOUT
005110  000204                  RTS R4                  ;RETURN FROM PCOUT
                        ;
                        ;*****
005112                  PSOUT:  ;OUTPUT OF EVENT PS; LINKAGE: PC
                        ;*****
005112  012700  137034          MOV #XMPS,R0            ;SEND * FOR PS
005116                          .CHOUT
005122  017700  001760          MOV @OLDSTK,R0          ;GET EV PS
005126  005720                  TST (0)+
005130  011000                  MOV (0),R0
005132                          .OCOUT
005136  000207                  RTS PC                  ;RETURN FROM PSOUT
                        ;
                        ;*****
005140                  SPOUT:  ;OUTPUT OF EVENT SP; LINKAGE: PC
                        ;*****
005140  012700  137036          MOV #XMSP,R0            ;SEND # FOR EVENT SP
005144                          .CHOUT
005150  017700  001732          MOV @OLDSTK,R0
005154                          .OCOUT
005160  000207                  RTS PC                  ;RETURN FROM SPOUT
                        ;
                        ;*****
005162                  CPCOUT: ;OUTPUT OF EVENT (PC-K); LINKAGE: R4
                        ;*****
005162  012700  137040          MOV #XMCPC,R0           ;SEND 5 FOR (PC-K)
005166                          .CHOUT
005172  017700  001710          MOV @OLDSTK,R0
005176  162400                  SUB (4)+,R0             ;1ST ARG.
005200  032700  000001          BIT #1,R0               ;PC ODD?
005204  001405                  BEQ CPC01               ;NO
005206  012700  137020          MOV #XAST,R0            ;YES-SEND ******
005212                          .CHOUT
005216  000403                  BR CPC02
005220  011004          CPC01:  MOV (0),R4
005222                          .OCOUT
005226  000204          CPC02:  RTS R4                  ;RETURN FROM CPCOUT
```

```
                        ;
                        ;*****
                        REGOUT: ;REGISTER OUTPUT; LINKAGE: R4
                        ;*****  ;WORD IN R2; IDENTIFIER IN (R4)
                        ;
005230  010400                  MOV R4,R0               ;ADDR.OF IDENTIFIER
005232                          .CHOUT
005236  022424                  CMP (4)+,(4)+           ;SKIP ARGUMENTS
005240  010200                  MOV R2,R0               ;GET VALUE OF WORD
005242                          .OCOUT
005244  000204                  RTS R4                  ;RETURN FROM REGOUT
                        ;
                        ;*****
005250                  TIMOUT: ;SEND TIME FROM KW11-P
                        ;*****
005250  005767  001070          TST TIMLG               ;TIME LOG?
005254  001406                  BEQ TMOUT1              ;NO
005256  013702  172544          MOV @#172544,R2         ;GET CNT
005262  004467  177742          JSR R4,REGOUT           ;SEND IT
005266  052     200     200     .BYTE 52,200,200,200    ;IDENTIF.I *
005271          200
005272  000207          TMOUT1: RTS PC                  ;RETURN FROM TIMOUT
                        ;
```

```
                        .TITLE REPLACEMENT PROCEDURE
                ;       ***************************
                ;
                ;*****
005274 005002           EXCH:   CLR R2                  ;R2=4*J (GROUP INDEX 0-M)
                ;*****
005276 012700 137170    REPL1:  MOV #XMSK0,R0           ;GET DEVICE MASK J
005302 062700                   ADD R2,R0
005304 011000                   MOV (0),R0
005306 005001                   CLR R1                  ;COUNT 0-15
005310 032700 000001    REPL2:  BIT #1,R0               ;NEXT DEVICE SET?
005314 001402                   BEQ REPL3               ;NO
005316 004767 000032            JSR PC,REPLAC           ;YES DO EXCHANGE
005322 022701 000017    REPL3:  CMP #15.,R1             ;END OF MASK?
005326 001403                   BEQ REPL4               ;YES
005330 005201                   INC R1                  ;NO
005332 006200                   ASR R0                  ;SHIFT MASK
005334 000765                   BR REPL2
005336 022702 000004    REPL4:  CMP #MAX4M,R2           ;ALL GROUPS DONE?
005342 001403                   BEQ REPL5               ;YES - DONE
005344 062702 000004            ADD #4,R2               ;INCR MASK OFFSET
005350 000752                   BR REPL1                ;NEXT GROUP
005352 000207           REPL5:  RTS PC                  ;RETURN FROM EXCH
                ;
                ;*****
005354                  REPLAC: ;SUBROUTINE TO REPLACE VECTOR I (=R1)
                ;*****   ;IN GROUP J (=R2/4), THE DIRECTION OF THE
                        ;REPLACEMENT IS SPECIFIED IN DIRECT
                ;
005354 010203                   MOV R2,R3               ;CALC. OFFSET IN VECTAB
005356 006303                   ASL R3                  ;(J*16+I)*2+#VECTAB
005360 006303                   ASL R3
005362 060103                   ADD R1,R3
005364 006303                   ASL R3                  ;*2 FOR WORD
005366 010304                   MOV R3,R4
005370 062704 137612            ADD #XVECTB,R4
005374 006303                   ASL R3                  ;OFFS. IN VECSAV=(J*4+I)*4
005376 010305                   MOV R3,R5
005400 062705 137412            ADD #XVECSV,R5
005404 011403                   MOV (4),R3              ;ADDR. OF VECTOR
005406 005737 140030            TST @#XDIR              ;WHICH WAY?
005412 001015                   BNE SAVBAK
005414 012325                   MOV (3)+,(5)+           ;SAVE ORIG. VECT. IN VECSAV
005416 011315                   MOV (3),(5)
005420 005743                   TST -(3)
005422 012704 137170            MOV #XMSK0,R4           ;GET PC(J)
005426 060204                   ADD R2,R4               ;ADDR. OF REPLACEMENT PC
005430 005724                   TST (4)+
005432 011423                   MOV (4),(3)+            ;REPLACE PC(J)
005434 012704 000340            MOV #BR7,R4             ;NEW PS=PRIO+I
005440 060104                   ADD R1,R4
005442 010413                   MOV R4,(3)              ;REPL.PS(J)
005444 000207                   RTS PC
005446 012523           SAVBAK: MOV (5)+,(3)+           ;REST. SAVED VECT. TO ORIG.
005450 011513                   MOV (5),(3)
005452 000207                   RTS PC                  ;RETURN FROM REPLAC
                                }
```

```
                                ;DUMPS N LINES (N=R2) OF 8 WORDS
                                ;FROM STARTING ADD IN R1
005454 010346                   MOV R3,-(SP)
005456 012703  000010           MOV #10,R3              ;8 WORDS
005462 012100                   MOV (R1)+,R0
005464 004767  000030           JSR PC,OCPUT
005470 077304                   SOB R3,DUM2
005472 012700  136440           MOV #XCRLF,R0
005476 004767  000112           JSR PC,CHPUT            ;PRINT CRLF
005532 004767  000162           JSR PC,SPNOBF
005536 077215                   SOB R2,DUM1
005540 012603                   MOV (SP)+,R3
005542 000207                   RTS PC
005544 000000                   CRLF:   .WORD 0
005546    040     200           BLANK:  .BYTE 40,200
                                ;
005550 010046                   OCPUT:  MOV R0,-(SP)
005552 010146                   MOV R1,-(SP)
005554 010246                   MOV R2,-(SP)
005556 012702  000006           MOV #6,R2               ;6 DIGITS
005562 010001                   OCPUT1: MOV R0,R1
005564 042701  177770           BIC #177770,R1          ;MASK RIGHTMOST DIGIT
005570 062701  000060           ADD #60,R1              ;CONVERT TO ASCII
005574 110146                   MOVB R1,-(SP)
005576 006000                   ROR R0
005600 006200                   ASR R0
005602 006200                   ASR R0
005604 077202                   SOB R2,OCPUT1
005606 012702  000006           MOV #6,R2
005612 112677  000454           OCPUT2: MOVB (SP)+,@TXTPTR      ;PUT NUMBER IN BUFFER
005616 005267  000450           INC TXTPTR
005622 077205                   SOB R2,OCPUT2
005624 012700  136442           MOV #XBLANK,R0          ;PUT A BLANC
005630                          .CHOUT
005634 012602                   MOV (SP)+,R2
005636 012601                   MOV (SP)+,R1
005640 012600                   MOV (SP)+,R0
005642 000207                   RTS PC                  ;RETURN FROM OCPUT
                                ;
                                ;
005644 121027  000200           CHPUT:  CMPB (R0),#200          ;ETX?
005650 001422                   BEQ CHPUT2              ;YES
005652 121027  000000           CMPB (R0),#0            ;ETX WITH CRLF?
005656 001405                   BEQ CHPUT1
005660 112077  000406           MOVB (R0)+,@TXTPTR      ;PUT CHAR
005664 005267  000402           INC TXTPTR
005670 000765                   BR CHPUT
005672 112777  000015  000372   CHPUT1: MOVB #15,@TXTPTR        ;PUT CRLF
005700 005267  000366           INC TXTPTR
005704 112777  000012  000360   MOVB #12,@TXTPTR
005712 005267  000354           INC TXTPTR
005716 000207                   CHPUT2: RTS PC                  ;RETURN FROM CHOUT
                                ;
                                .TITLE TRANSMISSION ROUTINE
                                ;*************************
                                ;
                                ;
```

```
115 005670 010046                SENDBF: MOV R0,-(SP)
116 005672 010146                        MOV R1,-(SP)
117 005674 012737 052525 177570          MOV #52525,@#177570     ;TEST
118 005702 042737 000100 164000          BIC #100,@#COMCHI
119 005710 012701 137046                 MOV #XTXTBF,R1          ;GET START ADDR. OF BUFFER
120 005714 020167 000322                 CMP R1,TXTPTR           ;ANY TEXT TO SEND?
121 005720 001002                        BNE SENDB0              ;YES
122 005722 000167 000124                 JMP SENDB3              ;NO-EXIT
123 005726 032767 000001 000306  SENDB0: BIT #1,TXTPTR           ;NEXT FREE=ODD?
124 005734 001405                        BEQ SENDB1              ;NO
125 005736 112777 000001 000276          MOVB #1,@TXTPTR         ;YES, PUT FILLER IN HIGH BYTE
126 005744 062767 000002 000272          INC TXTPTR              ;AND SET EVEN BOUNDARY
127 005752 012100                SENDB1: MOV (R1)+,R0            ;SEND A WORD
128 005754                               .SNDWRD
129 005764 020167 000252                 CMP R1,TXTPTR           ;END OF LINE?
130 005770 001367                        BNE SENDA1              ;NO
131 005772 032737 000001 177570          BIT #1,@#177570         ;CHECK SWR
132 006000 001402                        BEQ SENDB2              ;NOT SET
133 006002 052704 000002                 BIS #2,R4               ;MARK END
134 006006 012700 000004        SENDB2: MOV #EOT,R0             ;YES
135 006012                               .SNDWRD
136 006024                               .RCVWRD                 ;WAIT FOR ACK
137 006036 012737 177777 177570          MOV #177777,@#177570    ;TEST
138 006044 012767 137046 000170          MOV #XTXTBF,TXTPTR      ;RESET TEXT POINTER
139 006052 012601                SENDB3: MOV (SP)+,R1
140 006054 012600                        MOV (SP)+,R0
141 006056 052737 000100 164000          BIS #100,@#COMCHI       ;ENABLE SUPER INTERR.
142 006064 000240                        NOP
143 006066 000240                        NOP
144 006070 000240                        NOP
145 006072 000207                        RTS PC                  ;RETURN FROM SENDBF
146
```

```
                                        .TITLE RESIDENT DATA SECTION
                                ;       ***************************
                                ;
000074  052  052  052  AST:    .ASCII/****** /
000077  052  052  052
000102  040
000103  200                    .BYTE 200
000104  077  200       QUES:   .BYTE 77,200    ;?
000110  041  200       MPC:    .BYTE 41,200            ;!
000112  042  200       MPS:    .BYTE 42,200            ;"
000114  043  200       MSP:    .BYTE 43,200            ;#
000116  044  200       MCPC:   .BYTE 44,200            ;$
000120  045  200       MS2:    .BYTE 45,200            ;%
000122  000             TRAP:   .BYTE 0         ;-1: TRAP A, 1: TRAP 10.
                                .EVEN
                                ;
000122                  TXTBUF: .BLKW 40.               ;LINE OF 60 CHARS
000242  000000          TXTPTR: .WORD 0                 ;PNTR TO NEXT FREE CHAR POS
                                ;
                                ;
000244  000016          MASK0:  .WORD ^B0000000000001110
000246  154002          PC0:    XENTR0
000250  000000          MASK1:  .WORD ^B0000000000000200
000252  154024          PC1:    XENTR1
                                ;
                                ;               SUBMASK         INDEX
                                ;
000254  077777          MSK085: .WORD ^B0111111111111111  ;0    ;MASK PC,PS,SP FOR EV 1-5
000256  011062          MSK067: .WORD ^B0010010010011010  ;1    ;MASK SEL,PC-2,PS,SP,(PL-2),(S2) EV6,7
000260  017076          MSK101: .WORD ^B0011111010111110  ;2    ;MASK SEL PC,PS,SP,(REG),(BUF),EV 20,21
000262  011076          MSK123: .WORD ^B0010010010111110  ;3    ;MASK SEL,PC,PS,SP,(REG),(BUF), EV <2,23
000264  031016          MSK105: .WORD ^B0011001000001110  ;4    ;MASK  -"- FOR EV 24,25
000266  031062          MSK167: .WORD ^B0011001010011010  ;5    ;MASK  -"- FOR EV 26,27
000270  001602          MSK110: .WORD ^B0000000111010000  ;6    ;MASK SEL,PC,PS,SP,+REG=R6G5, EV 30=R6
000272  000000          MSK111: .WORD 0                   ;7    ;EV 31=PIRQ
000274  000000          MSK112: .WORD 0                   ;10   ;EV 32=PP
000276  000000          MSK113: .WORD 0                   ;11   ;EV 33=MAU
                                ;
                                ;       SELECT WORDS FOR CONDITIONAL EVENT REPORT
                                ;
                                ;               INDEX
                                ;
003500  350  351         SL0060: .BYTE 350,351   ;0      ;SEL EMT CODE FOR EV 6
003502  340  341                 .BYTE 340,341   ;1
003504  376  000                 .BYTE 376,0     ;2
003506  000  000                 .BYTE 0,0       ;3
003510  000  000         SL0070: .BYTE 0,0       ;4    ;SEL TRAP CODE FOR EV 7
003512  000  000                 .BYTE 0,0       ;5
003514  000  000                 .BYTE 0,0       ;6
003516  000  000                 .BYTE 0,0       ;7
003520  000  000                 .BYTE 0,0       ;10
003522  000  000         SL1100: .BYTE 0,0       ;11   ;SEL REG NB FOR EV 30 +RR
003524  000  000                 .BYTE 0,0       ;12
003526  000  000         SL1101: .BYTE 0,0       ;13
003530  000  000                 .BYTE 0,0       ;14
003532  000  000         SL1102: .BYTE 0,0       ;15
003534  000  000                 .BYTE 0,0       ;16
```

```
 56 006136  000     000        SLI103: .BYTE 0,0  ;17
 57 006140  000     000                .BYTE 0,0  ;20
 58 006142  000000             TIMEON: .WORD 0
 59 006144  000000             TIMLG:  .WORD 0
 60 006146  000000             TREG1:  .WORD 0
 61 006150  000000             TREG2:  .WORD 0
 62 006152  140036             TINT1:  XRTI
 63 006154  000340             TINT2:  BR7
 64 006156  000003             ENDCOM: .WORD 3
 65                            ;
```

;TIME INTERR. TO DUMMY RTI
;TIME INTERR. AT PRIO. 7
;END OF COMM. AREA

```
                        ;
                        ;
000360  134742         DISPT0: XDE             ;EV 0, DISPATCH ERROR
000362  134760                 XSG0B           ;EV 1, TRAP 4
000364  134760                 XSG0B           ;EV 2, TRAP 10
000366  134760                 XSG0B           ;EV 3, BPT
000370  134760                 XSG0B           ;EV 4, IOT
000372  134760                 XSG0B           ;EV 5, PWFL
000374  135070                 XSG01           ;EV 6, EMT
000376  135070                 XSG01           ;EV 7, TRAP
000400  134742                 XDE             ;EV 10,
000402  134742                 XDE             ;
000404  134742                 XDE             ;
000406  134742                 XDE             ;
000410  134742                 XDE             ;
000412  134742                 XDE             ;
000414  134742                 XDE             ;
000416  134742                 XDE             ;
000420  135300         DISPT1: XSG1B           ;EV 20, TK
000422  135300                 XSG1B           ;EV 21, TP
000424  135300                 XSG1B           ;EV 22, PR
000426  135300                 XSG1B           ;EV 23, PP
000430  135300                 XSG1B           ;EV 24
000432  135300                 XSG1B           ;EV 25
000434  135300                 XSG1B           ;EV 26
000436  135300                 XSG1B           ;EV 27
000440  135450                 XSG11           ;EV 30, RK
000442  135760                 XSG12           ;EV 31, PIRQ
000444  134742                 XDE             ;
000446  134742                 XDE             ;
000450  134742                 XDE             ;
000452  134742                 XDE             ;
                        ;
```

```
                        ;
                        DISPTZ: XDE                     ;00  DISP. ERROR
                                XTABUP                  ;02  TABLE UPDATE
                                XHANG                   ;04  SUSPEND HOST
                                XENDRU                  ;06  TERMINATE SESSION
                                XCONT                   ;10  CONTINUE AFTER HANG
                        ;
                        VECSAV: .BLKW N*2               ;CONT, ORIG, PC AND PS
                        ;
                        ;       VECT, ADDR,             -J-  -I-  -VECT-
                        ;
                        ;       *** GROUP 0 ***
                        ;
                        VECTAB: 0               ;0      0    0    NOT IMPLEMENTED
                                4               ;0      0    1    TRAP 4
                                10              ;0      0    2    TRAP 10
                                14              ;0      0    3    BPT
                                20              ;0      0    4    IOT
                                24              ;0      0    5    PDNFL
                                30              ;0      0    6    EMT
                                34              ;0      0    7    TRAP
                                .=.+16.         ;RESERVED FOR 8 VECT. IN GRP 0
                        ;
                        ;       *** GROUP 1 ***
                        ;
                                300             ;1      0    TK
                                304             ;1      1    LP
                                70              ;1      2    PR
                                74              ;1      3    PP
                                100             ;1      4    KW11-P
                                0               ;1      5
                                0               ;1      6
                                0               ;1      7
                                220             ;1      10   RK11
                                240             ;1      11   PTR0
                                244             ;1      12   FPP
                                250             ;1      13   .MMU
                                0               ;1      14
                                0               ;1      15
                                0               ;1      16
                                0               ;1      17
                        ;
                        ;DEVICE STATUS REGISTER TABLE
                        ;
                        DVSTAT: .=.+32.         ;NO STATUS FOR TRAPS IN GRP 0
                                177560          ;1      0    TKS
                                177564          ;1      1    TPS
                                177550          ;1      2    PRS
                                177554          ;1      3    PPS
                                172540          ;1      4    KW11-P
                                0               ;1      5
                                0               ;1      6
                                0               ;1      7
                                177440          ;1      10   RKCS
                                177772          ;1      11   PIR0
                                0               ;1      12   FLT ERR
                                177572          ;1      13   MMU SR0
```

```
RESIDENT DATA SECTION   RT-11 MACRO VM02-09    08111626 PAGE 13+

              007066                        .=.+8.        ;RESERVED
   007266                        REG0:     .BLKW 5       ;REGISTER SAVE AREA
   007100    000000              REG5:     .WORD 0
   007102    000000              PSWSAV:   .WORD 0       ;TO SAVE PSW ON ENTRY
   007104    000000              DIRECT:   .WORD 0       ;FLAG FOR REPLAG PROCEDURE
   007106    000000              OLDSTK:   .WORD 0       ;SP AT EVENT TIME
   007110    000000              RETADD:   .WORD 0       ;PC FOR JSR SIMUL.
   007112    000002              RTI1:     RTI           ;RETURN FROM CLOCK OR TRAP INTERR.
   007114    000000              ENDMON:   .WORD 0       ;END OF RELOC. PROGR.
                                           .TITLE S-MON-
              001014'                      .END START
```

```
LOG:: INTERACTIVE SYSTEM MONITO RT-11 MACRO VM02-09    09:15:55  PAGE 1

                                        .TITLE LOG:: INTERACTIVE SYSTEM MONITOR
                                ;*********************************************************
                                ;*
                                ;* CONTROL AND LOGGING PROGRAM FOR MONITOR SYSTEM "MTMOS"
                                ;* LOG IS THE MAIN PROGRAM AND HAS TO BE LINKED TO
                                ;* THE OBJECT MODULES LOGIO AND LOGUPF.
                                ;* ALL ENTRIES AND ROUTINES FOR THE HARDWARE MONITOR ARE
                                ;* FORESEEN, THE CORRESPONDING CODE IS REPLACED BY NOP'S.
                                ;*
                                ;*********************************************************
                                ;
                                        .GLOBL TTYOUT,OPRINT,OCRD,PRTBYT,GINRD
                                        .GLOBL BINPRT,BITPOS,BIAS,DECODE,IOERR,IOERS
                                        .GLOBL IOERI,PUTCRD,CARIN,CARERR,IOER2
                                        .GLOBL CNTOUT,PRINT,PRIARG,TTYLOG,LCLK
                                        .GLOBL LOGCNT,CRIND,CNTIM,COUNT,CKLP
                                        .GLOBL HANG,TRIEIO,SBUFN,SBUFC,DISPRO
                                        .GLOBL ESERR,SBUFSW,SBUF1,SWFCP,PRAAM1
                                        .MCALL .PRINT,.REQDEF,.EXIT,.QUTREF,.DFF,.SEND,.RECV,.DEF
                                        .MCALL .TTYIN,.TTYOUT,.WRITE,.V2;.,.TTINR,.WRITW,.READ,.READW
                                        .MCALL .CSIGEN,.CSISPC,.CLOSE,.WAIT,.FETCH,.PUTER,.RELEAS
R00004                                  ..V2..
R00000                                  .REGDEF
        000000                          .ASECT
        001000                          .=1000
        177564                  TPS=177564
        000004                  FUF=04
        000006                  ACK=06
        000300                  BR6=300
        000340                  BR7=340
        177776                  PSW=177776
        000200                  MNGMI=4                 ;MMOR INP. STATUS
        172542                  TBUF=172542             ;TIMER CNT BUFFER
        172540                  TST1=172540             ;TIMER STATUS REGISTER
        000520                  PLIV=520                ;VECTOR FOR PARALLEL LINK
        164000                  PLS=164000              ;STATUS REG.  -"-
        164002                  PLD=164002              ;OUTPUT BUF.  -"-
        164004                  PLI=164004              ;INPUT BUF.   -"-
        164000                  IIS=PLS                 ;ABBREVIATION FOR PAR. LINK
        164000                  IOS=PLS
        164004                  IIB=PLI
        164002                  IOB=PLO
        000520                  IIV=PLIV
        000524                  IOV=PLIV+4
        164000                  COMLNI=IIS              ;STATUS REG. OF COMM. CHANNEL IN
        164000                  COMLNO=IOS              ;       "       "     OUT
        177564                  TTYOUT=177564           ;TTY OUTPUT STATUS
        000032                  MAXCMD=32
        000010                  MAXEV=10
        000020                  LSPFIL=0.+MAXEV         ;LENGTH OF LOG SPECIFIC
        000114                  HSPFIL=ENDCOM-MASK0+2
                                ;
```

```
                        .TITLE  INIT AND WAITLP
                ;       .IDENT /V02.09/
                ;
                        .NLIST BEX
                        ;
                        .MACRO .SWAIT
                        TST @#PLSW              ;WAS PREVIOUS WORD RECEIVED
                        BMI .-4                 ;WAIT IF NOT
                        MOV R0,@#PLOB           ;SEND WORD IN R0
                        .ENDM
                ;
                        .MACRO .RCVWRD
                        TSTB @#PLSR             ;IS THERE A NEW WORD
                        BPL .-4                 ;WAIT IF NOT
                        MOV @#PLIB,R0           ;STORE WORD IN R0
                        .ENDM
                ;
001000  012706  001002  START:  MOV #.,SP
001004  005065                  CLR R5
001010  012767  011134  010112  MOV #SBUF1,SBPTR        ;INIT. BUFFER POINTER
001014  012700  007340          MOV #STAR,R0
001020  004767  000000G         JSR PC,PRINT
001024  012737  001200  164000  BIC #1200,@#176
001032  012737  006520  000520  MOV #RCVINT,@#IIV       ;SPT LOW. VECTOR
001040  012737  006520  000522  MOV #LRB,@#IIV+2
001046  004767  000064          JSR PC,INIT             ;INITIALIZE PARAMS AND TABLES
                ;
                ;*******************************************
                ;
001052                  WAITLP: ;SYSTEM WAIT LOOP AND TASK DISPATCHER
                ;
                ;*******************************************
                ;
001052  005037  177776          CLR @#PSW               ;PRIORITY=0-BITS CLEAR
                        .TTINR          ;IS THERE SOME TTY INPUT?
001056
001060  103402                  BCS .+6                 ;YES
001062  004767  001062          JSR PC,SYSCMD           ;YES-ANALYZE AND DISPATCH CMD
                ;
                ;*** SCAN DISPATCH WORD AND DISP. SCHEDULED TASK ***
                ;
001066  005767  012174  WAIT1:  TST DISPWD              ;ANYTHING THERE?
001072  001767                  BEQ WAITLP              ;NO-KEEP WAITING
001074  005004                  CLR R4                  ;YES
001076  012705  000401          MOV #401,R5             ;SET LOWER BYTE OF DISPWD
001102  130567  012160  WAIT2:  BITB R5,DISPWD          ;BIT?
001106  001406                  BEQ .+16                ;NO
001110  010046                  MOV R4,-(SP)            ;SAVE REGS
001112  010546                  MOV R5,-(SP)
001114  004774  012722          JSR PC,@DISPTB(4)       ;YES-DISPATCH TASK
001120  012605                  MOV (SP)+,R5
001122  012604                  MOV (SP)+,R4
001124  000241          WAIT3:  CLC
001126  006505                  ASL R5                  ;SHIFT PLEASE
001130  103750                  BCS WAITLP              ;END OF DISPWD
001132  064704                  ADD (4)+                ;BUMP INDEX TO DISPTB
001134  000762                  BR .-12                 ;CHECK NEXT BIT
                ;
```

```
                                        .TITLE  UTILITY TASKS
                                        ;**********************
                                ;
                                ;
                                ;*****
001136                          INIT:   ;INITIALIZE SYSTEM PARAMS AND TABLES
                                ;*****
001136  005037  177546                  CLR @#177546            ;INIT LINE CLOCK
001142  013767  000100  012204          MOV #100,LCXSV1
001150  012737  000000G 000100         MOV #LCLK,#100
001156  012700  010604                  MOV #TFIELD,R0
001162  005720                         TST (0)+
001164  012720  030060                  MOV #30060,(0)+         ;CLEAR TIME COUNTER
001170  012720  030060                  MOV #30060,(0)+
001174  112720  000060                  MOVB #60,(0)+
001200  105720                         TSTB (0)+               ;SKIP PERIOD
001202  012710  030060                  MOV #30060,(0)
001206  012700  013162                  MOV #COUNT,R0           ;CLEAR EVENT COUNTERS
001212  012701  000041                  MOV #33.,R1
001216  005220                 INIT1:   CLR (0)+
001220  005301                         DEC R1
001222  001375                         BNE INIT1
001224  005737  164004                  TST @#TIR               ;CLEAR INPUT BUFFER
001230  052737  000100  177546         BIS #100,@#177546       ;ENABLE INTERRUPTS
001236  052737  000100  164000         BIS #100,@#CSR+CHI
001244  000207                         RTS PC                  ;RETURN FROM INIT
                                ;
                                ;
                                ;*****
001246                          PRINTB: ;PRINT R1 CHARS, STARTING AT (R5)
                                ;*****
001246  010046                          MOV R0,-(SP)
001250  010146                          MOV R1,-(SP)
001252  010346                          MOV R3,-(SP)
001254  010446                          MOV R4,-(SP)
001256  012501                          MOV (5)+,R1
001260  022703  000120                  CMP #80.,R3             ;PLAUSTBILITY CHECK
001264  103405                          BLO PRTB1               ;OK
001266  012700  010640                  MOV #QUES,R0            ;"?"
001272  004767  000000G                 JSR PC,PRINT
001276  000432                          BR PRTB3
001300                          PRTB1:  ,TTYOUT (5)+
001300  111104                          MOVB (5),R4             ;ANALYSE CHAR
001310  042704  177400                  BIC #177400,R4          ;CLEAR HIGH BYTE
001314  162704  000040                  SUB #40,R4
001320  103410                          BLO PRTB2               ;CR OR LF
001322  001413                          BEQ PRTB2               ;BLANC
001324  022427  000014                  CMP R4,#14
001330  100410                          BLO PRTB2               ;>55
001332  006304                          ASL R4                  ;ID CODE*2
001334  016400  010756                  MOV IDADD(0),R0         ;ADR OF IDENTIFIER
001340  004767  000000G                 JSR PC,PRINT
001344  105721                          TSTB (1)+               ;SKIP PREVIOUS ID
001346  005303                          DEC R3
001350  001405                          BEQ PRTB3
001352  005303                 PRTB2:   DEC R3
001354  001351                          BNE PRTB1
```

UTILITY TASKS   RT-11 MACRO V002-09   00:15:55 PAGE 3+

```
001356  105737  177564          TSTB @#TPS
001362  100375                  BPL .-4
001364  012604          PRTB3:  MOV (SP)+,R4
001366  012603                  MOV (SP)+,R3
001370  012601                  MOV (SP)+,R1
001372  012600                  MOV (SP)+,R0
001374  000205                  RTS R5                  ;RETURN FROM PRINTB
```

```
                                        .TITLE  CMDI FOR HOST MODE SPECS
                                        ;*******************************
                                ;
                                ;
                                ;*****
001376                          CMDIPT: ;COMMAND INTERPRETER FOR MON,TABLE UPDATE
                                ;*****
001376  010046                          MOV R0,-(SP)
001400  010146                          MOV R1,-(SP)
001402  010246                          MOV R2,-(SP)
001404  010346                          MOV R3,-(SP)
001406  010546                          MOV R5,-(SP)
001410  005005                          CLR R5                  ;SET LOOP SWITCH
                                ;
001412                          CMDI0:  ;***MASK UPDATE ***
                                ;
001412  012700  006674                  MOV #MSKIN,R0
001416  004767  000000G                 JSR PC,PRINT
                                ;
001422                          CMDI1:  ;*** Y/N INTERPRETER ***
                                ;
001422                                  .TTYIN                  ;READ Y OR NO WITH LEADING GARBAGE
001426  042700  000200                  BIC #200,R0
001432  122700  000015                  CMPB #15,R0
001436  001402                          BEQ CMDI2
001440  010201                          MOV R2,R1               ;SAVE CHAR
001442  000767                          BR CMDI1
001444                          CMDI2:  .TTYIN                  ;EMPTY RING BUFFER
001450  122701  000116          CMDI3:  CMPB #116,R1            ;N?
001454  001002                          BNE CMDI4               ;NO
001456  000175  012642                  JMP @CMDT1(5)           ;DISPATCH
001462  122701  000131          CMDI4:  CMPB #131,R1            ;Y?
001466  001406                          BEQ CMDI5               ;YES
001470  012700  027764                  MOV #CMDERR,R0
001474  004767  000000G                 JSR PC,PRINT            ;PRINT ERROR
001500  000175  010656                  JMP @CMDI2(5)           ;DISPATCH
                                ;
001504                          CMDI5:  ;*** INDEX INPUT ***
                                ;
001504  012703  024724                  MOV #INREQ,R3           ;REQUEST INDEX
001510  004767  000000G                 JSR PC,PRINT
001514  012700  007256          CMDI6:  MOV #CRLFS+2,R0
001520  004767  000000G                 JSR PC,PRINT
001524  004767  000000G                 JSR PC,OCT0             ;READ INDEX INTO R1
001530  005703                          TST R3                  ;1?
001532  002402                          BLT CMDI7               ;YES
001534  000175  012664                  JMP @CMD(4(5)           ;NO-DISPATCH FOR VALUE OUTPUT
001540  000175  012656          CMDI7:  JMP @CMDT3(5)           ;YES-DISPATCH FOR NEXT INDEX
                                ;
001544                          CMDI8:  ;*** MASK I/O ***
                                ;
001544  006301                          ASL R1                  ;INDEX*4
001546  006301                          ASL R1
001550  062701  014006                  ADD #MASK0,R1           ;ADDR. OF MASK
001554  004767  000000G                 JSR PC,OCTPRT           ;PRINT MASK
001560  004767  000000G                 JSR PC,OCTRD            ;READ NEW MASK
001564  000753                          BR CMDI6                ;READ NEXT INDEX
```

```
                        ;
                        CMDI9:  ;*** SUBMASK I/O ***
                        ;
                                ASL R1                  ;INDEX*2
                                ADD #MSK025,R1          ;ADDR OF SUBMASK
                                JSR PC,BINPRT           ;PRINT MASK
                                JSR PC,BINRD            ;READ NEW MASK
                                BR CMDI6                ;READ NEXT INDEX
                        ;
                        CMDI10: ;*** SUBMASK UPDATE ***
                        ;
                                TST (S)+                ;INCR. LOOP SWITCH
                        CMDI11: MOV #SMSKIN,R0
                                JSR PC,PRINT
                                JMP CMDI1               ;READ Y/N
                        ;
                        CMDI12: ;*** SELECT WORD UPDATE ***
                        ;
                                TST (S)+                ;INCR. LOOP SWITCH
                        CMDI13: MOV #SELWIN,R0
                                JSR PC,PRINT
                                JMP CMDI1               ;READ Y/N
                        ;
                        CMDI14: ;*** SELECT WORD I/O ***
                        ;
                                ASL R1                  ;INDEX*2
                                MOV R1,R2               ;SAVE INDEX
                                CMP #62.,R1             ;INDEX>124
                                BLT CMDI16              ;YES=>DON'T SORD I/O
                                MOV #2,R3               ;PROCESS 2 BYTES
                        ;
                        CMDI15: ;*** BYTE I/O ***
                        ;
                                CLR R0
                                MOVB SL000F(2),R0       ;GET OLD BYTE VALUE
                                JSR PC,BINPRT           ;PRINT IT
                                MOV #CR,R0
                                JSR PC,PRINT            ;PRINT :
                                JSR PC,OCRD             ;GET NEW VALUE
                                TST R1
                                BMI CMDI5A              ;NO NEW VALUE
                                MOVB R1,SL0060-(2)      ;STORE IT
                        CMDI5A: INC R2
                                DEC R3                  ;SECOND BYTE?
                                BNE CMDI15
                                BR CMDI6
                        ;
                        CMDI16: ;*** WORD I/O ***
                        ;
                                MOV SL3660(2),R0        ;GET OLD WORD VALUE
                                JSR PC,PRTWRD           ;PRINT IT
                                MOV #CR,R0
                                JSR PC,PRINT
                                JSR PC,OCRD             ;GET NEW VALUE
                                TST R1
                                BMI CMDI6               ;NO NEW VALUE
                                MOV R1,SL0100(2)        ;STORE IT
```

```
                                        .TITLE  LOG OUTPUT ROUTINES
                                        ;**************************
                                ;
                                ;
                                ;*****
006214                          RECORD: ;OUTPUT OF A BUFFER ONTO MASS STORAGE
                                ;*****
                                ;
006214  105267  005044                  INCB CHRSY              ;INDICATE I/O BUSY
006220  042767  000002  005040          BIC #2,DISPWD           ;CLEAR DISPATCH BIT
006226  105767  005062                  TSTB SLGREC             ;RECORDING MODE?
006232  001445                          BEQ REC4                ;NO-RETURN
006234  105767  002665                  TSTB SWUFSW             ;WHICH BUFFER IS READY?
006240  001405                          BEQ REC1                ;BUFFER 1
006242  012703  012134                  MOV #SWUF2,R3           ;BUFFER 2
006246  105067  002653                  CLRB SWUFSW
006252  000402                          BR REC2
006254  012703  011134          REC1:   MOV #SWUF1,R3
006260                          REC2:   .WRITE #AREA,#4,R3,#256.,BLKCNT
006324  103004                          BCC REC3                ;NO ERROR
006326  004567  000000G                 JSR R5,IOERR            ;I/O ERROR
006332  004                     .BYTE 4,4                       ;ON CHANNEL 4
006334  000402                          BR REC4
006336  005267  000726          REC3:   INC BLKCNT              ;NEXT BLOCK
006342  105067  000716          REC4:   CLRB CHRSY
006346  105767  005001                  TSTB OVRUN              ;WAS THERE DATA OVERRUN?
006352  001406                          BEQ REC5                ;NO
006354  012746  000340                  MOV #340,-(SP)          ;YES-SET UP NEW STACK
006360  012746  006370                  MOV #REC5,-(SP)         ;FOR RTI IN RCVMOR
006364  020167  177240                  JMP RCVMOR              ;AND JUMP THERE
006370  000207          REC5:   RTS PC                          ;RETURN AND RECORD
                                ;
                                ;*****
006372                          DUMP:   ;DUMP LOG RECORD AS IS ON TERMINAL
                                ;*****
                                ;
006372  105267  004666                  INCB CHRSY              ;INDICATE I/O BUSY
006376  042767  000004  004662          BIC #4,DISPWD           ;CLR DISPATCH BIT
006404  016701  002522                  MOV SWEC2,R1            ;GET LOC OF CURR. REC.
006410  022167  002514                  CMP R1,SWEC1            ;IS REC WRAPPED OVER BUFFER END?
006414  101012                          BPL DUMP1               ;YES-TREAT IT AS 2 PARTS
006416  166701  002506                  SUB SWEC1,R1            ;NO-JUST DUMP IT
006422  010103                          MOV R1,R3
006424  016767  002500  000004          MOV SWEC1,DUMP0
006432  004567  172610                  JSR R5,PRINTH
006436  000000          DUMP0:  .WORD 0
006440  000424                          BR DUMP3
006442  012702  013134          DUMP1:  MOV #SWBUF,R2           ;WRITE IN 2 PARTS
006446  166702  002456                  SUB SWEC1,R2
006452  010203                          MOV R2,R3
006454  016767  002450  000004          MOV SWEC1,DUMP1B
006462  004567  172560                  JSR R5,PRINTH
006466  000000          DUMP1B: .WORD 0
006470  162701  011134                  SUB #SWBUF1,R1
006474  010103                          MOV R1,R3
006476  012767  011134  000004          MOV #SWBUF1,DUMP1A
006504  004567  172536                  JSR R5,PRINTH
```

```
                                DUMP1A: .WORD 0
006512  105067  004546        DUMP3:  CLRB CHBSY
006516  105767  004631                TSTB OVRUN              ;WAS THERE DATA OVERRUN
006522  001406                        BEQ DUMP4               ;NO
006524  012746  000300                MOV #300,-(SP)          ;YES-SET UP NEW STACK
006530  012746  006540                MOV #DUMP4,-(SP)        ;FROM RTI TO RCVNEC
006534  000167  177070                JMP RCVNOR              ;AND JUMP THERE
006540  000207                DUMP4:  RTS PC                  ;RETURN FROM DUMP
                                ;
                                ;
                                ;
                                ;*****
006542                          GRAPH:  ;DISPLAY DATA GRAPHICALLY
                                ;*****
006542  105267  004516                INCB CHBSY              ;INDICATE I/O BUSY
006546  042767  000010  004512        BIC #10,DISPRD          ;CLR DISPRTN BIT
006554  016700  002350                MOV SREG1,R0            ;GET REG
006560  005720                        TST (R)+
006562  004767  000000G               JSR PC,DECODE           ;INTO R1
006566  042701  177400                BIC #177400,R1
006572  010100                        MOV R1,R0
006574  004767  000000G               JSR PC,GRAPH1           ;GRAPH IN MODE 1
006600  105067  004460                CLRB CHBSY
006604  105767  004503                TSTB OVRUN              ;WAS THERE DATA OVERRUN
006610  001406                        BEQ GRPH1C              ;NO
006612  012746  000300                MOV #300,-(SP)          ;YES-SET STACK FORMAT
006616  012746  006626                MOV #GRPH1C,-(SP)
006622  000167  177062                JMP RCVNOR
006626  000207          GRPH1C: RTS PC                  ;RETURN FROM GRAPH
                                ;
                                ;
```

```
                                        .TITLE  DATA SECTION
                                        ;*******************
                                        ;
                                        ;
                                        ;
                                        ;
                                        .NLIST BEX
000630  000000            SPSAV:  .WORD 0
000632  067     066     065     BITPOS: .ASCIZ /7654321076543210/
000653  060     060     060     BITS:   .ASCIZ /0000000000000000/
000674  015     012     115     MSKIN:  .ASCII <15><12>/MASK UPDATE (Y,N)? : /<200>
000724  107     111     126     IREQ:   .ASCII /GIVE INDEX OR T/<000>
000744  015     012     125     SMSKIN: .ASCII <15><12>/SUBMASK UPDATE (Y,N)? :  /<200>
001000  015     012     123     SELWIN: .ASCII <15><12>/SELECT WORD UPDATE (Y,N)? :  /<200>
001040  015     012     052     STAR:   .ASCII <15><12>/*/<200>
001044  043     200             HASH:   .ASCII /#/<000>
001046  033     014     052     SCREEN: .BYTE 33,14,52,200
001052  040     000             CRLF:   .ASCIZ / /
001054  015     012     043     CROSS:  .ASCII <15><12>/#/<200>
001060  040     072     040     GAP:    .ASCII /  : /<200>
001264  077     111     114     CMDERR: .ASCIZ /?ILL CMD?/
001276  015     012     111     TIMST1: .ASCII <15><12>/INTERVAL FREQUENCY: /<200>
001324  015     012     124     TIMST2: .ASCII <15><12>/TIMER PREVIOUSLY (>3: NO TIME LOG): /<200>
001372  124     111     115     TIMCNT: .ASCII /TIMER COUNT      : /<200>
001416  111     057     117     IOER1:  .ASCII /I/O ERROR ON LH #
001436  000     054             IOER2:  .BYTE 0,54
001440  040     105     122             .ASCII / ERR,RD,I /
001452  000000  000000  000004  IOER3:  .WORD 0,0,0,4
001462  077     111     114     SYCMDE: .ASCII /?ILL SYS CMD?/<15><12>/*/<200>
001503  114     117     147     MES1:   .ASCIZ /LOG MODE DEFINITION/
001527  110     055     114     MES2:   .ASCII /H-LOG OR S-LOG SPECS (H,S) : /<200>
001565  117     114     104     MES3:   .ASCII /OLD, NEW OR UPDATE (O,N,U) : /<200>
001623  116     117     040     MES4:   .ASCII /NO LOG SPECIFIED/
001644  123     120     105     MES5:   .ASCIZ /SPEC FILE FOR SAFE LOG:/
001674  125     120     104     MES6:   .ASCII /UPDATE THIS FILE (Y,)  : /<200>
001720  117     120     105     MES7:   .ASCIZ /OPEN ERROR ON CH=/
002051  114     117     107     MES8:   .ASCII /LOG RECORDING (Y,)    : /<200>
002102  116     125     115     MES9:   .ASCII /MNEMONIC DISPLAY (Y,N) : /<200>
002135  107     122     101     MES10:  .ASCII /GRAPHIC DISPLAY (Y,) : /<200>
002164  123     101     126     MES12:  .ASCII /SAVE FILE (Y,N) : /<200>
002207  123     120     105     MES13:  .ASCII /SPEC FILE NOT OPEN FOR OUTPUT, CREATE NEW FILE (Y,N) : /<200>
002277  040     117     125     MES14:  .ASCIZ / OUTPUT FILE FOR SPECS/
010026  103     117     116     MES15:  .ASCII /CONT. OUTPUT ON OLD FILE (Y,N) : /<200>
010070  040     117     125     MES16:  .ASCIZ / OUTPUT FILE FOR LOG RECORDING/
010127  103     125     111     MES17:  .ASCIZ /CSISPC ERROR/
010144  106     105     124     MES18:  .ASCIZ /FETCH ERROR/
010160  105     116     124     MES19:  .ASCIZ /ENTER ERROR/
010174  110     117     123     MES20:  .ASCIZ /HOST TABLE UPDATE/
010216  123     120     105     MES21:  .ASCII /SPEC FILE-OLD OR UPDATE (O,U) : /<200>
010257  124     101     102     MES22:  .ASCIZ /TABLE FILE:/
010273  125     120     104     MES23:  .ASCII /UPDATE OLD FILE (Y,N) : /<200>
010324  124     101     102     MES24:  .ASCIZ /TABLE TRANSM. ERROR/
010355  123     120     056     MES25:  .ASCIZ /SP.EV. TABLE UPDATE:/
010375  107     111     126     MES26:  .ASCIZ /GIVE EVENTS (* OR T)/
010422  123     124     117     MES27:  .ASCII /STOP ON HARD EVENT (Y,N) : /<200>
010456  015     012     123     MES28:  .ASCII <15><12>/SP.EV.: /<200>
010471  077     110     117     MES29:  .ASCIZ /?H OR NOT SUSPECTED/
```

```
                58 010516  000052  000200          IDER:   .BYTE 52,200    ;ERROR IN TD CODE
                59 010520  000120  000103  000200 ;IDPC:   .ASCII /PC/<200>
                60 010523  000120  000123  000200  IDPS:   .ASCII /PS/<200>
                61 010526  000123  000120  000200  IDSP:   .ASCII /SP/<200>
                62 010531  000103  000120  000200  IDCP:   .ASCII /CP/<200>
                63 010534  000105  000127  000200  IDEW:   .ASCII /EW/<200>
                64 010537  000123  000122  000200  IDSR:   .ASCII /SR/<200>
                65 010542  000102  000106  000200  IDBF:   .ASCII /BF/<200>
                66 010545  000122  000200          IDR:    .ASCII /R/<200>
                67 010547  000110  000124  000200  IDHT:   .ASCII /HT/<200>
                68 010552  000123  000124  000200  IDST:   .ASCII /ST/<200>
                69 010555  000033  000014  000012  CNTMD:  .ASCII <33><14><12>/EVENT COUNT:/<12><0>
                70                                        .EVEN
                71 010576  000040  000060  000060  CNTIX:  .ASCII / 00:/<200>
                72                                        .EVEN
                73 010604  000040  000053  000060  TFIELD: .ASCII <40><53>/0000.00/<0><4><200>
                74 010621  000000                  SSPEV:  .BYTE 0         ;!! SOFT SP.EV. SET
                75 010622  000000                  HSPEV:  .BYTE 0         ;!! HARD SP.EV. SET
                76 010623  000000                  SUSPNO: .BYTE 0         ;!! FILTER SUSPENDED HOST ON SP.EV.
                77 010624  000131  000077  000040  Y:      .ASCII /Y? : /<200>
                78 010632  000116  000077  000040  N:      .ASCII /N? : /<200>
                79 010640  000077  000000          QUES:   .ASCIZ /?/
                80                                        .EVEN
                81 010642  001606                  CMDT1:  CMDI10                  ;CMD DISP. TABLE 1!
                82 010644  001624                          CMDI12                  ;NEXT GROUP AFTER 'N' INPUT
                83 010646  001766                          CMDI17
                84 010650  001412                  CMDT2:  CMDI0                   ;CMD DISP. TABLE 2!
                85 010652  001610                          CMDI11                  ;ERROR DISP. AFTER WRONG Y/N INPUT
                86 010654  001626                          CMDI13
                87 010656  001606                  CMDT3:  CMDI10                  ;CMD DISP. TABLE 3!
                88 010660  001620                          CMDI12                  ;DISP. AFTER T AT INDEX INPUT
                89 010662  001764                          CMDI17
                90 010664  001944                  CMDT4:  CMDIA                   ;CMD DISP. TABLE 4!
                91 010666  001566                          CMDI9                   ;I/O FOR NEW INDEX
                92 010670  001642                          CMDI14
                93                                      ;
                94                                      ;
                95                                      ;SYMBOL TABLE FOR SYSCMD        SYM  INDX
                96                                      ;
                97 010672  000127                  SYMTAB: 127                     ;W   00
                98 010674  000106                          106                     ;F   02
                99 010676  000123                          123                     ;S   04
               100 010700  000110                          110                     ;H   06
               101 010702  000114                          114                     ;L   10
               102 010704  000120                          120                     ;P   12
               103 010706  000124                          124                     ;T   14
               104 010710  000103                          103                     ;C   16
               105 010712  000132                          132                     ;Z   20
               106 010714  000105                          105                     ;E   22
               107 010716  000104                          104                     ;D   24
               108                                      ;
               109                                      ;DISPATCH TABLE FOR WAITLP      INDX
               110                                      ;
               111 010720  005776                  DISPT0: FILTER                  ;00
               112 010722  004214                          RECORD                  ;02
               113 010724  006372                          DUMP                    ;04
               114 010726  006542                          GRAPH                   ;06
```

```
                        ;
                        ;DISPATCH TABLE FOR SYSCMD       INDX
                        ;
010730  002326  DISPT1: WAITUP                  ;00
010732  002334         SETFLG                  ;02
010734  002506         CSIS                    ;04
010736  003214         CSIM                    ;06
010740  003216         CSIL                    ;10
010742  003114         PNOC                    ;12
010744  003166         TERM                    ;14
010746  003122         CONT                    ;16
010750  003574         EMAS                    ;20
010752  003476         SPEVUP                  ;22
010754  002220         ENTOUT                  ;24
                        ;
                        ;       IDENTIFIERS FOR NUM DUMP
                        ;
                        ;       NAME            INDX
                        ;
010756  010516  IDADD:  IDEN            ;0  *
010760  010520          IDPC            ;2  PC
010762  010523          IDPS            ;4  PS
010764  010526          IDSP            ;6  SP
010766  010531          IDCP            ;10 CP
010770  010534          IDEM            ;12 EM
010772  010537          IDSM            ;14 SM
010774  010542          IDHP            ;16 HP
010776  010545          IDP             ;20 P
011000  017316          IDER            ;22 *
011002  017307          IDHT            ;24 HT
011004  010552          IDST            ;26 ST
                        ;
                        ;
                        ;BEGINNING OF COMMUNICATION AREA
                        ;
011006  000016  MASK0:  .WORD "000000000000000000
011010  000000  PC0:    .WORD 0
011012  000000  MASK1:  .WORD "000000000000000000
011014  000000  PC1:    .WORD 0
                        ;
                        ;       .SUBMASK        INDEX
                        ;
011016  077777  MSK005: .WORD "077777           ;0      ;MASK PC,PS,SP FOR EV 1-5
011020  011062  MSK007: .WORD "011062           ;1      ;MASK SEL,PL=2,PS,SP,(PL=2),(52) EV6,7
011022  017076  MSK101: .WORD "017076           ;2      ;MASK SEL,PL,PS,SP,(REG),(BUF),EV 20,21
011024  011076  MSK123: .WORD "011076           ;3      ;MASK SEL,PL,PS,SP,(REG),(BUF), EV 22,23
011026  031016  MSK105: .WORD "031016           ;4      ;MASK =PL=PSW EV 24,25
011050  031062  MSK167: .WORD "031062           ;5      ;MASK =PL= PSW EV=26,27
011032  001602  MSK110: .WORD "001602           ;6      ;MASK SEL,PL,PS,SP,=REG=,REG=,=REG
011034  000000  MSK111: .WORD 0                 ;7      ;EV 31=EMT
011036  000000  MSK112: .WORD 0                 ;10     ;EV 32=TRAP
011040  000000  MSK113: .WORD 0                 ;11     ;EV 33=IOT
                        ;
                        ;       SELECT WORDS FOR CONDITIONAL EVENT REPORT
                        ;
                        ;               INDEX
```

```
                                        ;
                                        ;
011042  000  000          SL0060: .BYTE 0,0  ;W   ;SEL EMT CODE FOR EV 6
011044  000  000                  .BYTE 0,0  ;1
011046  000  000                  .BYTE 0,0  ;2
011050  000  000                  .BYTE 0,0  ;3
011052  000  000          SL0070: .BYTE 0,0  ;4   ;SEL TRAP CODE FOR EV 7
011054  000  000                  .BYTE 0,0  ;5
011056  000  000                  .BYTE 0,0  ;6
011060  000  000                  .BYTE 0,0  ;7
011062  000  000                  .BYTE 0,0  ;10
011064  000  000          SL1100: .BYTE 0,0  ;11  ;SEL REG NR FOR EV 50 #NX
011066  000  000                  .BYTE 0,0  ;12
011070  000  000          SL1101: .BYTE 0,0  ;13
011072  000  000                  .BYTE 0,0  ;14
011074  000  000          SL1102: .BYTE 0,0  ;15
011076  000  000                  .BYTE 0,0  ;16
011100  000  000          SL1103: .BYTE 0,0  ;17
011102  000  000                  .BYTE 0,0  ;20
011104  000002             TIMEON: .WORD 2
011106  000000             TIMLG:  .WORD 0
011110  000000             TREG1:  .WORD 0
011112  000000             TREG2:  .WORD 0
011114  000000             TINT1:  .WORD 0
011116  000000             TINT2:  .WORD 0
011120  002003             ENDCOM: .WORD 3 ;END OF COMM. AREA
                                        ;
011122  000000             XRTII:  .WORD 0,
011124     000             HANG:   .BYTE 0          ;I: SUSPEND ANY LOG ACTION
011125     000             SBUFSW: .BYTE 0          ;0: SBUF 1 RDY, 1: SBUF 2 RDY
011126  000000             SBPTR:  .WORD 0          ;POINTS TO NEXT FREE BUFFER LOC.
011130  000000             SREC1:  .WORD 0          ;POINTS TO FIRST WRD OF CURRENT RECORD
011132  000000             SREC2:  .WORD 0          ;  "    "  LAST "  "     "     "
011134                     SBUF1:  .BLKW 256.       ;BUFFER 1
012134                     SBUF2:  .BLKW 256.       ;BUFFER 2
013134  000000  000000  000000  ESBUF:  .WORD 0,0,0  ;END OF BUFFERS
013142                     SPEVT8: .BLKW 10         ;10 SPECIAL SOFT EVENTS
013162                     COUNT:  .BLKW 51.        ;50 EVENT COUNTERS +1 DUMMY
013264     000             CRNSY:  .BYTE 0          ;SOME OUTPUT LOG BUSY
013265     003             TUFLOP: .BYTE 3          ;TU: 1:TABLE UPDATE FILE OPEN
013266  000000             DISPND: .WORD 0          ;DISPATCH WORD FOR WAITLP
013270  000000             BLKCNT: .WORD 0          ;BLOCK CNT FOR DISK
013272                     AREA:   .BLKW 10         ;SAVE AREA FOR SYS MACROS
                                        .EVEN
                                        ;
                                        ;       LOG SPECIFICATION FILE, LENGTH=LSPFIL
                                        ;
013312     000             SLOGSP: .BYTE 0          ;1: LOG MODE IS DEFINED
013313     000             SLOGUP: .BYTE 0          ;1: LOG FILE UPDATE
013314     000             SLGREC: .BYTE 0          ;1: RECORD LOGGING ON MASS STORAGE
013315     000             SLGNUM: .BYTE 0          ;1: NUMERIC LOG DISPLAY
013316     000             SLGGRP: .BYTE 0          ;1: GRAPGIC LOG DISPLAY
013317     000             SXAXTM: .BYTE 0          ;1: TIME ON X-AXIS; 0 ON X
013320                             .BYTE 0
013321     000             STAXVL: .BYTE 0          ;1: VALUE ONLY;; 0 ON Y
013322  000000             SXAXPO: .WORD 0          ;PERIOD ON X IN SECS
013324                     XEVTAB: .BLKW MAXEV      ;TABLE OF EVENT CODES FOR X-
013504     000             SUBEVT: .BYTE 0          ;0 FOR SUBEVENT 1, 1 FOR SUBE 2
```

```
229 013345  000                 SM3IEV: .BYTE 0          ;TRIGGER EV FOR MODE 3.
230 013346  000                 SMODE3: .BYTE 0          ;DISPLAY MODE 3
231 013347  000                 SM1EV:  .BYTE 1          ;EVENT FOR T  IN MODE 1
232                                     .EVEN
233 013350  000000                      .WORD 0
234                             ;
235                             ;       END OF LOG SPEC FILE
236                             ;
237 013352  000                 SUPDFL: .BYTE 0          ;1: FILE UPDATE, FILE STILL OPEN
238 013353  000                 OVRUN:  .BYTE 0          ;1: S-LOGGING DATA OVERRUN
239                                     .EVEN
240 013354  000000              LCKSV1: .WORD 0          ;SAVE AREA FOR KW11 VECT.
241 013356                      DSKHND: .BLKW 400.       ;DISK HANDLER SPACE
242 015016                      TTYHND: .BLKW 400        ;TTY HANDLER SPACE
243 016016  074503              EXT1:   .RAD50 "SPE"     ;DEF.EXT. FOR LOG SPEC FILES
244 016020  074503                      .RAD50 "SPC"
245                                     .BLKW 2
246 016026  046537              EXT2:   .RAD50 "LOG"     ;DEF.EXT. FOR LOG,REC,OUTPUT
247 016030  046537                      .RAD50 "LOG"
248                                     .BLKW 2
249 016036  076452              EXT3:   .RAD50 "TAH"     ;DEF.EXT. FOR HOST TABLE
250 016040  076452                      .RAD50 "TAH"
251                                     .BLKW 2
252 016046                      OUTSPC: .BLKW 49.        ;BLOCK FOR CSISPC
253 016164  120240              TTSPEC: .RAD50 "TT "     ;BLOCK FOR TTY DEF.
254 016166  011125  052450  021424      .RAD50 "DUMMY.EXT"
255                                     .EVEN
256                                     .TITLE  LOG - HIMOS CONTROL PROGR.
257         001030'                             .END START
```

```
                        .TITLE LOGIO::I/O ROUTINE FOR LOG
                    ;*****************************************************
                    ;*
                    ;* THIS PROGRAM CONTAINS I/O ROUTINES FOR "MINOS"
                    ;* MOST OF THEM ARE OF GENERAL USE AND CAN BE EMPLOYED
                    ;* FOR ANY OTHER PROGRAM. THOSE ARE LISTED IN THE FOLLOWING:
                    ;* LCLK  : 11/2M LINE CLOCK HANDLER;KEEPS TIME IN SECS AND
                    ;*         60TH OF SECS.
                    ;* PRINT : SUBSTITUTE FOR RT-11 ".PRINT" MACRO,AVOIDING TROUBLES
                    ;* OPRINT: PRINT 6 DIGIT OCTAL NUMBER IN R0
                    ;* DECOUT: PRINT NUMBER IN R0 WITH FIELD LENGTH IN R1 IN DECIMAL
                    ;* OCRD  : READ OCTAL NUMBER FROM TERMINAL INTO R1
                    ;* BINRD : READ BINARY NUMBER FROM TERMINAL INTO R1
                    ;* BINPRT: PRINT NUMBER IN R1 IN BINARY
                    ;* DECODE: CONVERT ASCII NUMBER INTO BINARY; RESULT IN R1
                    ;* CMDIN : READ ONE LETTER CMD FROM TTY AND COMPUTE INDEX
                    ;*         ACCORDING TO ARGUMENT LIST OF VALID COMMANDS.
                    ;*
                    ;*****************************************************
                    ;
                            .GLOBL TTYOUT,OPRINT,OCRD,PRTBYT,BINRD,BINPRT
                            .GLOBL BITPOS,BITS,DECODE,INPNR,INDEX3,INDEX
                            .GLOBL PUT4RD,CMDIN,CMDCPR,INDEX2
                            .GLOBL CNTOUT,PRINT,PRTARG,TT.LOG,LCLK
                            .GLOBL CONCNT,CNTRD,CNTBX,COUNT,CRLF
                            .GLOBL HANG,TPIELD,SAPTR,SAVE2,DISPAD
                            .GLOBL ESAVE,SPUPSW,SAVE1,SAVE0
                            .MCALL .REGDEF,.DEF,.TTYOUT,.TTYIN
000000                      .REGDEF
000000*                     .DEF
                    .=020*  .LSECT LOGIO
                            ;
                    ;*****
000000              CNTOUT: ;PRINT COUNT TABLE
                    ;*****
000000  010046              MOV R0,-(SP)
000002  010146              MOV R1,-(SP)
000004  010246              MOV R2,-(SP)
000006  010346              MOV R3,-(SP)
000010  010446              MOV R4,-(SP)
000012  042737  000100  000006    BIC #100,@#CONCNT
000020  012700  000006          MOV #CNTHD,R0            ;PRINT HEADING
000024  004767  000360          JSR PC,PRINT
000030  012703  000016          MOV #CNTIX+1,R3
000034  112720  000060          MOVB #60,(3)+
000040  112713  000060          MOVB #'0,(3)
000044  012704  000026          MOV #COUNT+2,R4
000050  012701  000040          MOV #32.,R1              ;32 COUNTS
000054  012702  000005      CNTOT1: MOV #5,R2            ;5 EVENTS PER LINE
000060  012703  000026      CNTOT2: MOV #CNTIX+2,R3      ;INCR. EV. INDEX
000064  105213                  INCB (3)
000066  005301                  DEC R1
000070  001424                  BEQ CNTOT4               ;FINISHED
000072  122713  000070          CMPB #'0,(R3)            ;OVERFLOW?
000076  001373                  BNE CNTOT3               ;NO
000100  112713  000060          MOVB #60,(3)             ;YES
000104  105203                  INCB -(3)
```

```
                                CNTOT3: MOV #CNTIx,R0           ;PRINT IT
                                        JSR PC,PRINT
                                        MOV (4)+,R0             ;GET COUNT
                                        JSR PC,OPRINT
                                        DEC R2
                                        BNE CNTOT2
                                        MOV #CRLF,R0
                                        JSR PC,PRINT
                                        BR CNTOT1
                                CNTOT4: MOV #CRLF,R0
                                        JSR PC,PRINT
                                        BIS #100,@#COMCHI
                                        MOV (SP)+,R4
                                        MOV (SP)+,R3
                                        MOV (SP)+,R2
                                        MOV (SP)+,R1
                                        MOV (SP)+,R0
                                        RTS PC                  ;RETURN FROM CNTOUT
                                ;
                                ;*****
                                TIMLOG: ;APPEND SYS-TIME TO A RECORD
                                ;*****
                                        MOV R1,-(SP)
                                        MOV R2,-(SP)
                                        TSTB HANG               ;LOGGING ACTIVE?
                                        BNE TMLOG5              ;NO
                                        MOV #IFIELD,R1
                                        MOV SBPTR,R2
                                        DEC R2                  ;BACKUP OVER EOT
                                TMLOG1: CMP R2,#SBUF2           ;END OF BUFFER 1?
                                        BNE TMLOG2              ;NO
                                        BIS #2,DISPWD           ;YES-DISPATCH BUFFER RECORDING
                                TMLOG2: CMP R2,#ESBUF           ;END OF BUFFER 2?
                                        BEQ TMLOG4              ;YES-WRAP AROUND
                                TMLOG3: MOVB (1)+,(2)+          ;STORE IT
                                        CMPB (1),#200           ;END OF IFIELD?
                                        BNE TMLOG1              ;NO- EXIT
                                        BIT #1,R2               ;EVEN BOUNDARY?
                                        BEQ TMLG3A              ;YES
                                        CLRB (2)+               ;NO-ADJUST TO EVEN BOUNDARY
                                TMLG3A: MOV R2,SBPTR            ;ADJUST RECORD END POINTERS
                                        MOV R2,SBEC2
                                        BR TMLOG5               ;YES-FINISH
                                TMLOG4: BIS #2,DISPWD           ;DISPATCH BUFFER RECORDING
                                        MOV #SBUF1,R2           ;WRAP AROUND
                                        INCB SBUFSW             ;SWITCH BACK TO BUF 1
                                        BR TMLOG4
                                TMLOG5: MOV (SP)+,R2
                                        MOV (SP)+,R1
                                        RTS PC                  ;RETURN FROM TMLOG
                                ;
                                ;*****
                                LCLK1:  ;LINE CLOCK INTERRUPT HANDLER
                                ;*****
                                        MOV R0,-(SP)
                                        MOV #IFIELD+11,R0
                                        INCB (R)
```

```
                115 000332  122710  000072          CMPB #72,(0)            ;OVERFLOW?
                116 000336  001401                  BEQ LCLK1               ;YES
                117 000340  000421                  BR LCLK5                ;NO-DONE
                118 000342  112710  000060  LCLK1:  MOVB #60,(0)
                119 000346  105200                  INCB =(0)
                120 000350  122710  000066          CMPB #66,(0)            ;OVERFLOW?
                121 000354  001401                  BEQ LCLK2               ;YES
                122 000356  000412                  BR LCLK5                ;NO-DONE
                123 000360  112710  000060  LCLK2:  MOVB #60,(0)
                124 000364  105740                  TSTB =(R0)              ;SKIP PERIOD
                125 000366  000402                  BR LCLK4
                126 000370  112710  000060  LCLK3:  MOVB #60,(0)
                127 000374  105240  LCLK4:  INCB =(0)
                128 000376  122710  000072          CMPB #72,(0)            ;OVERFLOW?
                129 000422  001772                  BEQ LCLK3               ;YES
                130 000424  012600  LCLK5:  MOV (SP)+,R0
                131 000426  000302                  RTI                     ;RETURN FROM LCLK
                132                                 ;
                133                                 ;
                134                                 ;
                135                                 ;*****
                136 000430                          PRINT:  ;PRINT TEXT STARTING AT (R0)
                137                                 ;*****
                138 000430  010146                  MOV R1,-(SP)
                139 000432  010001                  MOV R0,R1
                140 000434  112100  PRINT0: MOVB (1)+,R0
                141 000436                          .TTYOUT
                142 000442  105737  0000006         TSTB @#TTYOUT
                143 000446  100375                  BPL .-6
                144 000450  105700                  TSTB R0
                145 000452  001404                  BEQ PRINT1
                146 000454  122700  000200          CMPB #200,R0
                147 000460  001411                  BEQ PRINT2
                148 000462  000764                  BR PRINT0
                149 000464  112700  000015  PRINT1: MOVB #15,R0
                150 000470                          .TTYOUT
                151 000474  112700  000012          MOVB #12,R0
                152 000500                          .TTYOUT
                153 000504  105737  0000006 PRINT2: TSTB @#TTYOUT
                154 000510  100375                  BPL .-6
                155 000512  012601                  MOV (SP)+,R1
                156 000514  000207                  RTS PC                  ;RETURN FROM PRINT
                157                                 ;*****
                158 000516                          OPRINT: ;PRINT OCTAL NR, 6 DIGITS IN R0
                159                                 ;*****
                160 000516  010046                  MOV R0,-(SP)
                161 000520  010146                  MOV R1,-(SP)
                162 000522  010246                  MOV R2,-(SP)
                163 000524  010346                  MOV R3,-(SP)
                164 000526  012703  000600'         MOV #OPRINA,R3
                165 000532  012702  000006          MOV #6,R2
                166 000536  112743  000040          MOVB #40,-(3)
                167 000542  112743  000200          MOVB #200,-(3)
                168 000546  010001  OPRIN1: MOV R0,R1
                169 000550  042701  177770          BIC #177770,R1
                170 000554  062701  000060          ADD #60,R1
                171 000560  112143                  MOVB R1,-(3)
```

```
                                CLC
                                ROR R0
                                ASR R0
                                ASR R0
                                DEC R2
                                BNE OPRIN1
000550 012700  000570'  OPRIN2: MOV #OPRIN8,R0
000554 004767  177622           JSR PC,PRINT
000560 000404                   BR OPRIN4
000570                  OPRIN8: .BLKW 8
000570 012603           OPRIN1: MOV (SP)+,R3
000574 012602                   MOV (SP)+,R2
000600 012601                   MOV (SP)+,R1
000604 012600                   MOV (SP)+,R0
000610 000207                   RTS PC                  ;RETURN FROM OPRINT
                        ;
                        ;*****
000612                  OCRD:   ;READ OCTAL NUMBER; LINKAGE: PC
                        ;*****  ;R0=-1 IF T, =0 IF EMPTY; RESULT IN R1
                        ;
000612 010346                   MOV R3,-(SP)
000614 005001                   CLR R1                  ;RESULT BUFFER
000616 005003                   CLR R3                  ;SHIFT COUNTER
000620 005046                   CLR -(SP)               ;START MARKER ON TOP
000622                  OCRD1:  .TTYIN
000626 042700  000200           BIC #200,R0
000632 122700  000012           CMPB #12,R0
000636 001402                   BEQ OCRD3
000640 010046                   MOV R0,-(SP)    ;STACK NUMBER
000642 000767                   BR OCRD1
000644 005726           OCRD3:  TST (SP)+
000646 005716                   TST (SP)                ;TOP OF STACK?
000650 001423                   BEQ OCRD5       ;YES
000652 022716  000124           CMP #124,(SP)   ;IF
000656 001415                   BEQ OCRD4
000660 162716  000060           SUB #60,(SP)    ;CONVERT
000664 011600                   MOV (SP),R0
000666 010057  177504           MOV R0,@#177504 ;SHIFT UP 3 TIMES LEFT
000672 010057  177506           MOV R0,@#177506
000676 013700  177504           MOV @#177504,R0
000702 000301                   SWAB R0
000704 062703  000003           ADD #3,R3       ;FINAL SHIFT CNT
000710 000755                   BR OCRD3
000712 012700  177777   OCRD4:  MOV #-1,R0               ;-1 IF T
000716 000752                   BR OCRD3
000720 005703           OCRD5:  TST R3
000722 001402                   BNE OCRD6
000724 052700  177775           BIS #177775,R0          ;-2 IF EMPTY
000730 005726           OCRD6:  TST (SP)+
000732 012603                   MOV (SP)+,R3
000734 000207                   RTS PC                  ;RETURN FROM OCIN
                        ;
                        ;*****
000736                  PRTBYT: ;PRINT BYTE OCTAL; LINKAGE: PC
                        ;*****  ;VALUE IN R0
                        ;
```

```
                        ;******
                        PRTWRD: ;PRINT WORD OCTAL; LINKAGE: PC
                        ;******  ;VALUE IN R0
                        ;
                                MOV R1,-(SP)
                                MOV R2,-(SP)
                                BIC #177400,R0          ;CLEAR UPPER BYTE
                                MOV #3,R1               ;PRINT 3 DIGITS
                                BR TYP1
                        ;******
                        PRTWRD: ;PRINT WORD OCTAL; LINKAGE: PC
                        ;******  ;VALUE IN R0
                        ;
                                MOV R1,-(SP)
                                MOV R2,-(SP)
                                MOV #6,R1               ;PRINT 6 DIGITS
                        TYP1:   MOV R1,R2               ;SAVE COUNT
                        TYP2:   MOV R0,-(SP)
                                BIC #177770,(SP)
                                CLC
                                ROR R0
                                ASR R2
                                ASR R2
                                DEC R1
                                BNE TYP2
                        TYP3:   MOV (SP)+,R0
                                DEC R2                  ;LAST DIGIT?
                                BEQ TYP4                ;YES
                                TST R1                  ;ANY TYPED YET?
                                BNE TYP4                ;YES
                                TST R0                  ;NO,LEADING 0?
                                BEQ TYP5                ;YES, SKIP
                        TYP4:   ADD #'0,R0
                                .TTYOUT
                                INC R1
                        TYP5:   TST R2                  ;DONE?
                                BGE TYP3                ;NO
                                MOV (SP)+,R2
                                MOV (SP)+,R1
                                TSTB @#177564
                                BPL .-6
                                RTS PC                  ;RETURN FROM PRTWRD/PRT
                        ;
                        ;******
                        BINRD:  ;READ WORD BINARY; LINKAGE: PC
                        ;******  ;WORD ADDR: R1
                        ;
                                MOV R3,-(SP)
                                MOV R2,-(SP)
                                CLR R2
                                CLR R3
                        BINRD1: .TTYIN
                                BIC #^C377,R0
                                CMPB #15,R0
                                BEQ BINRD3
                                INC R3
                                ASL R2
                                CMPB #60,R0
                                BEQ BINRD1
                                BIS R0,R2
                                BR BINRD1
```

```
                286 001126  005703          BINRD3: TST R3                  ;EMPTY WORD
                287 001130  001401                  BEQ BINRD4             ;STORE NEW WORD
                288 001132  010211                  MOV R2,(1)             ;TO BIT BINARY BUFFER
                289 001134                  BINRD4: ,TTYIN
                290 001134  012602                  MOV (SP)+,R2
                291 001142  012603                  MOV (SP)+,R3
                292 001144  000207                  RTS PC                 ;RETURN FROM BINRD
                293                 ;
                294                 ;*****
                295 001146                  BINPRT: ;PRINT WORD BINARY; LINKAGE: PC
                296                 ;*****  ;WORD ADDR: R1
                297                 ;
                298 001146  010146                  MOV R1,-(SP)
                299 001150  012700  000000G         MOV #BITPOS,R0         ;PRINT BIT IDENTIFIER STRING
                300 001154  004767  177230          JSR PC,PRINT
                301 001160  011102                  MOV (1),R2             ;DATA TO BE PRINTED
                302 001162  012701  000020G         MOV #BITS,R1
                303 001166  032702  100000  BINPT1: BIT #100000,R2          ;CHECK LEFTMOST BIT
                304 001172  001403                  BEQ BINPT2
                305 001174  112721  000061          MOVB #61,(1)+
                306 001200  000402                  BR BINPT3
                307 001202  112721  000060  BINPT2: MOVB #60,(1)+
                308 001206  006302  BINPT3: ASL R2
                309 001210  020127  000320G         CMP R1,#BITS+16.
                310 001214  001364                  BNE BINPT1
                311 001216  012700  000000G         MOV #BITS,R0
                312 001222  004767  177162          JSR PC,PRINT
                313 001226  012601                  MOV (SP)+,R1
                314 001230  000207                  RTS PC                 ;RETURN FROMBINPRT
                315                 ;
                316                 ;*****
                317 001232                  DECODE: ;CONVERT ASCII NUMBER TO BIN; LINKAGE: PC
                318                 ;*****  ;R0 POINTS TO MOST SIGN. CHAR. OF ASCII STRING
                319                 ;RESULT IN R1
                320                 ;
                321 001232  010046                  MOV R0,-(SP)
                322 001234  010346                  MOV R3,-(SP)
                323 001236  005001                  CLR R1                 ;RESULT BUFFER
                324 001240  005003                  CLR R3                 ;SHIFT CNT
                325 001242  005046                  CLR -(SP)              ;MARK T.O.S. STACK
                326 001244  142710  000200  DECOD1: BICB #200,(0)          ;CLEAR PARITY
                327 001250  121027  000060          CMPB (0),#60           ;CHECK IF NUMERIC
                328 001254  100405                  BMI DECOD2             ;NO
                329 001256  122710  000071          CMPB #71,(0)
                330 001262  100402                  BMI DECOD2             ;NO
                331 001264  112046                  MOVB (0)+,-(SP)        ;PUSH-STORE IT REVERSED
                332 001266  000766                  BR DECOD1              ;NEXT
                333 001270  005716  DECOD2: TST (SP)                       ;TOP OF STACK?
                334 001272  001416                  BEQ DECOD3             ;YES
                335 001274  162716  000060          SUB #'0,(SP)           ;ASCII-CONVERT
                336 001300  011600                  MOV (SP),R0
                337 001302  010037  177304          MOV R0,@#177304        ;SHIFT 7 TIMES LEFT
                338 001306  012737  177516          MOV #17,@#177516
                339 001312  013700  177304          MOV @#177304,R0
                340 001316  050001                  BIS R0,R1
                341 001320  062703  000003          ADD #3,R3              ;INCR. SHIFT CNT
                342 001324  000726                  BR DECOD2
```

```
                        ;
     344 001526  000760                 BR DECOD2
     345 001530  005726          DECOD3: TST (SP)+
     346 001532  012603                 MOV (SP)+,R3
     347 001534  012600                 MOV (SP)+,R0
     348 001536  000207                 RTS PC                  ;RETURN FROM DECOD
     349                        ;
     350                        ;
     351                        ;
     352                        ;*****
     353 001540                 IOERR:  ;CREATE TEXT FOR I/O ERRORS ON CHANNEL I
     354                        ;*****  ;CH-NO=1ST ARG., LINKAGE=R5
     355                        ;
     356 001540  010046                 MOV R0,-(SP)
     357 001542  010146                 MOV R1,-(SP)
     358 001544  062715  000060         ADD #60,(5)
     359 001550  112567  000000G        MOVB (5)+,IOER2         ;GET CH-NO
     360 001554  005205                 INC R5                  ;CORRECT LINKAGE REG.
     361 001556  013700  000052         MOV #52,R0              ;GET I/O ERROR WORD
     362 001562  012701  000000G        MOV #IOER3,R1           ;STARTING ADD. OF BUFFER
     363 001566  004767  000016         JSR PC,PUTWRD
     364 001572  012700  000000G        MOV #IOER1,R0
     365 001576  004767  177006         JSR PC,PRINT            ;PRINT ERROR MESSAGE
     366 001602  012601                 MOV (SP)+,R1
     367 001604  012600                 MOV (SP)+,R0
     368 001606  000205                 RTS R5                  ;RETURN FROM IOERR
     369                        ;
     370                        ;*****
     371 001610                 PUTWRD: ;PUT R0 CONV. TO 6 DIGIT ASCII IN LOC
     372                        ;*****  ;STARTING AT R1; LINKAGE: PC
     373                        ;
     374 001610  010246                 MOV R2,-(SP)
     375 001612  012702  000006         MOV #6,R2               ;6 DIGITS
     376 001616  010046          PUTWD1: MOV R0,-(SP)            ;STACK THEM DOWN
     377 001620  042716  177770         BIC #177770,(SP)
     378 001624  062716  000060         ADD #60,(SP)
     379 001630  000241                 CLC
     380 001632  006000                 ROR R0
     381 001634  006200                 ASR R0
     382 001636  006200                 ASR R0
     383 001640  005302                 DEC R2
     384 001642  001365                 BNE PUTWD1
     385 001644  012702  000006         MOV #6,R2               ;POP REVERSED SEQUENCE
     386 001650  112621          PUTWD2: MOVB (SP)+,(1)+
     387 001652  005302                 DEC R2
     388 001654  001375                 BNE PUTWD2
     389 001656  012602                 MOV (SP)+,R2
     390 001660  000207                 RTS PC                  ;RETURN FROM PUTWRD
     391                        ;*****
     392 001662                 CMDIN:  ;CMD INPUT; LINKAGE: R5
     393                        ;*****  ;RESULT IN R0:-1=ERR, 0=CR, 1=CMD(1)
     394                        ;
     395 001662  010146                 MOV R1,-(SP)
     396 001664  010246                 MOV R2,-(SP)
     397 001666  005001                 CLR R1
     398 001670  005002                 CLR R2
     399 001672                 CMDIN1: .TTYIN                  ;GET CMD CHARACTER
     400 001676  042700  000200         BIC #200,R0
     401 001702  122700  000000         CMPB #0,R0
```

```
402 001506 001771                   BEQ CMDIN1
403 001510 122700 000015            CMPB #15,R0             ;CR?
404 001514 001302                   BNE CMDIN2
405 001516 005202                   INC R2                  ;FLAG CR RECEIVED
406 001520 000412                   BR CMDIN4
407 001522 122715 000000    CMDIN2: CMPB #0,(5)             ;END OF ARG LIST?
408 001526 001404                   BEQ CMDIN3              ;YES=ERROR
409 001530 005201                   INC R1                  ;BUMP INDEX
410 001532 120025                   CMPB R0,(5)+            ;IS CHAR IN LIST?
411 001534 001404                   BEQ CMDIN4              ;YES
412 001536 000771                   BR CMDIN2               ;NO
413 001540 012701 177777    CMDIN3: MOV #-1,R1              ;FLAG ERROR
414 001544 000404                   BR CMDIN5
415 001546 105715           CMDIN4: TSTB (5)                ;FIND END OF LIST
416 001550 001422                   BEQ CMDIN5
417 001552 005205                   INC R5
418 001554 000774                   BR CMDIN4
419 001556 032705 000001    CMDIN5: BIT #1,R5               ;R5 EVEN?
420 001562 001401                   BNE CMDIN6              ;NO
421 001564 005205                   INC R5
422 001566 005205           CMDIN6: INC R5
423 001570 005702                   TST R2                  ;CR IN?
424 001572 001407                   BNE CMDIN8              ;YES
425 001574                  CMDIN7: .TTYIN                  ;GET CR
426 001600 042700 000200           BIC #200,R0
427 001604 122700 000015           CMPB #15,R0
428 001610 001371                   BNE CMDIN7
429 001612                  CMDIN8: .TTYIN                  ;GET LF
430 001616 010100                   MOV R1,R0
431 001620 012602                   MOV (SP)+,R2
432 001622 012601                   MOV (SP)+,R1
433 001624 005700                   TST R0                  ;ERROR?
434 001626 100401                   BMI CMDIN9              ;YES
435 001630 000205                   RTS R5                  ;NO=RETURN FROM CMDIN
436 001632 012700 000000    CMDIN9: MOV #CMDERR,R0
437 001636 004767 176546            JSR PC,PRINT
438 001642 012700 177777            MOV #-1,R0
439 001646 000205                   RTS R5                  ;RETURN FROM CMDIN
440
441                 000001'         .END
```

```
                                        .TITLE LOGGRF: GRAPHICS FOR LOGGING
                                    ;*
                                    ;*************************************************************
                                    ;*
                                    ;* ROUTINES FOR GRAPHIC DISPLAY OF 3-40K EVENTS IN PRIMOS
                                    ;* A HISTOGRAM IS DRAWN ON THE TEKTRONIX 4012 SCREEN.
                                    ;* EVENTS ARE ENTERED ALONG THE X-AXIS, THEIR FREQUENCY IS
                                    ;* PLOTTED ON THE Y-AXIS.
                                    ;* UPON OVERFLOW, THE PICTURE IS RESCALED BY FACTOR 2.
                                    ;*
                                    ;*************************************************************
                                    ;*
                                        .MCALL .REGDEF,.TTYIN,.TTYOUT,.DEF
                                        .GLOBL DECOUT,OCOUTV,ATEXT,PRINT
                                        .GLOBL GRINAI,PICTI,DRAWAI
                                        .REGDEF
                                        .DEF
000006                                  STACK=SP
                                    ;
                                        .TITLE PLOTTING I/O ROUTINES
                                        ;************************
                                    ;
                                    ;
                                    ;*****
000000                              CHINI:  ;READ AN 8 BIT CHAR FROM TEKTRONIX
                                    ;*****
000000                                  .TTYIN
000004  000205                          RTS R5
                                    ;
                                    ;*****
00000e                              CHOUT:  ;PUT AN 8-BIT CHAR. TO TEKTRONIX
                                    ;*****
000006                                  .TTYOUT
000012  105737  177564                  TSTB @#TPS
000016  100375                          BPL .-4
000020  000205                          RTS R5
                                    ;
                                    ;*****
000022                              SAVE:   ;SAVE REGISTERS
                                    ;*****
000022  010267  001656                  MOV R0,SAVAR
000026  012700  001706'                 MOV #SAVAR+2,R0
000032  010120                          MOV R1,(0)+
000034  010220                          MOV R2,(0)+
000036  010320                          MOV R3,(0)+
000040  010420                          MOV R4,(0)+
000042  010520                          MOV R5,(0)+
000044  000207                          RTS PC
                                    ;
                                    ;*****
000046                              UNSAVE: ;RESTORE REGISTERS
                                    ;*****
000046  012700  001706'                 MOV #SAVAR+2,R0
000052  012001                          MOV (0)+,R1
000054  012002                          MOV (0)+,R2
000056  012003                          MOV (0)+,R3
000060  012004                          MOV (0)+,R4
```

```
                                        MOV (0)+,R5
                                        MOV SAVAR,R0
                                        RTS PC
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;       TPLOT
                                ;
                                ;       THIS ROUTINE IS CALLED TO PLOT
                                ;       IN VECTOR,OR POINT
                                ;       PLOT MODE DEPENDING ON THE
                                ;       VALUE OF REG 0 AS DESCRIBED BELOW.
                                ;
                                ;       IF
                                ;       A=0 INITIALIZE AND DARK VECTOR TO X,Y
                                ;
                                ;       A>0 BRIGHT VECTOR TO X,Y
                                ;
                                ;       A<0 POINT PLOT TO X,Y
                                ;
                                ;
                                ;
000072 010067 001622    TPLOT:  MOV     R0,TPTMOD       ;CHECK REG 0,TPTMOD
000076 010046                   MOV     R0,-(SP)        ;SAVE R0 ON STACK
000100 001425                   BEQ     TPTDV           ;JUMP IF INIT. AND DARK VELT
000102 100410                   BPL     TPTNRM          ;JUMP IF NORMAL VECTOR
000104 012700 000037            MOV     #37,R0
000110 004567 177672            JSR     R5,CHOUT
                                ;
000114 012700 000035    TPTDV:  MOV     #35,R0          ;OUTPUT A GS TO INITIALIZE
000120 004567 177662            JSR     R5,CHOUT
                                ;
000124 012500           TPTNRM: MOV     (R5)+,R0        ;MOV X COORD TO REG 0
000126 100001                   BPL     TPT10           ;JUMP IF GEQ 0
000130 005000                   CLR     R0              ;IF LSS SET TO 0
000132 020027 002000    TPT10:  CMP     R0,#1024.       ;CHECK FOR 0, SCREEN
000136 100402                   BMI     TPT12           ;JUMP IF IN RANGE
000140 012700 001777            MOV     #1023.,R0       ;SET TO EDGE IF TOO HIGH
000144 010067 001552    TPT12:  MOV     R0,TPTX         ;SAVE X VALUE
                                ;
000150 012500                   MOV     (R5)+,R0        ;GET Y COORD
000152 100001                   BPL     TPT14           ;JUMP IF GEQ 0
000154 005000                   CLR     R0
000156 020027 001415    TPT14:  CMP     R0,#781.        ;CHECK FOR TOO LARGE Y
000162 100402                   BMI     TPT16           ;JUMP IF IN RANGE
000164 012700 001414            MOV     #780.,R0        ;MOVE TO EDGE OF SCREEN
000170 010067 001530    TPT16:  MOV     R0,TPTY         ;SAVE Y VALUE
                                ;
000174 016700 001524    TPTPNT: MOV     TPTY,R0         ;GET Y VALUE
000200 006100                   ROL     R0              ;MOVE UPPER 5 BITS
000202 006100                   ROL     R0              ;TO UPPER BYTE
000204 006100                   ROL     R0
000206 000300                   SWAB    R0              ;SWAP UPPER AND LOWER BYTE
000210 042700 177740            BIC     #177740,R0      ;MASK OFF EXTRA
000214 052700 000040            BIS     #40040,R0       ;SET IN HI Y TAG
```

```
    000220  004567  177562          JSR     R5,CHOUT        ;OUTPUT HI Y
    000224  016700  001474          MOV     TPIY,R0         ;GET Y COORD
    000230  042700  177740          BIC     #177740,R0      ;MASK TO LOW 5 BITS
    000234  052700  000140          BIS     #000140,R0      ;AND SET LOW Y TAG
    000240  004567  177542          JSR     R5,CHOUT        ;SHIP OUT LOW Y BITE

    000244  016700  001452          MOV     TPIX,R0         ;GET X COORD
    000250  006100                  ROL     R0              ;AND ADJUST LIKE Y
    000252  006100                  ROL     R0
    000254  006100                  ROL     R0
    000256  000300                  SWAB    R0              ;SWITCH BYTES
    000260  042700  177740          BIC     #177740,R0      ;MASK OFF EXTRA
    000264  052700  000040          BIS     #000040,R0      ;SET HI X TAG
    000270  004567  177512          JSR     R5,CHOUT        ;OUTPUT HI X BYTE
    000274  016700  001422          MOV     TPIX,R0         ;GET X COORD
    000300  042700  177740          BIC     #177740,R0      ;LEAVE ONLY LOW BITS
    000304  052700  000100          BIS     #000100,R0      ;SET IN LOW X BITS
    000310  004567  177472          JSR     R5,CHOUT        ;OUTPUT LOW X BYTE
    000314  005767  001400          TST     TPIMOD          ;CHECK FOR POINT PLOT
    000320  100003                  BPL     RTN             ;RETURN
    000322  005067  001372          CLR     TPIMOD          ;CLEAR AND BRIGHT VECTOR
    000326  000722                  BR      TPIPNT
                            ;
    000330  012600          RTN:    MOV     (SP)+,R0        ;RESTORE R0 AND EXIT
    000332  000205                  RTS     R5      ;RETURN FROM TPLOT
                            ;
```

```
                                        .TITLE  GRAPH ROUTINES
                                        ;**********************
                                ;
                                ;*****
000534                          GRINM1: ;INITIALIZE FOR GRAPH MODE 1
                                ;*****
000534  010046                          MOV R0,-(SP)
000536  010146                          MOV R1,-(SP)
000540  010246                          MOV R2,-(SP)
000542  010346                          MOV R3,-(SP)
000544  012703  001614'                 MOV #YPT,R3
000550  012700  001544'                 MOV #XPT,R0
000554  012701  000305                  MOV #197.,R1
000560  012702  000024                  MOV #20.,R2
000564  010120                  GRIN11: MOV R1,(R0)+            ;SET XPT TO STARTING VAL.
000566  005023                          CLR (3)+                ;FOR COL. AND CLEAR YPT
000570  062701  000052                  ADD #42.,R1
000574  005302                          DEC R2
000576  001372                          BNE GRIN11
                                ;
000600  012767  000010  001256          MOV #8.,DIVISR  ;INITIALLY 8 LINES PER EVENT
000606  005067  001254                  CLR ITEM                ;# COLS. IN GRAPH
000612  012767  000022  001250          MOV #18.,WIDTH          ;WIDTH PER COL.
000620  012700  001672'                 MOV #YTEXT,R0
000624  012701  000004                  MOV #4,R1
000630  012702  000070                  MOV #56.,R2
000634  010220                  GRIN13: MOV R2,(0)+             ;SET VALUES FOR Y-AXIS
000636  162702  000016                  SUB #14.,R2             ;VALUE DESCRIPTION
000642  005301                          DEC R1
000644  001373                          BNE GRIN13
000646  005010                          CLR (0)                 ;NOTTD4 ALWAYS 0
000650  012603                          MOV (SP)+,R3
000652  012602                          MOV (SP)+,R2
000654  012601                          MOV (SP)+,R1
000656  012600                          MOV (SP)+,R0
000660  000207                          RTS PC                  ;RETURN FROM GRINM1 0
                                ;
                                ;*****
000662                          PICT1:  ;INITIALIZE GRAPH FOR MODE 1; LINKAGE: PC
                                ;*****  ;DRAW PICTURE AS IN XPT AND YPT
                                ;
000662  004767  177534                  JSR PC,SAVE
000666  012700  001726'                 MOV #SCREEN,R0          ;ERASE SCREEN
000672  004767  000000G                 JSR PC,PRINT
000676  005003                          CLR R3
000700  012700  002024'                 MOV #24,R0              ;PRINT 4 BLANC
000704  004767  000000G                 JSR PC,PRINT
000710  012701  000006                  MOV #6,R1               ;FIELD LENGTH FOR DECOUT
000714  016300  001672'                 MOV YTEXT(3),R0         ;PRINT TEXT FOR Y-AXIS
000720  004767  000000G                 JSR PC,DECOUT
000724  005723                  PICT14: TST (3)+
000726  020327  000012                  CMP R3,#10.             ;FINISHED?
000732  001411                          BEQ PICT15              ;YES
000734  012700  002431'                 MOV #TX1,R0
000740  004767  000000G                 JSR PC,PRINT
000744  016300  001672'                 MOV YTEXT(3),R0
```

```
    58 000550  004767  000000G              JSR PC,DECOUT
    59 000554  000763                       BR PICT14
    60 000556  012700  002124'     PICT15:  MOV #GAP,R0          ;PRINT TEXT FOR X-AXIS
    61 000562  004767  000000G              JSR PC,PRINT
    62 000566  016703  001374               MOV ITEM,R3           ;NR OF EV
    63 000572  005002                       CLR R2
    64 000574  005703               PICT16: TST R3                ;FINISHED?
    65 000576  001410                       BEQ PICT17            ;YES
    66 000600  016200  001474'              MOV ECTAB(2),R0
    67 000604  004567  000200G              JSR R5,OCOUTV         ;PRINT EC
    68 000610  000002                       .WORD 2               ;LENGTH 2
    69 000612  005743                       TST -(3)
    70 000614  005722                       TST (2)+
    71 000616  000766                       BR PICT16
    72 000620  012738  002145'     PICT17:  MOV #CRLF,R0          ;PRINT NAME OF X-AXIS
    73 000624  004767  000000G              JSR PC,PRINT
    74 000630  012700  001731'              MOV #BGS,R0
    75 000634  004767  000000G              JSR PC,PRINT
    76 000640  012700  002112'              MOV #XTEXT,R0
    77 000644  004767  000000G              JSR PC,PRINT
    78 000650  005000                       CLR R0
    79 000652  004567  177214               JSR R5,IPLOT          ;SET GRAPH MODE
    80 000656  000262                       .WORD 178.
    81 000660  001410                       .WORD 7+8.
    82 000662  005200                       INC R0                ;SFT LIGHT VECTOR
    83 000664  004567  177202               JSR R5,IPLOT          ;DRAW Y-AXIS
    84 000670  000262                       .WORD 178.
    85 000672  000511                       .WORD 329.
    86 000674  004567  177172               JSR R5,IPLOT          ;DRAW X-AXIS
    87 000700  001774                       .WORD 1020.
    88 000702  000511                       .WORD 329.
    89 000704  005001                       CLR R1                ;CNT # OF EXISTING EVENTS
    90 000706  026701  000754       PICT11: CMP ITEM,R1           ;ALL EV COL. DRAWN?
    91 000712  001404                       BEQ PICT13            ;YES
    92 000714  005002                       CLR R2                ;NO-DRAW TEXT
    93 000716  016103  001614'              MOV YPT(1),R3         ;GET NR OF LINES
    94 000722  016167  001544'  000036      MOV XPT(1),PICT1A     ;SET ARGUMENTS
    95 000730  016167  001544'  000042      MOV XPT(1),PICT1C
    96 000736  066767  000726  000034      ADD WIDTH,PICT1C
    97 000744  012767  000512  000016      MOV #512.,PICT16
    98 000752  012767  000512  000022      MOV #512.,PICT10
    99 000760  005722               PICT12: CLR R0                ;POSITION VECTOR
   100 000762  004567  177104               JSR R5,IPLOT
   101 000766  000000               PICT1A: .WORD 0
   102 000770  000000               PICT1B: .WORD 0
   103 000774  005200                       INC R0
   104 000776  004567  177072               JSR R5,IPLOT
   105 001002  000000               PICT1C: .WORD 0
   106 001004  000000               PICT1D: .WORD 0
   107 001006  004567  177760               INC PICT1B            ;INC Y-COORD.
   108 001012  005267  177766               INC PICT1D
   109 001016  005303                       DEC R3                ;DEC LINE CNT
   110 001020  001360                       BNE PICT12            ;NEXT LINE
   111 001022  005721                       TST (1)+              ;BUMP EV CNT
   112 001024  000751                       BR PICT11             ;NEXT COL.
   113 001026  000000               PICT13: CLR R0                ;SET VECTOR TO RETURN LOC.
   114 001030  004567  177040               JSR R5,IPLOT
```

```
                                        .WORD 0
                                        .WORD 242.
001032 000000
001034 000362
001036 012700  002147'                  MOV #ALPHA,R0            ;SET ALPHA MODE
001042 004767  0000006                  JSR PC,PRINT
001046 004767  176774                   JSR PC,UNSAVE
001052 000207                           RTS PC                   ;RETURN FROM PICT1
                                        ;
                                        ;*****
001054                                  DRAWM1: ;DRAW COLUMN INCREMENT FOR GRAPH MODE 1
                                        ;*****  ;EC IN R4; LINKAGE: PC
                                        ;
001054 010046                           MOV R0,-(SP)
001056 010146                           MOV R1,-(SP)
001060 010246                           MOV R2,-(SP)
001062 010546                           MOV R5,-(SP)
001064 005001                           CLR R1                   ;TABLE INDEX
001066 026701  000574                   DRM11: CMP ITEM,R1       ;NEW EVENT?
001072 001405                           BEQ DRM12                ;YES
001074 020261  001474'                  CMP R0,ECTAB(1)          ;NO-SEARCH FOR EC
001100 001404                           BEQ DRM14                ;FOUND
001102 005721                           TST (1)+                 ;NOT FOUND-NEXT
001104 000770                           BR DRM11
001106 020127  000052                   DRM12: CMP R1,#42.       ;NEW EVENT-TOO MANY?
001112 100406                           BMI DRM13                ;NO-ENTER IT IN TABLE
001114 012700  002046'                  MOV #TY2,R0              ;YES-EVENT OVERFLOW
001120 004767  0000006                  JSR PC,PPRINT
001124 000167  000532                   JMP DRM11B               ;EXIT
                                        ;
001130 062767  000002  000530           DRM13: ADD #2,ITEM       ;ENTER NEW EV IN TABLE
001136 010061  001474'                  MOV R0,ECTAB(1)
                                        ;
001142 016102  001544'                  MOV XPT(1),R2            ;ENTER EC ON X-AXIS
001146 162702  000003                   SUB #3,R2
001152 010267  000206                   MOV R2,DRM1E
001156 005000                           CLR R0
001160 004567  176776                   JSR R5,TPLOT
001164 000200                           DRM1E: .WORD 7
001166 000061                           .WORD 505.
001170 012700  002147'                  MOV #ALPHA,R0
001174 004767  0000006                  JSR PC,PRINT
001200 016100  001474'                  MOV ECTAB(1),R0
001204 004567  0000006                  JSR R5,DECOUT
001210 000342                           .WORD 2
                                        ;
001212 026127  001614' 000660           DRM14: CMP YPT(1),#432.  ;DRAW IT
001220 100064                           BPL DRM16                ;Y-OVERFLOW
001222 016702  000436                   MOV DIVISN,R2
001226 016167  001544' 000052           MOV XPT(1),DRM1A         ;SET ARG, FOR X
001234 016767  000446  000056           MOV DRM1A,DRM1C
001242 066767  000422  000050           ADD #10H,DRM1C
001250 012767  000512  000032           MOV #512.,DRM1B          ;SET ARG, FOR Y
001256 012767  000512  000046           MOV #512.,DRM1D
001264 066167  001614' 000016           ADD YPT(1),DRM1B
001272 066167  001614' 000022           ADD YPT(1),DRM1D
001300 005000                           DRM15: CLR R0
001302 004567  176564                   JSR R5,TPLOT             ;POSITION VECTOR
001306 000002                           DRM1A: .WORD 2
```

```
172 001310 000000          DRM18:  .WORD 0
173 001312 005200                  INC R0
174 001314 004567  176552          JSR R5,TPLOT          ;DRAW LINE
175 001320 000000          DRM1C:  .WORD 0
176 001322 000000          DRM1D:  .WORD 0
177 001324 005261  001e14'         INC YPT(1)            ;BUMP Y-COORD.
178 001330 005267  177754          INC DRM1B
179 001334 005267  177762          INC DRM1D
180 001340 005302                  DEC R2                ;DEC ITER. CNT
181 001342 001356                  BNE DRM15             ;ANOTHER LINE
182                        ;
183 001344 005002                  CLR R0                ;RESET CURSOR
184 001346 004567  176520          JSR R5,TPLOT
185 001352 000000                  .WORD 0
186 001354 000362                  .WORD 242.
187 001356 012700  002147'         MOV #ALPHA,R0         ;SET ALPHA MODE
188 001362 004767  000000G         JSR PC,PPINT
189 001366 000167  000070          JMP DRM11A            ;EXIT
190                        ;
191 001372 026727  000266  000001  DRM16:  CMP DIVISR,#1  ;Y-OVERFLOW?
192 001400 001006                  BNE DRM17             ;NO
193 001402 012700  002072'         MOV #TXT,R0           ;YES-PRINT THAT
194 001406 004767  000000G         JSR PC,PPINT
195 001412 000167  000044          JMP DRM11B            ;EXIT
196                        ;
197 001416 006267  000242          DRM17:  ASR DIVISR    ;RESCALE
198 001422 005000                  CLR R0                ;FIRST DO Y-AXIS
199 001424 006360  001672'         DRM18:  ASL YTBL(0)
200 001430 005720                  TST (0)+
201 001432 020027  000010                  CMP R0,#8.
202 001436 001372                  BNE DRM18
203                        ;
204 001440 005000                  CLR R0                ;RESCALE CNT TABLE
205 001442 006260  001614'         DRM19:  ASR YPT(2)
206 001446 005720                  TST (0)+
207 001450 020067  000212                  CMP R0,ITEM
208 001454 001372                  BNE DRM19
209 001456 004767  177000          JSR PC,PLT1           ;DRAW RESCALED PICTURE
210 001462 012605          DRM110: MOV (SP)+,R5
211 001464 012602                  MOV (SP)+,R2
212 001466 012601                  MOV (SP)+,R1
213 001470 012600                  MOV (SP)+,R0
214 001472 000207                  RTS PC                ;RETURN FROM DRAW1
215                        ;
```

```
                                        .TITLE  DATA SECTION
                                        ;********************
                                        ;
                                        ;
001474                          ECTAB:  .BLKW 20.               ;ECT'S IN X SEQUENCE
001544                          XPT:    .BLKW 26.               ;ARG. X COORD.
001614                          YPT:    .BLKW 26.               ;Y-VALUE REL. TO X-AXIS
001644 000000                   DIVISR: .WORD 0                 ;# OF LINES PER FV
001666 000000                   ITEM:   .WORD 0                 ;#+2 OF CURRENT FV IN GRAPH
001670 000000                   WIDTH:  .WORD 0                 ;WIDTH OF D.S. CUR.
001672                          YTEXT:  .BLKW 5                 ;ARRAY FOR Y INITIAL, TOP DOT
001704                          SAVAR:  .BLKW 6                 ;RESERVED FOR MCR
001720 000000                   TPTMOD: .WORD 0         ;VAR. FOR TPLOT
001722 000000                   TPTX:   .WORD 0
001724 000000                   TPTY:   .WORD 0
                                ;
001726  033     014     200    SCREEN: .BYTE 33,14,200
001731  040     040     040    B63:    .ASCII /                                /
001734  040     040     040
001737  040     040     040
001742  040     040     040
001745  040     040     040
001750  040     040     040
001753  040     040     040
001756  040     040     040
001761  040     040     040
001764  040     040     040
001767  040     040     040    B33:    .ASCII /                        /
001772  040     040     040
001775  040     040     040
002000  040     040     040
002003  040     040     040
002006  040     040     040
002011  040     040     040
002013  040     040     040    B15:    .ASCII /     /
002016  040     040
002020  040     040     040    B8:     .ASCII /    /
002023  040
002024  040     040     040    B4:     .ASCII /    /<200>
002027  040     200
002031  040     055     012    TX1:    .ASCII/ -/<12><12><12><12><12><15>/    /<200>
002034  012     012     012
002037  012     015     040
002042  040     040     040
002045  200
002046  105     126     105    TX2:    .ASCIZ /EVENT OVERFLOW ON X/
002051  116     124     040
002054  117     126     105
002057  122     106     114
002062  117     127     040
002065  117     116     040
002070  130     000
002072  131     055     131    TX3:    .ASCIZ /Y-AXIS OVERFLOW/
002075  130     111     123
002100  040     117     126
002103  105     122     106
002106  114     117     127
```

```
            002111      000
     26     002112      040     105     126  XTEXT:  .ASCII / EVENT   /<200>
            002115      105     116     124
            002120      040     040     040
            002123      200
     27     002124      012     015     040  GAP:    .ASCII <12><15>/              /<200>
            002127      040     040     040
            002132      040     040     040
            002135      040     040     040
            002140      040     040     040
            002143      040     200
     28     002145      015     012          CRLF:   .BYTE 15,12
     29     002147      037     200          ALPHA:  .BYTE 37,200
     30                                               .EVEN
     31                                          ;
     32                                               .TITLE  LOGGRF
     33             000000'                           .END
```

```
LOGPRT : HIMOS OUTPUT INTERPRET RT-11 MACRO VM02-09    08:10:25 PAGE 1

                                        .TITLE LOGPRT : HIMOS OUTPUT INTERPRETER
                                    ;*****************************************************
                                    ;*
                                    ;* THIS PROGRAM PROCESSES "HIMOS" LOG FILES
                                    ;* THE FILES ARE TRANSFORMED INTO READABLE FORM AND
                                    ;* PRINTED ON THE SYSTEM TERMINAL, ONE RECORD PER LINE.
                                    ;* THE FIRST VALUE IS THE EC, THE REMAINING ITEMS ARE
                                    ;* EXPLAINED BY A LEADING 2 LETTER ID. CODE.
                                    ;*
                                    ;*****************************************************
                                    ;
                                        .MCALL .PRINT,.REGDEF,.EXIT,.READW,.WAIT
                                        .MCALL .CSIGEN,.CLOSE,.TTYOUT,.TTYIN,.EXIT
                                        .WAIT
                                        .REGDEF
                    177564              TTYOUT=177564
                    177564              TPS=TTYOUT
                    000066              LINES=54.
                    000000              .ASECT
                    001000              .=1000
001000  012706  001000          START:  MOV #.,SP
001004                                  .CLOSE #3
001016                                  .CSIGEN #DSKHND,#EXT,#0
001034  103761                          BCS START               ;OPEN ERROR,TRY AGAIN
001036  012705  000066                  MOV #LINES,R5           ;SET LINE CNT
001042  004767  000114                  JSR PC,GETLIN
001046  005002                          CLR R2                  ;BLOCK CNT
                                    ;
001050                              RDBLK:  .READW #AREA,#3,#BUFF,#256.,R2  ;READ ONE BLOCK
001114  103004                          BCC RDOK                ;READ OK
001116  005737  000052                  TST @#52                ;EOF?
001122  001411                          BEQ TERM                ;YES-TERMINATE
001124  000725                          BR START                ;NO-HARD ERROR
001126  012703  001000              RDOK:   MOV #512.,R3            ;512 BYTES
001132  012701  003144                  MOV #BUFF,R1            ;STARTING AT BUFF
001136  004767  000146                  JSR PC,PRINTB           ;PRINT AND DECODE
001142  005202                          INC R2                  ;READ NEXT BLOCK
001144  000741                          BR RDBLK
                                    ;
001146                              TERM:   .CLOSE #3
001160                                  .EXIT
                                    ;
                                    ;*****
001162                              GETLIN: ;READ HEADING LINE
                                    ;*****
001162  012702  000160                  MOV #LINBUF,R2
001166                              GETLI1: .TTYIN
001172  042700  000200                  BIC #200,R0
001176  110022                          MOVB R0,(2)+
001200  122700  000012                  CMPB #12,R0
001204  001370                          BNE GETLI1
001206  105012                          CLRB (2)
001210  012700  004156                  MOV #CRLF,R0
001214  004767  000002                  JSR PC,PRINT
001220  000207                          RTS PC                  ;RETURN FROM GETLIN
                                    ;*****
001222                              PRINT:  ;PRINT TEXT STARTING AT (R0)
```

```
                                ;*****
                                        MOV R1,-(SP)
                                        MOV R3,R1
                                PRINT0: MOVB (1)+,R0
                                        .TTYOUT
                                        TSTB @#TTYOUT
                                        BPL .-4
                                        TSTB R0
                                        BEQ PRINT1
                                        CMPB #200,R0
                                        BEQ PRINT2
                                        BR PRINT0
                                PRINT1: MOVB #15,R0
                                        .TTYOUT
                                        MOVB #12,R0
                                        .TTYOUT
                                PRINT2: TSTB @#TTYOUT
                                        BPL .-4
                                        MOV (SP)+,R1
                                        RTS PC                  ;RETURN FROM PRINT
                                ;*****
                                PRINT8: ;PRINT R3 CHARS, STARTING AT (R1)
                                ;*****
                                        MOV R0,-(SP)
                                        MOV R1,-(SP)
                                        MOV R3,-(SP)
                                        MOV R4,-(SP)
                                PRT81:  .TTYOUT (1)+
                                        MOVB (1),R4             ;ANALYSE CHAR
                                        BIC #177402,R4          ;CLEAR HIGH BYTE ;0
                                        SUB #44,R4
                                        BMI PRTB2               ;<0 OR LF
                                        BEQ PRTB2               ;=LF
                                        CMP R4,#14
                                        BPL PRTB2               ;>54
                                        ASL R4                  ;16 CODE+2
                                        MOV IDADD(4),R0         ;ADDR OF IDENTIFIER
                                        JSR PC,PRINT
                                        TSTB (1)+               ;SKIP PREVIOUS LF
                                        DEC R3
                                        BEQ PRTB3
                                PRTB2:  CMPB #12,(1)
                                        BNE PRTB2A
                                        DEC R5
                                        BNE PRTB2A
                                        MOV #HEAD,R0
                                        JSR PC,PRINT
                                        MOV #LINBUF,R0
                                        JSR PC,PRINT
                                        MOV #LINES,R5
                                PRTB2A: DEC R3
                                        BNE PRT81
                                        TSTB @#TPS
                                        BPL .-4
                                PRTB3:  MOV (SP)+,R4
                                        MOV (SP)+,R3
                                        MOV (SP)+,R1
```

```
                                        MOV (SP)+,R0
                                        RTS PC                  ;RETURN FROM PRINTR
                                ;
                                DSKHND: .BLKW 400.
                                EXT:    .RADS0 "LOG"
                                        .BLKW 3
                                AREA:   .BLKW 10
                                BUFF:   .BLKW 256.
                                HEAD:   .BYTE 15,12,12,12,12
                                        .BYTE 12,12,12,12,12
                                CRLF:   .WORD 0
                                LINBUF: .BLKW 80.
                                ;
                                ;       NAME            INDX
                                ;
                                IOADD:  IOER            ;0  *
                                        IOPC            ;2  PC
                                        IOPS            ;4  PS
                                        IOSP            ;6  SP
                                        IOCP            ;10 CP
                                        IOEM            ;12 EM
                                        IOSR            ;14 SR
                                        IOBF            ;16 BF
                                        IOR             ;20 R
                                        IOER            ;22 *
                                        IOMT            ;24 MT
                                        IOST            ;26 ST
                                ;
                                ;
                                IOER:   .BYTE 52,200    ;ERROR IN IO CODE
                                IOPC:   .ASCII /PC/<200>
                                IOPS:   .ASCII /PS/<200>
                                IOSP:   .ASCII /SP/<200>
                                IOCP:   .ASCII /CP/<200>
                                IOEM:   .ASCII /EM/<200>
                                IOSR:   .ASCII /SR/<200>
                                IOBF:   .ASCII /BF/<200>
                                IOR:    .ASCII /R/<200>
                                IOMT:   .ASCII /MT/<200>
                                IOST:   .ASCII /ST/<200>
                                        .END START
```

SOFTHOOK IDENTIFICATION EMITTER RT-11 MACRO VM02-09 00:07:19 PAGE 1

```
                                .TITLE SOFTHOOK IDENTIFICATION EMITTER
                        ;*******************************************************
                        ;*
                        ;* SUBROUTINE FOR FORTRAN EMIT CALLS
                        ;*
                        ;*******************************************************
                        ;
                                .MCALL .REGDEF
                                .REGDEF
000000'                         .CSECT EMIT
000004  005725            EMIT:  TST (5)+                ;SET R5 TO ID.CODE
000002  012701  000016'          MOV #16,R1             ;R1 = OP.CODE FOR TRAP
000006  117521  000000          MOVB @(5),(1)+         ;SET ID AS TRAP ARG.
000012  112711  000211          MOVB #211,(1)          ;CREATE OP.CODE FOR TRAP
000016  000000                  .WORD 0                ;GENERATE TRAP
000020  000207                  RTS PC                 ;RETURN FROM EMIT
000000'                         .END EMIT
```

MINOS INIT FOR EMIT UNDER FORTR RT-11 MACRO VM02-09 00:08:22 PAGE 1

```
                                .TITLE MINOS INIT FOR EMIT UNDER FORTRAN
                        ;*******************************************************
                        ;*
                        ;* THIS ROUTINE HAS TO BE CALLED PRIOR TO THE FIRST EMIT CALL
                        ;* IT RESETS THE TRAP VECTOR TO POINT TO SYSLOG
                        ;*
                        ;*******************************************************
                        ;
000000'                         .CSECT INEMIT
000000  012737  154002  000034  INEMIT: MOV #154002,@#34      ;RESET TRAP VECTO
000006  012737  000347  000036          MOV #347,@#36          ;TO POINT TO SYSLOG
000014  000207                  RTS X7
000000'                         .END INEMIT
```

5. Program Examples for Monitor Sessions

5.1 HISTO - Program for Compiler Analysis

```
RT-11 FORTRAN IV     V01B-08                    PAGE 001

                C PROGRAM TO PRINT A DISTRIBUTION HISTOGRAM
                C SAMPLE VALUES ARE READ FROM THE INPUT FILE FTN1.DAT
                C THE SORTED ARRAY IS STORED IN FTN2.DAT
     0001             LOGICAL*1 STAR(80)
     0002             REAL ARRAY(200)
     0003             INTEGER HIST(10)
     0004             DATA STAR/80*'*'/
     0005             DO 10 I=1,200
     0006          10 READ (1,20,END=130) ARRAY(I)
     0007          20 FORMAT (F5.2)
     0008             CONTINUE
     0009         100 ISIZE=I-1
     0010             DO 120 J=1,ISIZE-1
     0011             DO 110 K=J+1,ISIZE
     0012             IF (ARRAY(J).GE.ARRAY(K)) GO TO 110
     0013             TMP=ARRAY(J)
     0014             ARRAY(J)=ARRAY(K)
     0015             ARRAY(K)=TMP
     0016         110 CONTINUE
     0017         120 CONTINUE
     0018             DO 125 K=1,ISIZE
     0019             WRITE(2,23)ARRAY(K)
     0020         125 DO 126 K=1,10
     0021             HIST(K)=0
     0022         126 CONTINUE
     0023             DO 150 K=1,ISIZE
     0024             N=IFIX(ARRAY(K))/10+1
     0025         130 HIST(N)=HIST(N)+1
     0026             WRITE(7,135)
     0027         135 FORMAT (1X,'DISTRIBUTION IN INTERVALS OF 10,00:',/)
     0028             DO 150 K=1,10,1
     0029             J=K-10
     0030             WRITE (7,140)HIST(K/10),J,K
     0031         140 FORMAT (/,1X,I3,' VALUES, RANGE: ',I3,' TO ',I3,5)
     0032             IF (HIST(K/10).EQ.0) GO TO 150
     0033             WRITE(7,145) (STAR(I),I=1,HIST(K/10))
     0034         145 FORMAT (1H+,24A1)
     0035         150 CONTINUE
     0036             WRITE (7,160) ISIZE
     0037         160 FORMAT (//,1X,'TOTAL NUMBER OF SAMPLES= ',I3)
     0038             CALL EXIT
     0039             END
     0040
```

5.2 SAMGEN - Random Number Generator for FPP Performance Analysis

```
RT-11 FORTRAN IV    V010-00                    PAGE 001

      C PROGRAM TO GENERATE J RANDOM NUMBERS IN THE RANGE 0-99.99
      C FOUR EMIT CALLS ARE INSERTED FOR MONITORING "HAN" PERFORMANCE
0001        REAL MAX,MIN
0002        CALL INCMIT
0003        CALL EMIT(0),
0004        WRITE (7,20)
0005     20 FORMAT (1X,'# OF SAMPLES: ',$)
0006        READ (5,30)J
0007     30 FORMAT (I5)
0008        MIN=0
0009        MAX=99.99
0010        I=0
0011        N=0
0012        DO 100 K=1,J
0013        CALL EMIT(1)
0014        X=HAN(I,M)*(MAX-MIN)+MIN
0015        CALL EMIT(2)
0016        WRITE (1,60)X
0017        CALL EMIT(3)
0018     60 FORMAT (F5.2)
0019    100 CONTINUE
0020        CALL EMIT(0)
0021        CALL EXIT
0022        END
```

APPENDIX C

Hardware Documentation

C.1 Hardware Monitor Design

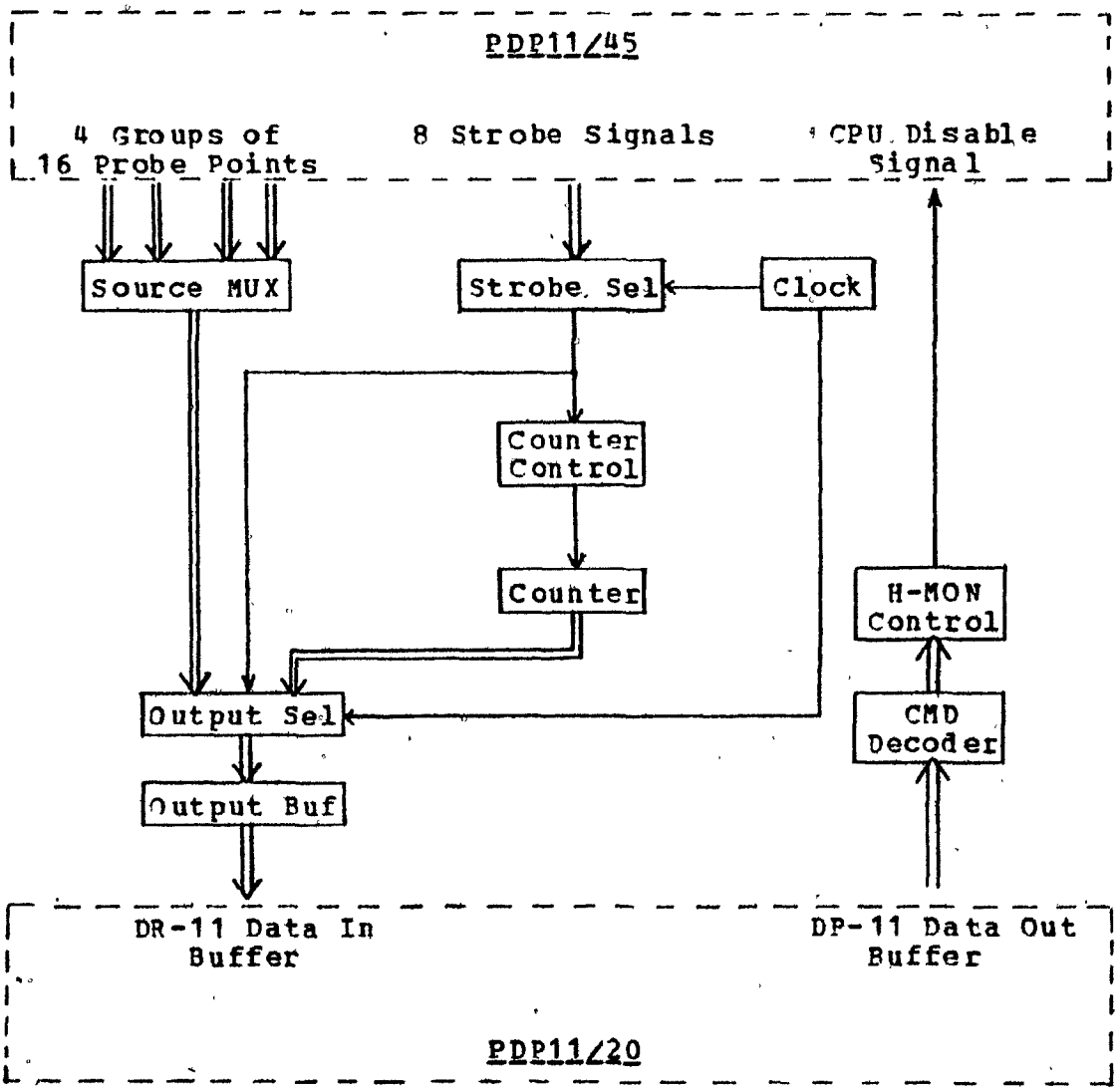

Figure C.1.1: H-MON, Data and Control

BIT

0 2	3 4	5 7	8 -	9 11	12 15
Mode SEL <0..2>	Source SEL <0..1>	CLK SEL <0..2>	STOP/START	STROBE SEL <0..2>	RESERVED
000 TRACE	00 SEL SOURCE 0	000 10 KHz	0 STOP	000 SEL STROBE 0	
000 COUNT	01 " " 1	001 5 KHz	1 START	001 " " 1	
010 EVENT DRIVEN	10 " " 2	010 2.5 KHz		010 " " 2	
011 TIME	01 " " 3	011 1.25 KHz		011 " " 3	
100 NOT VALID		100 500 Hz		100 " " 4	
101 NOT VALID		101 250 Hz		101 " " 5	
110 SAMPLE		110 125 Hz		110 " " 6	
111 FAST COUNT		111 62.5 Hz		111 " " 7	

Fig. C.1.2: H-MON Control Word, Bit Assignment.

C.2 Parallel communication Link

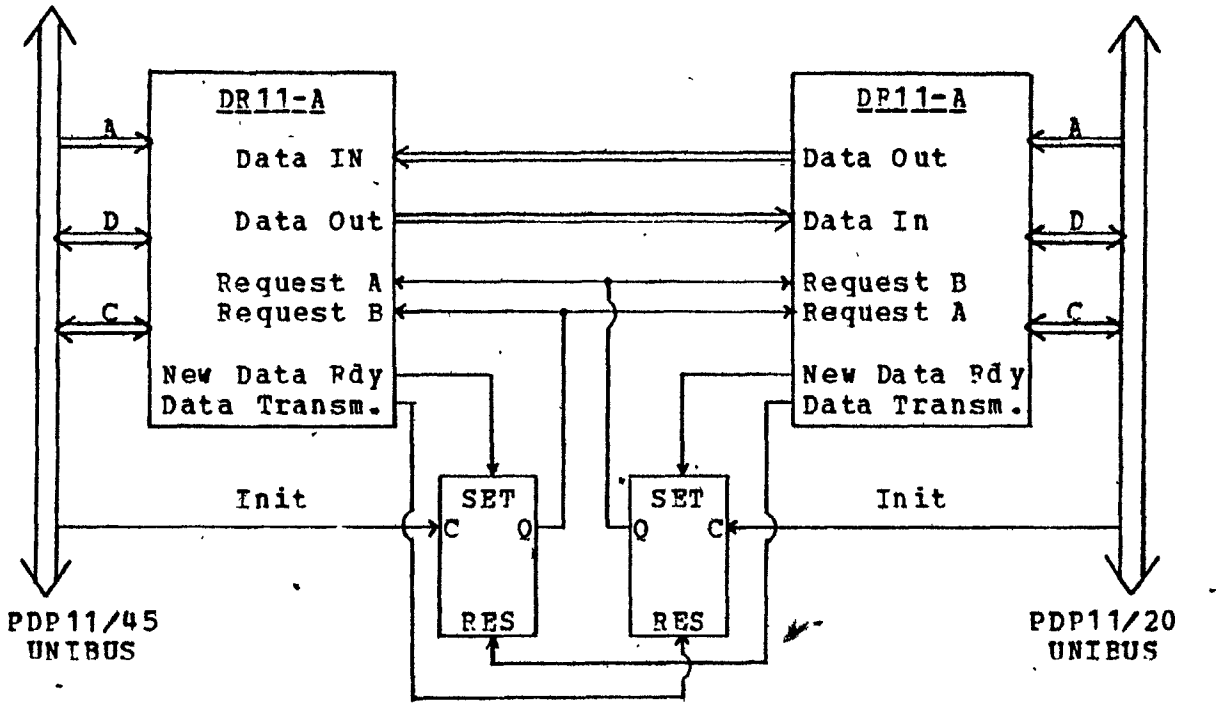

Figure C.2.1: Communication Link

REFERENCES

[1] FERRARI, D., "Workload Characterization and Selection in Computer Performance Measurement", Computer, Vol.5, July 1972.

[2] CALINGAERT, P., "System Performance Evaluation: Survey and Appraisal", Comm. of the ACM, Vol. 10, No. 1, January 1967.

[3] SCHATZOFF, M., TSAO, R., and WIIG, R.,"An Experimental Comparison of Time-Sharing and Batch Processing",Comm. of the ACM, Vol. 10, No. 5, May 1967.

[4] GREENBERGER, M., "The Priority Problem and Computer Time-Sharing", Management Science, Vol. 12, July 1966, pp. 888-906.

[5] GRAHAM, R.M., "Performance Prediction", Advanced Course on Software Engeneering, Springer-Verlag (1973),pp. 395-463.

[6] BLATNY, J., CLARK. S.R., and ROURKE, T.A., "On the Optimization of Performance of Time-Sharing Systems by Simulation", Comm. of the ACM, Vol.15, No.6,June 1972.

[7] KLEINROCK,L., "Time-Shared Systems: A Theoritical Treatment", Journal of the ACM, Vol.14, April 1967,pp. 242-261.

[8] HELLERMAN, H., and SMITH Jr. H.J.,"Throughput Analysis of Some Idealized Input, Output and Compute Overlap Configuration", ACM Computing Surveys, Vol.2, No. 2, June 1970, pp. 111-118.

[9] DENNING, P.J., "A Note of Paging Drum Efficiency", ACM Computing Surveys, Vol.4, No.1, March 1972, pp. 1-3.

[10] LOWE, T.C., "Analysis of Boolean Program Models for Time-Shared, Paged Environments", Comm. of the ACM, Vol.8, no.4, April 1965.

[11] RUSSEL, E.C., and ESTRIN, G., "Measurement Based Automatic Analysis of FORTRAN Programs", Proceedings of SJCC, 1969, pp. 723-752.

[12] EPSTEIN, A., Hardware Monitor and Interprocessor Communications Link, Project Report for 304-531B, McGill University, S.O.C.S., April 1976.

[13] FOLEY, J.D., "A Markovian Model of the University of Michigan Executive System", Comm. of the ACM, Vol.10, No.9, Sept. 1967.

[14] SEAMAN, P.H., and SOUCY, R.C., "Simulating Operating Systems", IBM Systems Journal, Vol.8, No.4, 1969, pp. 264-280.

[15] GOTLIEB, C.C., "Performance Measurement", Advanced Course on Software Engeneering, Springer-Verlag (1973), pp. 464-491.

[16] TOTARO, J.B., "Real Time Processing Power: A Standardized Evaluation", Computers and Automation, Vol.17, No.4, April 1967, pp. 16-19.

[17] BUCHHOLZ, W., "A Synthetic Job for System Measurements", IBM Systems Journal, Vol.8, No.4, 1969, pp. 264-280.

[18] SALTZER, J.H., and GINTELL, J.W., "The Instrumentation of Multics", Comm. of the ACM, Vol.13, No.8, Aug. 1970.

[19] CANTRELL, H.N. and ELLISON, A.L., "Multiprogramming System Performance Measurement and Analysis", Spring Joint Comp. Conf., 1968, pp. 213-221.

[20] CORBATO, F.J., "A New Remote-Accessed Man-Machine

System", Proc. of the AFIPS 1965 Fall Joint C. C., Vol.27, Part 1, pp. 185-247.

[21] ESTRIN, G., "SNUPER COMPUTER-A Computer in Instrumentation Automaton", Spring Joint Comp. Conf., 1967, pp. 645-656.

[22] FABRY, R.S., "Dynamic Verification of Operating System Decisions", Comm. of the ACM, Vol.16, No.11, Nov.1973.

[23] CURETON, H.O., "A Philosophy to System Measurement", Fall Joint Comp. Conf., 1972.

[24] MARIN, M.A., personal communication, McGill Univ., 1976

[25] DRUMMOND Jr., M.E., Evaluation and Measurement Techniques for Digital Computer Systems, Prentice-Hall Inc., Englewood Cliffs, N.J., 1973.

[26] ROSK, D.J. and EMMERSON, W.C., "A Hardware Instrumentation Approach to Evaluation of a Large Scale System", Proc. of the ACM, 24th Nat. Conf., 1969, pp. 351-368.

[27] SCHULMAN, F.D., "Hardware Measurement Device for IBM System/360 Time-Sharing Evaluation", Proc. of the ACM Nat. Meeting, 1967, pp. 103-109.

[28] EPSTEIN, A., A Survey of Hardware Monitors, Project Report, McGill University, S.O.C.S., 1976.

[29] NEMETH, A.G., and ROVNER, P.D., "User Program Measurement in a Time-Shared Environment", Comm. of the ACM, Vol.14, no.10, Oct. 1971.

[30] DSP-1 Digital Sampling Processor, Computer Performance Instrumentation, Kitchener, Ont., 1974.

[31] COCKRUM, J.S. and CROCKETT, E.D., "Interpreting the Results of a Hardware Systems Monitor", Spring Joint Comp. Conf., 1971, pp. 23-38.

[32] HIRSCH, A.R., Hardware Monitor Probe Points, Standard Oil Co., Indiana, 1972.

[33] NOE, J.D., "Acquiring and Using a Hardware Monitor", Datamation, April 1974, pp. 89-95.

[34] RUUD, R.J., "The CPM-X - A System Approach to Performance Measurement", Fall Joint Comp. Conf., 1972, pp. 949-957.

[35] HOARE, C.A.R., and LAUER, P.E., "Consistent and Complementary Formal Theories of the Semantics of Programming Languages", Acta Informatica, Vol.3, 1974,pp.135-153.

[36] RITCHIE, D.M., and THOMPSON, K., "The UNIX Time-Sharing System", Comm. of the ACM, Vol.17, No.7, July 1974.

[37] HANSEN, P.B., Operating System Principles, Prentice-Hall Inc., Englewood Cliffs, N.J., 1973.

[38] DEC, RT-11 System Reference Manual, Digital Equipment Corp., Maynard, Mass., DEC-11-ORUGN-C-D, 1975.

[39] DEC, Processor Handbook, PDP 11/20, Digital Equipment Corp. Maynard, Mass., 1972.

[40] DEC, Processor Handbook, PDP-11/45, Digital Equipment Corp., Maynard, Mass., 1973.

[41] DEC, PDP-11 Peripherals Handbook, Digital Equipment Corp. Maynard, Mass., 1975.

[42] DEC, RT-11 Software Support Manual, Digital Equipment Corp., Maynard, Mass., DEC-11-ORPGA-B-D, 1975.

[43] PINKERTON, T.B., "Performance Monitoring in Time-Sharing System", Comm. of the ACM, Vol.12, No.11, 1969.

[44] DENISTON,W.R., "SIPE: A TSS/360 Software Measurement Technique", Proc. of the ACM 24th Nat. Conf., 1969,

pp. 229-239.

[45] SEDGEWICK, R., STONE, R., and McDONALD, J.W., "SYP- A Program to Monitor OS/360", Fall Joint Comp. Conf., 1970, pp. 119-128.

[46] ASCHENBRENNER, R.A., AMIOT, L., and NATARAJAN, N.K., "The Neurotron Monitor System", Fall Joint Comp. Conf., 1971, pp.31-37.

[47] Comress, Dynaprobe 7900 System Specifications, Comten, Rockville, Maryland.

[48] MICRO-SUM, System 1000, General Information Manual, Tesdata Systems Corp., McLean, Virginia, 1973.

[49] CPI, Digital Sampling Processor DSP-1, Computer Performance Instrumentation, Kitchener, Ont., 1974.

[50] DEC, PDP-11/45 System Engeneering Drawings, Digital Equipment Corp., 1973.

[51] SLACMON, SLAC MVT Software Monitor, Stanford University, Stanford, Cal., Dec. 1970.

[52] DEC, RT-11/RSTS/E FORTRAN IV User's Guide, Digital Equipment Corp., DEC-11-LRRUA-A-D, 1975.

[53] DEC, PDP-11, FORTRAN, Language Reference Manual, Digital Equipment Corp., DEC-11-LFLRA-A-D, 1974.

[54] HOLT, R.C., "Some Deadlock Properties of Computer Systems", ACM Computing Surveys, Vol.4, No.3, Sept. 1972, pp.179-196.