A digital experiment monitor for the PDP-12 computer

by

John Joseph Jackley

A Thesis Submitted to the
Graduate Faculty in Partial Fulfillment of
The Requirements for the Degree of
MASTER OF SCIENCE

Major Subjects: Biomedical Engineering
Veterinary Physiology

Approved:

For the Graduate [illegible]

Iowa State University
Ames, Iowa

1971

TABLE OF CONTENTS

INTRODUCTION

Biological investigations offer many possibilities for the use of on-line computers in monitoring and control. Progress in an experiment is often interrupted by the need to stop and evaluate the data obtained from each phase of an experiment and may involve days of delay since the data must be tabulated by hand and evaluated either by hand or by a remote computer. This can result in the necessity to dismantle all or part of the experimental setup and it discourages experiments in which data must be collected continuously over a period of days or weeks. Furthermore, these difficulties can make certain types of experiments completely infeasible.

An example is provided by the methods attempted for continuous cell culture in a controlled physiological environment. A major difficulty in the use of in vitro culture chambers is the maintenance of normal values for the physiological parameters of the growth medium. One method, batch cultures, requires that the cells be transferred serially to fresh medium after several days of growth. Between transfers the medium becomes increasingly degraded since there are no control mechanisms present analogous to those that exist in vivo.

In 1962 a continuous culture apparatus was devised by Quinn (4) in which commercial analog control devices were used to control the feeding rate, pH, oxidation-reduction potential, stirring rate, salinity and volume of the culture. However, automatic recording of the parameter changes occurring during the culture growth was not available with this

apparatus. Improvements in the system, referred to as the Ecoanalyzer, were described by Zimmerli (10) and included a cell counter for determining the population density and a digital voltmeter and card punch for recording the chemical and physical properties of the culture. A pilot plant, the Trophocell, for the culture of lymphocytoid cells has been described by Vosseller and Moore (6). It is a large (1000 liters) apparatus, in which four environmental parameters are controlled and five variables are monitored by the instrumentation.

An Ecoanalyzer, to be monitored and controlled by an on-line digital computer, has been proposed for the study of the primary immune response of lymphocytes. Normal growth of the lymphocytes over a period of time sufficient for studying the immune response requires that the physiological parameters of the growth medium be maintained within relatively narrow limits. The present design specifies that at least eight experimental parameters are to be controlled. Laboratory data have provided initial values for the parameter limits; however, these values will require modification during the course of experimentation. In addition, there will be interaction between at least some of the parameters. Therefore, for continuous cell growth a relatively sophisticated control mechanism which includes the controller, the transducers and the actuators will be required. An on-line digital computer, the PDP-12 manufactured by Digital Equipment Corporation, has been selected as the controller for the Ecoanalyzer.

As illustrated by the Ecoanalyzer, two functions are to be provided by the instrumentation: data acquisition and parameter control. Automatic

data acquisition alone is sufficient for many biological experiments; however, control without human intervention is necessary if the experiment involves many varying parameters to which manual response would be inadequate. Analog computers have been used to implement data acquisition and control, but direct digital control is increasingly being used because of its advantages which include:

1. Representation of values by more significant digits since an analog computer is limited to two or three significant digits while a digital computer, by multiple precision arithmetic, can provide as many as required

2. Implementation by numerical methods of mathematical functions which are physically unrealizable, e.g. cannot be implemented by an analog device

3. Easier implementation and modification of complex logical and mathematical algorithms via the computer program

4. Measurement and analysis of data by statistical and numerical methods while the experiment is in progress, thus eliminating the need for intermediate storage of data in analog form

5. Immediate presentation of data and results of analysis in alphanumeric form

6. Selection and logging by program control of only those values that are significant, thus reducing the quantity of data which must be saved

7. Lower cost for a digital "minicomputer" than for an analog computer with comparable capabilities.

When the controlled process or the control strategy cannot be described by a set of equations, the capability of a digital computer to selectively store past information is particularly important. As the controller, the computer can be programmed to search both for the "solution" which has been defined as the object of the experiment and for an optimal control strategy and adapt it to a changing environment. To accomplish the search and adaptation such methods as optimum seeking techniques, dynamic programming and mathematical filters can be incorporated into the control program.

Although the computer can be thought of as the controller, it is the computer program and the control scheme or algorithm contained within the program that determines the performance and effectiveness of the computer as a controller. The subject of this thesis is the design and implementation of a computer program for on-line, real-time monitoring and control of a biological experiment using the PDP-12 computer. A model for determining the requirements of the program is provided by the Ecoanalyzer, but the design is not based on any particular control algorithm. Rather, an objective of the design is to provide a program structure which can be modified to accommodate changes (such as with the control algorithm) in a given experiment, and to accommodate experiments with other types of biological systems.

ANALYSIS

Statement of the Problem

Major computer tasks - acquiring and analyzing data, and providing command signals for the peripheral hardware - are to be combined in a program such that the computer becomes an element in a feedback loop of the entire system composed of the experiment, interfacing hardware and computer. Thus, for the initial continuous cell culture experiments, the computer is to maintain environmental parameters at specified levels by obtaining data samples from the culture chamber and initiating corrective action if the parameters deviate from their preset levels.

Program design is determined by the capabilities and limitations of both the computer and the peripheral hardware as well as by the requirements of the experiment. Since the peripheral hardware and the experiment will require modification as experience is gained, the program should be designed so that it can be easily adapted to these modifications. From a practical standpoint, the program must reflect a compromise determined by the speed of the central processor, the size of the memory, the limitations of the peripheral hardware, and the generality of the design. Limitations on program efficiency are imposed primarily by the size of the memory and by the response times of the peripheral hardware, input-output (I/O) devices and tape units. Within these limitations, the program is intended to be as general as is practical.

Successful control of the cell culture requires investigation of various sampling sequences and control algorithms. More effective use of

the computer results if the program design establishes rules or conventions for making these changes.

Computer Capabilities

Digital Equipment Corporation's PDP-12 has a 4096 word, 12 bit, random access memory and can be programmed with both LINC and PDP-8 symbolic assembly languages using the DIAL editor and assembler (3). Both languages may be used in the same program. In LINC mode the memory is organized as four, 1024 word segments and, of these four, a single memory addressing instruction can access two which are referred to as the Instruction Field and the Data Field. These two fields can be assigned to any of the four segments by the field setting instructions. In PDP-8 mode the memory is organized as thirty two, 128 word pages. By indirect addressing, a memory addressing instruction can access any word in the memory.

Communication between the central processing unit (CPU) and the operator is provided by a Teletype[1], CRT display screen and a variety of console switches. Two tape units provide auxiliary storage. Eight A/D convertors, six relays and an I/O bus line provide the interface between the CPU and the peripheral hardware. Additional features of the CPU include an external program interrupt and an illegal instruction trap. A 12 bit digital clock, scaled in tenths of seconds, has been attached via the I/O bus and it will trigger an external program interrupt when its register contains

---

[1]Trademark of the Teletype Corporation, 5555 Touhy Avenue, Skokie, Illinois.

the number "2260 octal". The clock register is read and reset by the CPU via program instructions, but it cannot be reset to any value other than zero.

Hardware Limitations

As opposed to I/O devices which provide communication between the computer operator and the CPU, I/O data and control terminals that interface the experiment to the computer are referred to as "peripheral hardware" or "peripheral data terminals". Hardware requirements will depend on the type of experiment being performed. Also, for a particular experiment changes in the design of the peripheral hardware may be required. Therefore, an objective of the program design is to allow for future hardware requirements and to avoid placing any restrictions on the type of hardware used.

Characteristics of the terminals are illustrated by the apparatus of the cell culture system; its actuators are two-state (on-off) devices and its sensors transmit a range of voltage values to the A/D convertors. Particular devices include the temperature sensing thermistor, ion specific electrodes, magnetic stirring assembly, titrators and the heater. Response times vary from nearly instantaneous for the temperature readings to many seconds for the ion concentration readings. A common amplifier is used for all of the electrodes and, as a result, a "settling period" is required after an electrode is switched to the amplifier before a reading can be taken.

Program Requirements

Logical structure

The real-time nature of the problem requires that the program have both a data dependent and a time dependent logical structure. Since the experiment, a biological system, cannot usually be fully or even partially described by a set of deterministic functions, the exact sequence of computer processing cannot be programmed. Rather, the computer must be able to respond to both the acquired data and to the time. Therefore, the program must be constructed from a set of rules which, based on the data and the time, govern the processing sequence of the computer.

For instance, when a physical parameter such as the temperature is to be kept above a given level by the computer, the decision to turn the heater "on" depends on the temperature value sampled by the computer. If the "on" time for the heater cannot be determined via a formula or a table of values, the correction can be made iteratively by alternately pulsing the heater and testing the temperature and therefore, the program sequence is determined by successive data samples. However, each pulse may introduce temperature transients and a time lag may be required before a valid sample value, a steady state value, can be obtained. In effect, unless these transients can be determined and the steady state value predicted, the response time of the computer controller must be slowed to match the reaction rates of the experimental process.

Scanning cycles

To maintain parameters at specified levels, each must be scanned periodically and corrected if it is outside an acceptable range of values; thus, requiring for each parameter a record kept of the time elapsed since it was last scanned. When the scanning cycle time for a parameter has been reached, the processing sequence (also referred to as a processing task) for that parameter should be initiated.

Parameter priorities and processing delays

It is desirable to assign relative priorities to the various parameters such that, if two or more are ready for processing at the same time, the most important one is processed first. A further refinement is necessary to allow a partially processed parameter to be interrupted in favor of a higher priority parameter. If a processing sequence requires a time delay because of physical or chemical characteristics of the parameters, performance of the system will be improved if processing of a lower priority parameter is initiated during the delay; and when the delay is complete, the higher priority processing should be restarted. Figure 1 illustrates the decisions involved in using priorities, delays and scanning cycle times.

Time management

To implement scanning cycles and time delays, the hardware clock must be read since it cannot be set to initiate an external interrupt at a specified time. Thus, a clock routine is required for reading and resetting the clock to zero and for updating necessary time records. If the

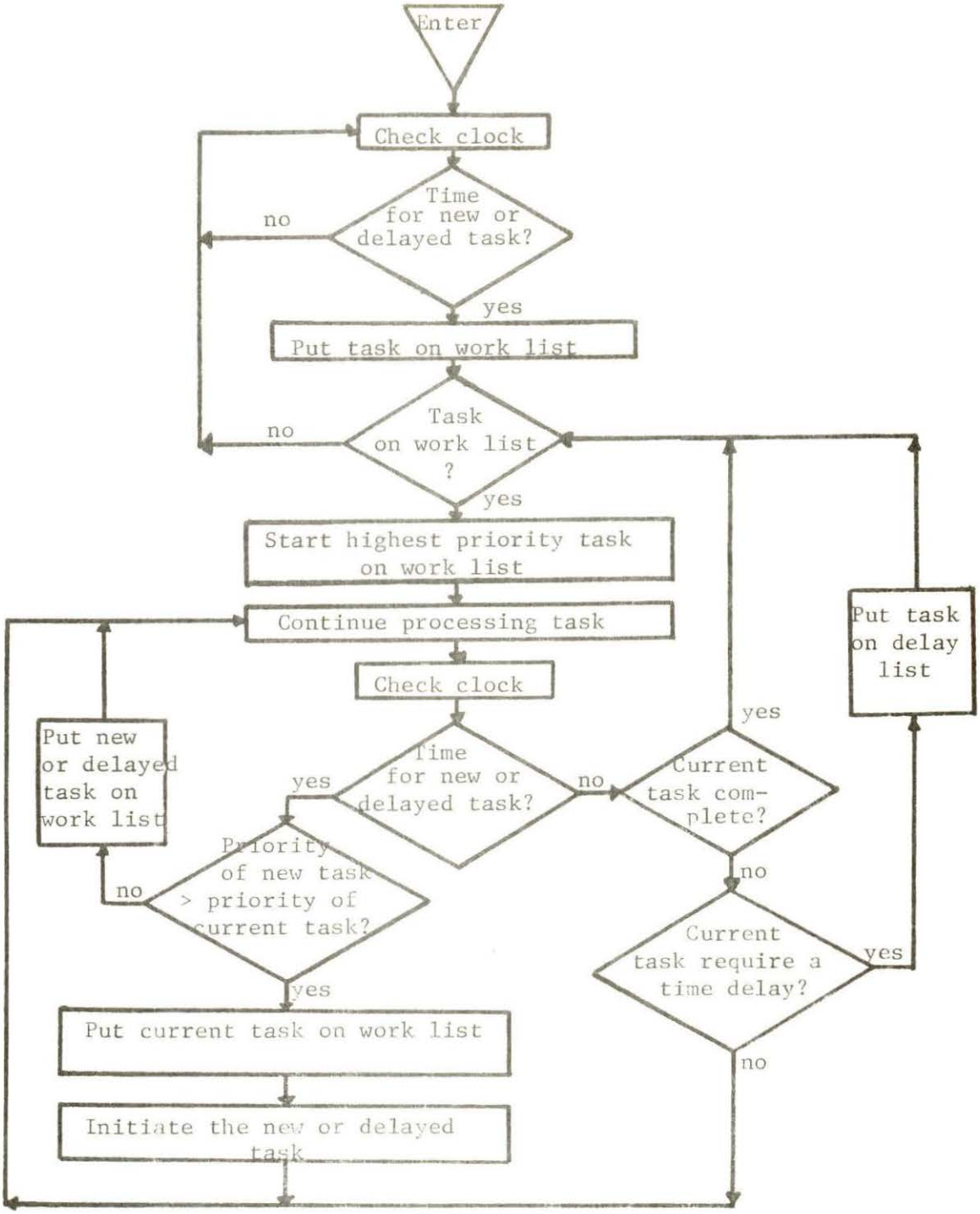


Figure 1. Task scheduling decisions

clock routine is designed so that it can be called by any routine in the program, it can then be used by the main program for scheduling and for servicing clock interrupts; and it can also be called upon by subroutines added to the program to furnish them with an updated record of the time. It is useful to use the clock to insert short pauses of specified lengths into a processing sequence without that sequence being interrupted. As illustrated later, a pause may be necessary if both processing tasks require the use of the same peripheral hardware device.

Parameter processing tasks

A method is required for specifying the parameter processing and control tasks and incorporating them into the program such that they can be contained within the memory space available and easily modified. Use of the tape units to store subroutines can help solve the space problem and will allow the size and capabilities of the processing programs to be expanded beyond the limits imposed by the main memory. Implementing the parameter priority interrupt requires that clock checks must be inserted in the parameter processing sequences and also that information necessary for continuing the interrupted sequence not be destroyed while the sequence is delayed. To satisfy these requirements a monitor program is required which will establish rules for programming the processing programs and which will establish the conventions for calling subroutines. Clock checks can be performed by the monitor, thus relieving the processing programs of this requirement.

CPU scheduling

The programming problem can be described as one of "scheduling" the work of the computer according to a set of priorities such that the basic time requirements of the experiment are satisfied. An objective of the scheduling rules should be to prevent any single computer task from excessively slowing the operation of the entire system. Such a situation can arise when a parameter processing sequence initiates a message, requires a data file from the magnetic tape, or must wait for an experimental parameter to come to equilibrium. In general, it can occur with any data transfer between the CPU and an external device because of the latter's slow response during which the CPU would be idle. Response times can vary from fractions of a second to minutes, such as with the ion electrodes, as compared with the few microseconds required by the CPU to execute one instruction.

An illustration is provided by the Ecoanalyzer, which at present uses one meter as an amplifier for six ion specific electrodes. Each electrode may require up to one minute of "settling" time after it is switched into the meter and before a sample value can be obtained. To improve the efficiency of the system, the computer should be scheduled for some other function during this idle period. However, the other function cannot be one of the other ion measurements since they all use a common device, the meter.

Thus, a basic objective of the program design is to interleave the time lags inherent in the peripheral devices and experimental process. For instance, during a parameter processing cycle, data is generated that may need to be stored on tape or logged on the Teletype. If these

operations can be queued, they can then be completed as time allows during the idle periods. Or, an alternate, but more complex solution is to make use of the tape and Teletype interrupts so that these operations can be executed in parallel with the CPU's execution of the parameter processing tasks.

Input/output operations

In general, input/output operations refer to all information transfers between the memory and peripheral devices. Characteristics and requirements of most of these operations will specifically depend on the experiment being performed and on the type of data and message terminals being used. However, messages from the computer concerning data, conditions of the experiment, and operation of the program will be required for most experiments. Additional considerations in programming for I/O operations are memory space requirements, the real-time required for performing the operations and the time and difficulty involved in adding or changing specific I/O requests in the program.

A message routine which establishes uniform rules for writing messages and which can be used to implement all message requests greatly simplifies the mechanics of inserting messages into the program. Since the computer has both a CRT display and a Teletype, a message request should have the option of specifying which output device is to be used. As mentioned previously, system efficiency requires that the message processor queue messages for output on a time available basis. However, for testing and debugging the program, trace or diagnostice messages are needed at the

point in time at which they are requested in the program. Therefore, the routine should provide the option for specifying either immediate or queued output of a message. Also, it should establish rules for specifying the message format and the message variables thus conserving memory space since a single format can then be used for two or more different messages, each with its own unique list of variables.

In some experiments it may be necessary for the experimenter to make modifications of the program instructions and tables or to enter data while the system is in operation. Making entries from the console in response to a request displayed on the screen is the simplest way to implement this function. An alternate method is to halt the CPU and toggle the information into the specific memory locations provided for it; however, to avoid disrupting the normal operation of the program, this must be done while the computer is "idle" and requires a signal from the program indicating that it is safe to stop the CPU. A more elaborate solution involves using the Teletype keyboard thus requiring a specific routine to handle the keyboard interrupt.

Interrupts and traps

External program interrupts, such as the keyboard interrupt, and instruction traps can occur either by design or as the result of an error. A general interrupt and trap processor can be used to provide the register saving and restoring capability and to disable and enable the interrupt mechanism. If two or more interrupts can occur simultaneously, the interrupt processor must determine, on the basis of priorities, which to

service first.

A specific routine is required to service a particular interrupt (or trap) and, if the routine is used elsewhere such that it can be interrupted (or trapped), it must be designed as a reentrant routine. Types of interrupts that need to be considered are those caused by the clock, Teletype, tape units and power failure. Design of the general interrupt processor should allow for adding or removing specific interrupts as required.

Data files

Because structures of data records and files are specific to the particular application and because of core limitations, a general file structure is not defined. However, provision should be made for incorporating data files into the program, and memory space should be set aside for data records. In particular, space is necessary for "volatile" files which must be accessed quickly and easily with little or no searching. These include the records of the recent sample data and control values which should be readily available for use by the parameter processing and control routines.

At the other extreme are records that are to be filed on tape and processed off-line. Space is necessary in the memory for a buffer area in which records can be assembled as a block for tape transfer. Providing two buffers, which are filled alternately, eliminates the need for the computer to wait until the transfer is complete before storing more records in the buffer area.

Between these two extremes are the records which must be filed in buffer areas or on tape and later accessed on-line during the experiment. For such records, the format, contents and the "key" will depend upon the intended use of the record and the method of search or addressing used to locate it.

Program modules

Experience may indicate need for change in the program. Examples include changing the time base of the clock, adding or deleting external program interrupts and changing peripheral hardware devices. With respect to the monitor and its processing and control tasks, modifications may vary in complexity from simple changing table entries to changing the logic and mathematics of the control scheme. A recommended procedure is to design the program as a set of "modules" such that the changes can be restricted to the module or modules which perform the function to be modified.

PROGRAM DESIGN

Block Organization and Sequence of Operation

The program is designed as a set of independent routines which provide the following general capabilities: task definition, scheduling and execution; time management; output processing; and interrupt and trap processing. On the basis of their functions, individual routines are organized into program blocks and, with the exception of the Supervisor Block (SVB), primary program control is passed to these blocks through only a few entry points as shown in Figures 2 and 3. Since they are intended to provide general capabilities, such as timing and message output, the Supervisor Block and the Message Processing Block may be called by any other routine in the program.

Data acquisition and parameter control functions for the experiment are organized as a set of one or more tasks and each task consists of one or more steps as illustrated in Figure 4. Task organization is provided by a Monitor Control Block (MCB) which functions as a table processor. For each task the sequence of steps is defined by coded entries in the Monitor Control Table (MCT) illustrated in Figure 6. Each entry's position in the MCT is identified by its horizontal coordinate, the Step Identification Number (STEPID), and by its vertical coordinate, the Parameter Identification Number (PARMID). Concatenated together, the PARMID and the STEPID form the entry's MCT Position Identifier (POSID).

Corresponding to each step is an independent service routine, identified by the STEPID, which implements the execution of that step. To

conserve memory, each service routine is intended to provide a common function for all tasks that require it by means of information in its own tables, which are indexed by the PARMID. Individual service routines can be added, deleted, or modified as processing requirements change and, to facilitate this, are organized in a group as the Monitor Service Routine Block (MSB).

Table 1. Abbreviations of program components' names

CCW:	Core Contents Word
ITTR:	Interrupt Time Test Routine
MCB:	Monitor Control Block
MCL:	Monitor Control Loop
MCT:	Monitor Control Table
MQ:	Message Queue
MQL:	Message Queue Loader
MQP:	Message Queue Processor
MSB:	Monitor Service Routine Block
PARMID:	Parameter Identification Number
POSID:	Monitor Control Table Position Identifier
PQ:	Priority Queue
SCB:	Scheduler Block
SRAT:	Service Routine Address Table
STU:	Scheduling Time Unit
SVB:	Supervisor Block
STEPID:	Step Identification Number
TASKID:	Task Identification Number
TDT:	Task Definition Table

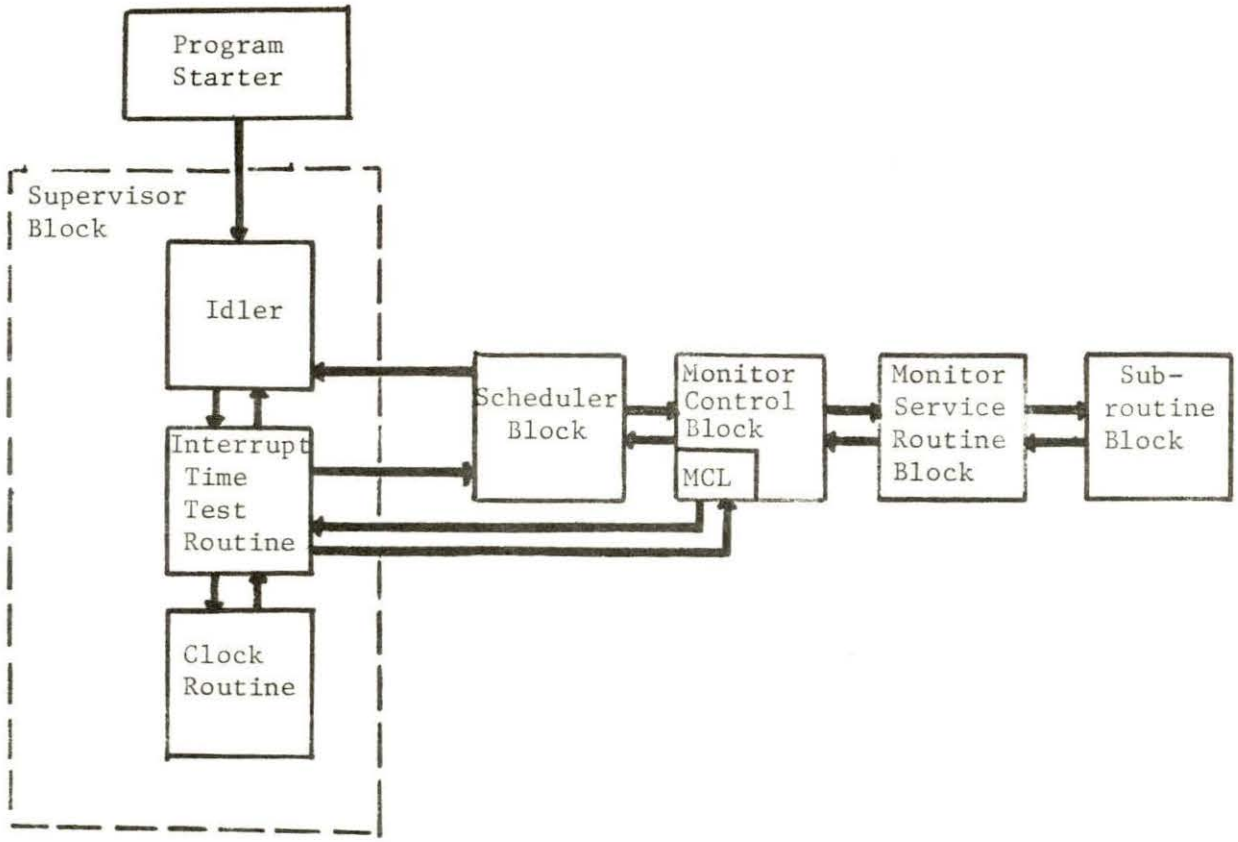


Figure 2. Basic block linkage for task scheduling and processing

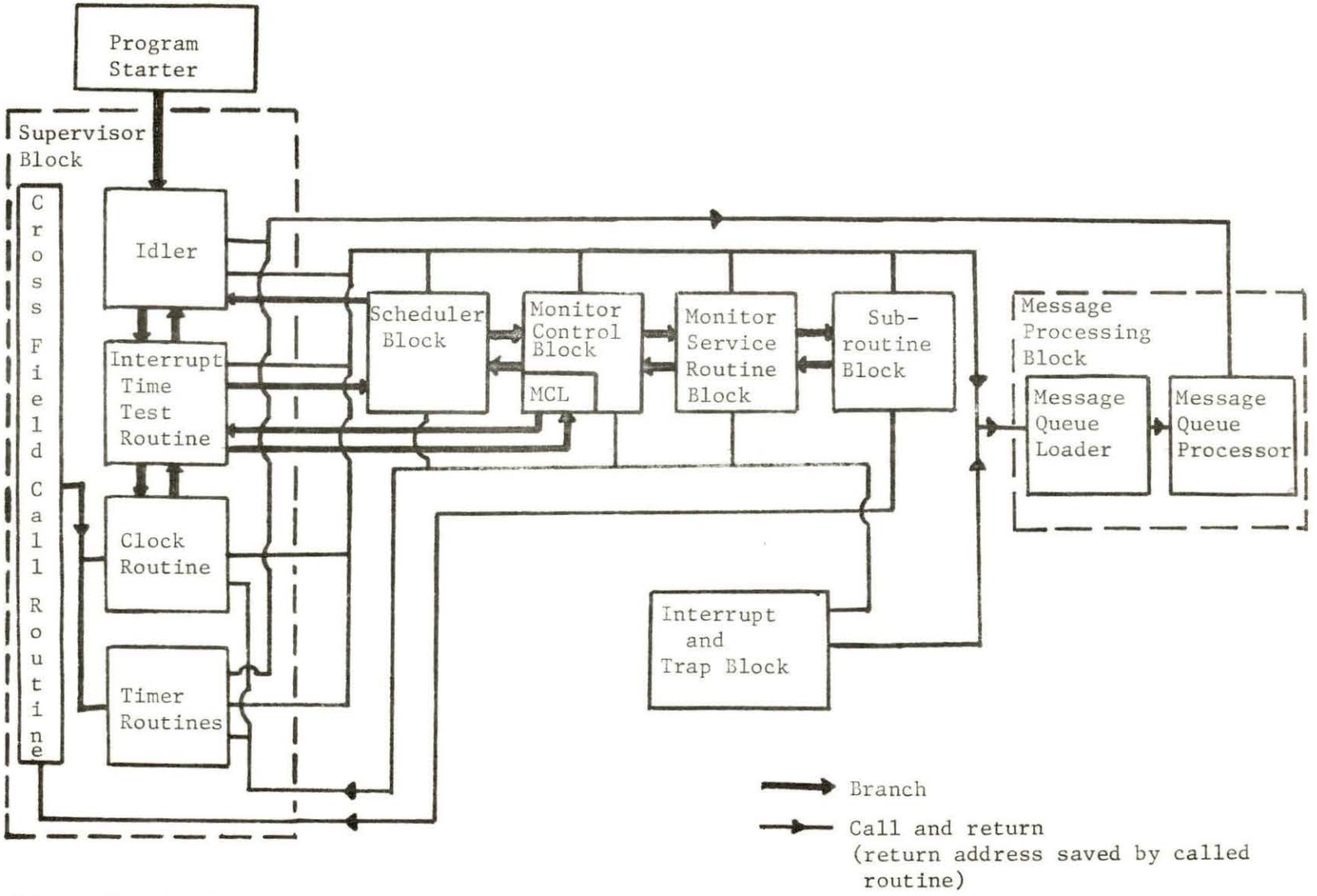


Figure 3. Complete block linkage

Task 1: Maintain parameter 1 within two limit values (set points)

step 1: Initialize for this task

step 2: Obtain parameter sample

step 3: File parameter sample

step 4: Is the parameter sample within the limits?

If "yes", go to step 9
If "no", go to step 5

step 5: Compute parameter correction

step 6: Send command signal to the peripheral hardware

step 7: Delay processing of task 1

step 8: Return to step 2

step 9: Return to scheduler (end of task 1)

Figure 4. Example of a task's sequence of steps

One service routine serves to load subroutines from the tape into the memory and each subroutine performs a single step and is identified by its STEPID. Thus, subroutines can be used to provide computations, data file operations or for any function not included as a specific service routine. In general, a routine which performs a particular step can be programmed either as a service routine or as a stored subroutine thus providing flexibility in adapting the program to a particular experiment.

Task execution is initiated by a set of scheduling routines which comprise the Scheduler Block (SCB) and are referred to collectively as the scheduler. The sequence in which tasks are initiated is determined

dynamically by task cycle times and by task priorities. These are defined by the experimenter by entries in the scheduler's Task Definition Table and by assigned Task Identification Numbers (TASKID). A Priority Queue (PQ) of tasks which are ready for processing is maintained by the scheduler and from this queue it selects the highest priority task and passes it to the MCB.

Records of the scanning cycle times for tasks to be initiated on a definite cycle, and records of the delay times for tasks which must be delayed are maintained by the scheduler. From these two sets of tasks come the entries for the Priority Queue as illustrated in Figure 5. Delay and scanning times are managed in conjunction with the program's Clock Routine which also keeps a record of the time since the scheduler's lists were last processed. These times are measured in terms of a Scheduling Time Unit (STU) defined by the experimenter as a multiple of the time unit of the hardware clock.

At present, all of the program tables and most of the flags are initialized at the time of assembly by the DIAL assembler using the LINC assembly language. After subroutines are assembled and loaded into the selected tape blocks on tape zero and the program is loaded into the memory (all done using the DIAL system), program execution begins in memory segment 2 with a simple Start Routine which initializes the hardware clock register and some flags. After turning the clock "on" the experimenter, via a sense switch, allows program execution to proceed to the Idler routine of the Supervisor Block as illustrated in Figure 2. Now the

progress execution is controlled by the clock via the Interrupt Time Test Routine (ITTR) which is called by the Idler.

When a Scheduling Time Unit has elasped, control is passed from the ITTR to the Scheduler Block (Figure 2) which processes its lists, looking for any task whose cycle time has elapsed. If none is found, control is returned to the Idler; otherwise, the highest priority task which is ready to be processed is passed to the Monitor Control Block which then processes the task using entries in the MCT. What is actually transferred to the MCB is the starting location of that task in the MCT. Using this POSID, the MCB selects the proper service routine for the first step and passes control to it (Figure 2); if a stored subroutine is required, it is loaded (if it is not already in the memory) and control is passed to the subroutine. After the step is completed control is returned to the MCB which has direct access to the ITTR without going through the scheduler. As described above, a time check is performed; and if one or more Scheduling Time Units have elapsed, control is passed to the scheduler. Otherwise, control is returned to the MCB and processing of the current task is resumed.

If the scheduler receives control, its lists are checked again and if another task is ready for execution, the priority of that task is compared with the priority of the task currently being executed. The lower priority task is returned to the Priority Queue and the higher priority task is passed to the MCB. Therefore, a partially processed task can be interrupted in favor of a higher priority task as mentioned previously in the section concerning program requirements. A task is processed step by

step until it is either finished, interrupted or requires a time delay. In the latter two cases, a POSID indicating the next step at which to continue the task, is returned to and saved by the scheduler. Task processing continues until there are no more tasks ready for processing and then control is returned to the Idler.

Memory Allocation

Each block is assigned to a particular memory segment and blocks may not cross memory segment boundaries. Areas of memory are also allocated for tables, for data and temporary record storage and for subroutines stored on tape. The Monitor Control Block, Monitor Service Routine Block, Scheduler Block, Supervisor Block and Interrupt and Trap Block are assigned to segment zero. The Message Processing Block and program tables are assigned to segment one. Segment two is reserved for subroutines and segment three is reserved for temporary record storage. Half of the latter is presently used for a pair of 256 word data buffers into which records are assembled for storage on tape while the other half is unused.

Supervisor Block

Overall control of the program is provided by the Supervisor Block since program execution begins here and returns to this block when no parameter processing tasks are scheduled. In effect, the SVB, via its supervisory routines, serves as a master scheduler for the entire program. As shown by Figure 3, the Timer Routines and the Clock Routine can be called by any other routine and provide general service functions for the

program. Other routines, intended to provide comparable service functions, can be added to the SVB since each routine in this block has its own unique "call list" of arguments as opposed to the routines of the MSB which must conform to the structure established by the MCB.

Idler Routine

When all scheduled parameter processing tasks have been completed, control of the program returns to the Idler and it maintains control until sufficient time has elapsed that another task is scheduled. During this idle period, the clock is "watched" via the Interrupt Time Test Routine and messages that have been stacked, e.g. a "background" job, are processed. This routine is a simple loop into which the scheduling of other background jobs (such as file sorting) can be inserted.

Clock Routine

Reading and resetting the clock is performed by the Clock Routine which also uses the "read" value to update the Total Experiment Time and the Elapsed Time Registers. Scale factors for these registers may be set by the experimenter and the scale factor of the Elapsed Time Register is the experiment's STU as used by the scheduler.

Interrupt Time Test Routine

The Elapsed Time Register is tested by the Interrupt Time Test Routine. If its value is not zero, indicating that the scheduler's lists should be processed again, control is passed to the scheduler. Otherwise, control is returned to the calling routine which is primarily either the Idler or the Monitor Control Loop (MCL) as shown in Figure 2.

Timer Set Routine

In conjunction with the Timer Test and the Clock Routines, the Timer Set Routine implements a time delay without rescheduling such as is provided by the Monitor Control Loop's pause option. (The MCL calls the Timer Routines to implement the pause, Figure 7.) Via the accumulator the delay interval, measured in the units of the hardware clock, is passed to this routine which uses it to set the Test Registers of the Timer Test Routine. Control is returned to the calling routine.

Timer Test Routine

After the Test Registers are set, the Timer Test Routine is called upon to perform the comparison with the Total Experiment Time Registers of the Clock Routine. Separation of the Timer Routines allows the calling routine to do some operations between the setting and testing of the Test Registers; in addition, the Timer Test Routine, between time tests, will initiate the processing of messages that have been queued for output on a time available basis. If a finer degree of time control is desired, the calling routine can turn this message option "off"; however, the option is always returned to its default condition, "on", upon return to the calling routine which occurs when the time interval set by the Timer Set Routine has elapsed.

Cross Field Call Routine

Because a call for a routine is this block may be from across a memory segment boundary, the identification of the calling segment must be saved (LINC mode only). The Cross Field Call Routine provides a

common "field" (segment) register saving and restoring capability for all of the routines in this block such that all calls from across a segment boundary must go through this routine.

Scheduler Block

Scheduling is implemented by processing the information lists shown in Figure 5. Information that must be defined by the experimenter for each task includes its cycle time (entered into the Task Cycle Time Table) and its Task Level Code, Task Interrupt Flag, and starting POSID (all entered in Task Code Table). The Task Cycle Time Table and Task Code Table comprise the Task Definition Table which is indexed, as are the Task Cycle Time List and the Priority Queue, by the TASKID. Priorities are assigned to the tasks according to their TASKID in inverse order.

Tasks for which the MCB initiates a delay are placed on the Task Delay List which requires three items of information: the delay time, the TASKID and the POSID at which the task is to be resumed. These are obtained from the program registers shown in Figure 5, and when the scheduler selects the next task from the Priority Queue, the TASKID of that task is entered into the TASKID Register and the contents of the Priority Queue entry for that task is transferred to the POSID Register. Note that the Priority Queue receives its entries from the Task Definition Table, the Task Delay List, or the TASKID Register.

Because the Priority Queue is indexed by the TASKID, it may contain only one entry for each defined task, but more than one entry at a time

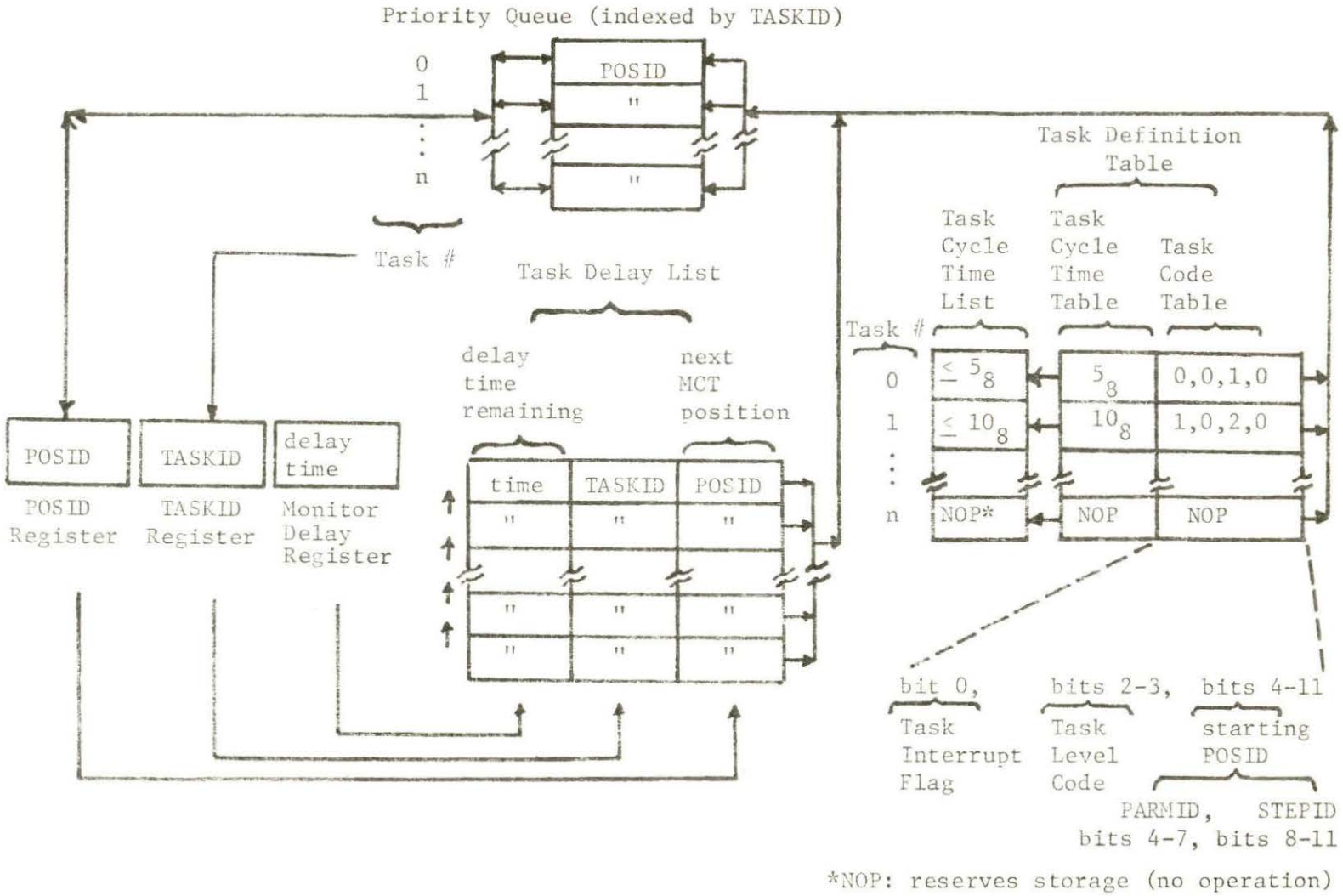


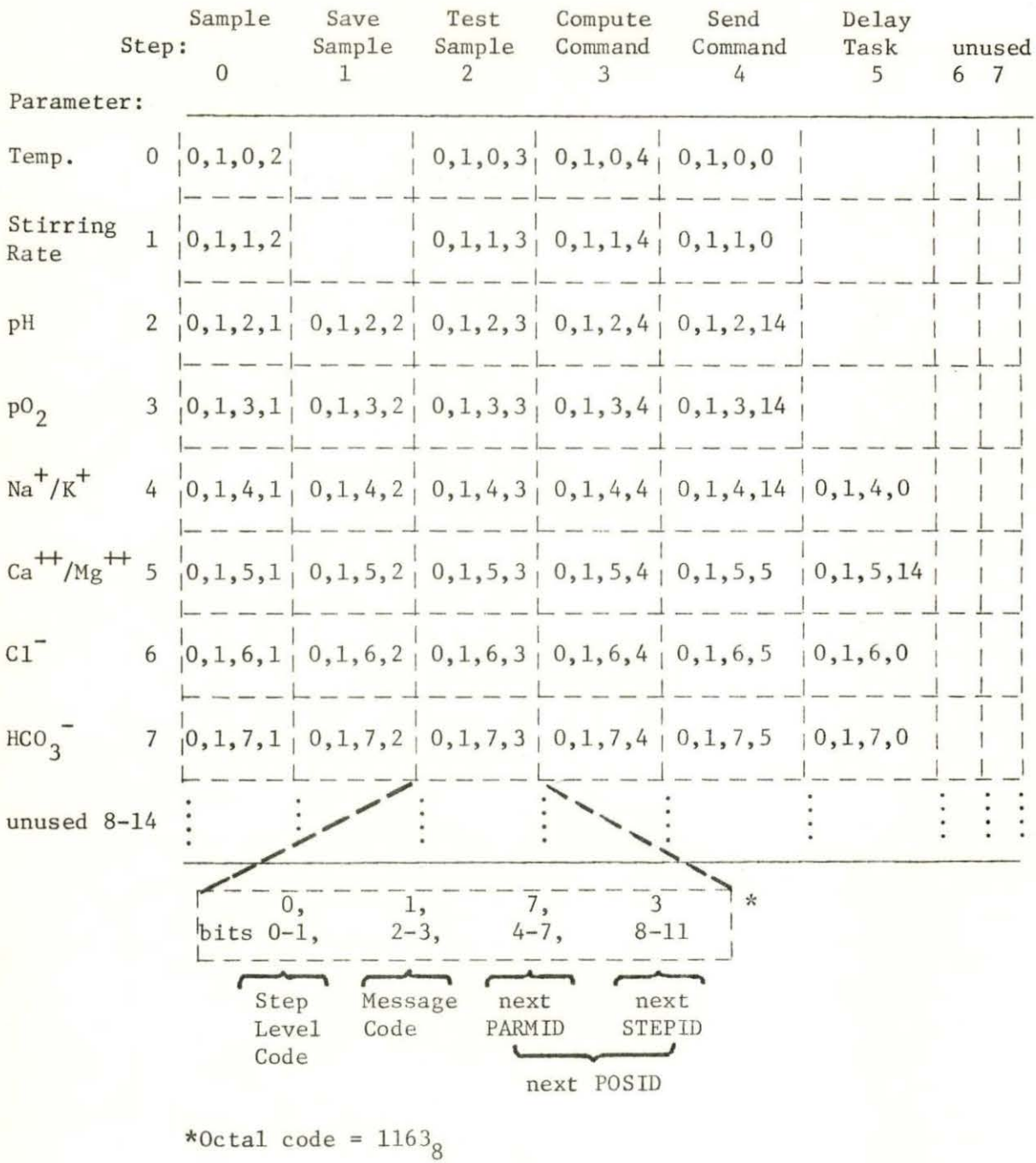
Figure 5. Scheduler information Lists

may be attempted for a given task since there are three sources for the entry. By assigning priorities from low to high, respectively, to the three sources listed above, the scheduler resolves this problem; the entry from the highest priority source is saved and the others are discarded.

Monitor Control Block

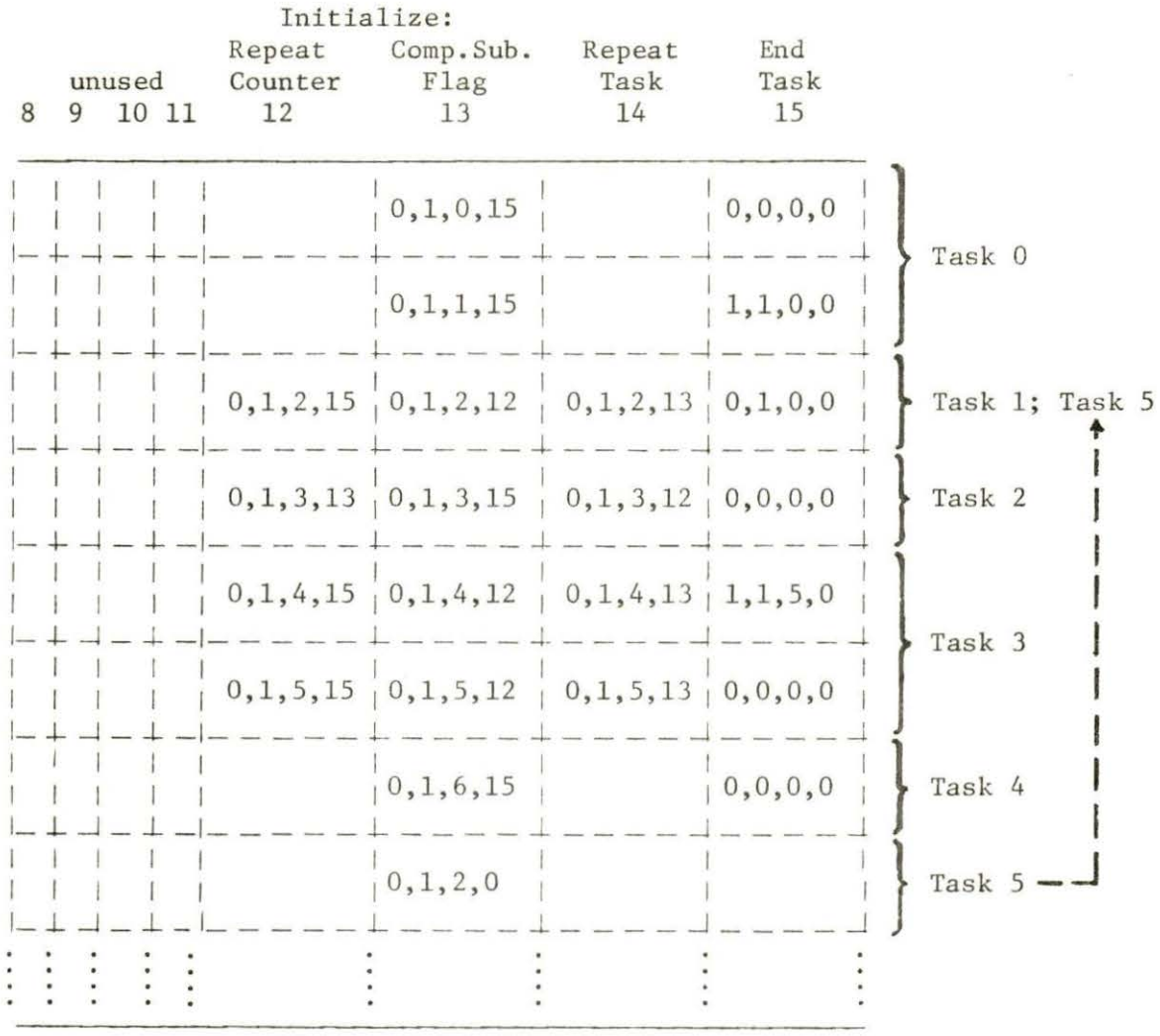
To identify the program's status with respect to the experiment, four registers are used: the PARMID, STEPID, POSID, and TASKID Registers which contain the identification numbers previously described. The TASKID Register is set by the scheduler and the other three are set by the MCB except that when a task is passed to the MCB from the scheduler, the scheduler determines the MCT position (e.g. POSID) at which the task is to begin. Each entry in the MCT, shown in Figure 6, has four fields: the Step Level Code, Message Code, next PARMID, and next STEPID. The latter two fields identify the POSID of the next MCT entry to be used for the current task after the current step is completed. Each table entry occupies one word (twelve bits) of memory and the present allocation of bits per field limits the MCT dimensions to fifteen parameters by sixteen steps. Since the maximum PARMID is fifteen, the combination of all bits set to "one" is an invalid MCT entry; therefore, this value is used in the TASKID Register to signify that no task is currently active.

Two routines, the Monitor Control Loop and the Interpreter are involved in processing the MCT and resetting the identification registers.



*Octal code = 1163_8

Figure 6. Monitor Control Table with test experiment


Initialize:
unused
Repeat Counter
Comp.Sub. Flag
Repeat Task
End Task
8 9 10 11
12
13
14
15
0,1,0,15
0,0,0,0
Task 0
0,1,1,15
1,1,0,0
0,1,2,15
0,1,2,12
0,1,2,13
0,1,0,0
Task 1; Task 5
0,1,3,13
0,1,3,15
0,1,3,12
0,0,0,0
Task 2
0,1,4,15
0,1,4,12
0,1,4,13
1,1,5,0
Task 3
0,1,5,15
0,1,5,12
0,1,5,13
0,0,0,0
0,1,6,15
0,0,0,0
Task 4
0,1,2,0
Task 5

Other routines in this block implement a set of standard monitor messages, such as the Trace Message (indicated by the Message Code set to "one"), and implement the feature of inserting a time delay between steps.

Monitor Control Loop

Each pass through the MCL (Figure 7) completes the processing of one step. The MCT Pointer, which is the actual memory address of the MCT entry being processed, is reset via the POSID and is passed to the Interpreter. After control is returned from the Interpreter, the MCL examines the Monitor Message Register and the Monitor Delay Register which are set by either the Interpreter or a service routine for a nonzero value. If either is nonzero, a message request and/or a delay request is processed as illustrated by the flow diagram in Figure 7. Two options are provided for the time delay feature as flagged by bit zero of the Monitor Delay Register. If it is set, a "pause", measured in hardware clock's time units, is specified and all task processing ceases during this pause. If bit zero is clear, the delay is measured in terms of the STU and the task is returned to the scheduler allowing another task to be processed.

As mentioned previously, a clock check normally (by default) occurs between each step. If, however, the Task Interrupt Flag has been set by the scheduler via information contained in its Task Definition Table, the clock check is ignored. Thus, the experimenter has the option of specifying that a task (such as one of low priority) be processed to completion once it has been started. Note that the clock check feature defines a step as the largest unit of parameter processing work that cannot be interrupted by program action.

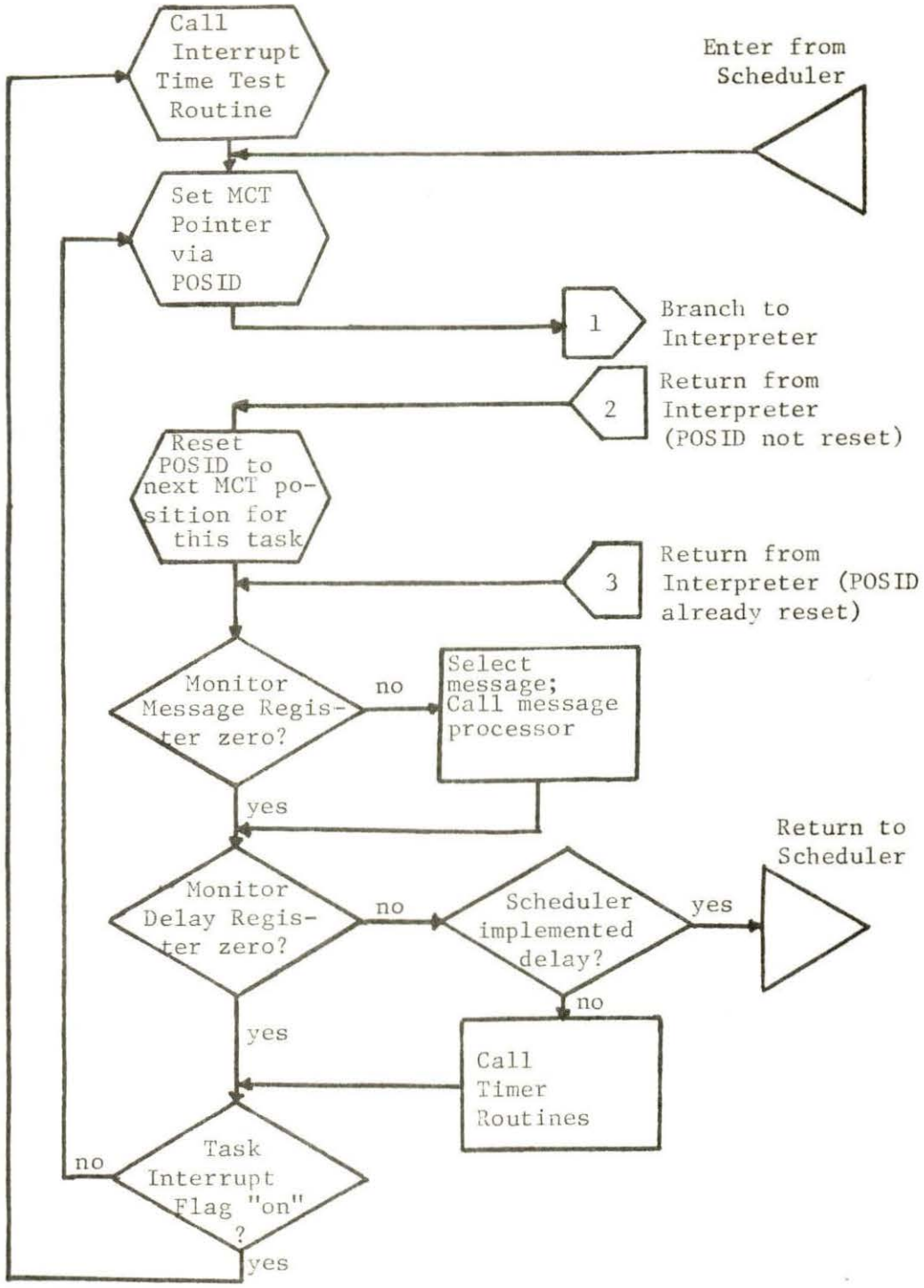


Figure 7. Flow diagram of Monitor Control Loop

Interpreter

Using the MCT pointer, the MCT entry for the current step is obtained (Figure 8) and its Step Level Code is compared with the contents of the Task Level Register which has been set by scheduler from the current task's entry in the Task Definition Table. If the Step Level Code is greater, the step is bypassed and control is returned to the MCL. This feature allows two tasks to be overlaid on the same positions of the MCT and increases the flexibility of the MCT's use. If the Step Level Code is not greater, the Interpreter uses the STEPID as an index to find in its Service Routine Address Table (SRAT) the address of the service routine which implements this step (Figure 9). Therefore, the procedure for linking a service routine to the rest of the program is to enter its memory address into the SRAT. When the step is completed, control is returned to the MCL, but before the return occurs, the service routine may cause a "branch" in the MCT if it stores a new POSID and returns to the MCL at the return point which follows the MCL's POSID reset operation as shown in Figures 7 and 8.

Monitor Service Routine Block

Routines in this block can be classified, as either processing routines (e.g. computation routines) or program control routines (e.g. a loop counter) and can be added, deleted, or modified in order to tailor the program to the requirements of a particular experiment. As already stated, the entry address of each of these service routines must be included in the Interpreter's Service Routine Address Table.

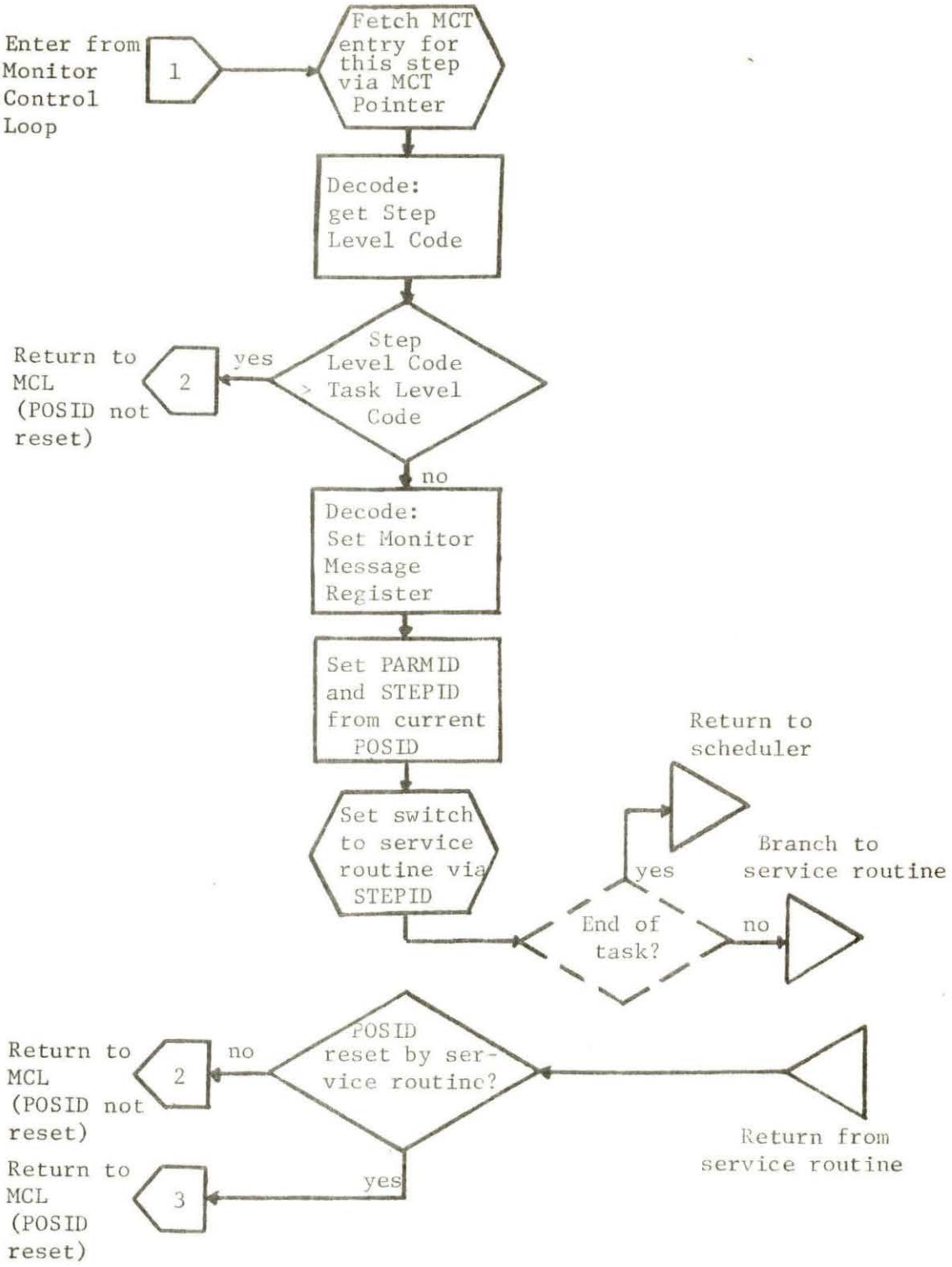


Figure 8. Flow diagram of Interpreter

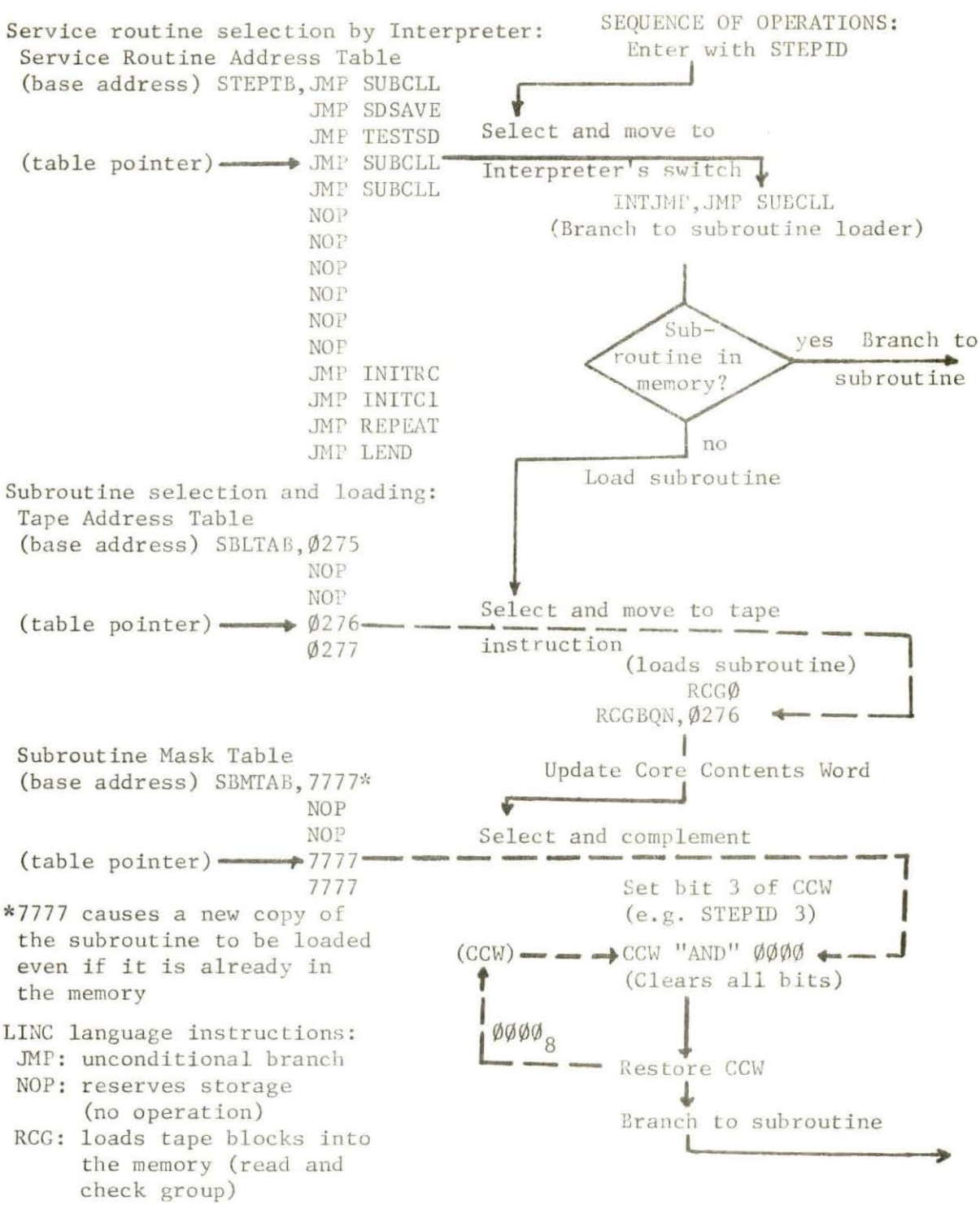


Figure 9. Example of service routine (subroutine loader) and subroutine calling operations for STEPID = 3 (computation step in the test experiment)

If a step, such as step 3 in Figure 6, requires the use of a stored subroutine, the address used from the SRAT is that of the subroutine loader and the STEPID identifies the subroutine and is used by the loader to find and load the subroutine as described below and shown in Figure 9. A Core Contents Word (CCW) is used to keep a record of the subroutines currently in the memory, and if the required subroutine is not in the core, the Tape Address Table is used to find the subroutine and load it; the Subroutine Mask Table entry for this subroutine is used to update the CCW by clearing the bits corresponding to subroutines that have been assigned to the same core locations as the subroutine which is loaded. Thus, tape areas and core areas are statically assigned by the experimenter at the time of program assembly and for each subroutine an entry is required in both the Subroutine Mask Table and the Tape Address Table. Use of a single 12-bit memory word for the CCW limits the allowable number of subroutines to twelve. However, at the expense of using more storage for the tables and for the CCW, a greater number of subroutines can be used.

Program control routines include decision routines, loop counters, and flag setting routines. Depending upon the results of a decision or a counter test, these routines can reset the POSID Register as mentioned previously. For example, in the test experiment, the Sample Data Test Routine compares the parameter sample to a pair of set points for that parameter. If the value is within the set point range, the routine resets the POSID Register to the last step in the MCT which clears the task from the TASKID Register. Otherwise, the MCL resets the POSID Register to the step which computes the correction. As illustrated, program control routines can be quite short but usually require one or more tables of

information.

Record filing operations can be assigned to this block as illustrated by the test experiment's Sample Data Save Routine which establishes a storage area for the most recent sample value for each of the parameters and also provides the option and the mechanics of storing these values on tape using the data buffer areas assigned to memory segment three.

Interrupt and Trap Block

Saving and restoring the field registers, index registers, accumulator, and "link" is provided by the Interrupt and Trap Processors (Figure 3) and when an external program interrupt occurs, the interrupt hardware is turned "off" until the interrupt has been processed. Priorities of interrupts are defined by the order in which the "cause" of the interrupt is determined. Via this structure, new interrupts can be added to the system and the priorities of existing interrupts can be changed.

At present, the only interrupt used is the clock interrupt and the instruction trap is used only for simulating the input data "sampling", of the peripheral data terminals by the CPU as described later. However, Teletype and magnetic tape unit interrupts are available and can be added to the program.

Message Processing Block

Two main routines comprise this block: the Message Queue Loader (MQL) and Message Queue Processor (MQP). See Figure 3. Message writing rules are established by the MQL and MQP as a subroutine call with a call list

which includes the format's name, the device code and the variable list's name (Figure 10). Both routines provide an entry point to the Message Processing Block and also the field register saving which is necessary since this block is called from across a memory segment boundary; however, they share a common field register restoring routine.

For each variable in the variable list, the MQL looks up the current value and loads it, together with the format's name and the device code, on the Message Queue (MQ). An option is provided which allows a string of bits to be lifted from the value of the variable, right adjusted, and used by the MQP as the value of the variable. The option is used by setting bit zero of the variable's name to "one". A "mask" word, specifying the bits to be used, must follow the variable's name in the variable list. The option is used when two or more values are packed into a single memory word and are to be included in a message.

If a message is to be sent immediately when requested, flagged by bit zero of the call list's name, the message is loaded into the MQ's buffer area (which also provides for overflow from the main MQ area) and control is passed to the MQP. Otherwise, the message is loaded into the main MQ area and control is returned to the calling routine. Messages are sent from the main MQ area on a time available basis as determined by the Idler or the Timer Test Routine, both of which call the MQP. Two address pointers are used by the MQP when processing messages from either the MQ or its buffer: the Value List Pointer which addresses the next variable's value, and the Format Character Pointer which addresses the next character in the format specification.

Message formats contain delimiting characters dividing the formats into fields corresponding to the following format elements: character string constants, message variables, character spacing (blanks), CRT vertical and horizontal coordinates, Teletype carriage return and line spacing (new line) and "end of format". For both the CRT and the Teletype, automatic control of line spacing and character position is provided by the MQP so that it is unnecessary to use these specifications unless a non-standard format is desired. Since a format is organized and processed as a set of fields, new format specifications can be incorporated into the design of the MQP.

```
Message "call list":
                        ODMES,  ODFORM&5777
                                7777
                                ODLIST!2000
Message "variable list":
                        ODLIST, ODVAL!2000
                                ODPARM!2000
                                ODHRS!2000
                                ODMIN!2000
                                ODSEC!2000
                                TEXT ZØ;Z
Message "format":
                        /OUTPUT COMMAND SUBROUTINE MESSAGE FORMAT/
                        ODFORM, TEXT Z( OUTPUT VALUE = )VZ
                                TEXT Z( PARMID = )VCLB7,(TIME=Z
                                TEXT Z)V( HOURS )V( MINUTES )Z
                                TEXT ZV( SECONDS. )CL;Z
```

Message "calling instruction sequence" for a call from segment 2 with the call list and variable list in segment 2 and the format in segment 1:

```
                                LDA I
                                ODMES!2000&3777
                                LDF 2
                                LIF 1
                                JMP OLLOAD
```

Teletype message:

```
OUTPUT VALUE = -1      PARMID = 4
     TIME=Ø      HOURS  12      MINUTES  3ØØ      SECONDS.
```

note: numerical values are in octal and the unit for the "seconds" value is decimal tenths of seconds (the unit of the hardware clock)

LINC mode instructions:

LDA I - loads accumulator with contents of the following memory location
LDF 2 - changes Data Field to segment two
LIF 1 - changes Instruction Field to segment one when the next branch instruction occurs
JMP - branches to location labelled by the symbol, OLLOAD (in segment one)

DIAL assembler instructions:

"!" - logical "OR" of the two adjacent values
"&" - logical "AND" of the two adjacent values
(used to set bit zero, the flag, for queued processing and to set bit one, the field bit, to the I.F. segment or to the D.F. segment)

Figure 10. Teletype message for queued processing

PROGRAM TESTING

A test experiment, designed from requirements of the cell culture system, was coded into the Monitor Control Table (Figure 6) and Task Definition Table (Figure 5). Subroutines, stored on tape (Figure 9), were used for steps zero, three, and four in the test experiment. Respectively, these steps provided for obtaining data samples, determining command signals, and sending command signals as described below.

Since testing was done without peripheral hardware, the interaction of the program with the peripheral hardware was simulated by replacing the A/D convertor sampling instruction with an illegal instruction in the sampling subroutine. The instruction trap processor displayed a message on the screen indicating that a data value for the particular parameter should be entered via the console switches. Command signals to the peripheral hardware were replaced by a message (Figure 10) containing the control signal value.

A mathematical model of the cell culture system was not available for use in computing the value of the command. Rather, an iterative correction procedure was used such that the signal value was either plus one or minus one depending upon whether the sample value was below or above, respectively, the set point range defined for the particular parameter. This scheme is compatible with the culture system since its peripheral hardware control devices respond to either single pulses or to "on-off" commands. Thus, the sequence of sampling and correcting continued for each scanning cycle of a given parameter until the sample value, as entered from the console, was within the set point range.

Initial testing was conducted without the hardware clock and time increments were entered into the Clock Routine from the console switches. As additional time was entered, more of the program loops and routines were called into operation so that the progress of the testing was controlled by the time entries.

With the hardware clock added to the system, a real-time environment was established and it was possible to test the program with respect to its efficiency for various combinations of task cycle times and task delay times. Since testing was done off-line to the peripheral hardware and data were entered from the console, the trap processor removed the system from its real-time environment and after the sample was entered, the clock was restored to the point in time that existed when the trap occurred. The pause in the real-time environment provided the opportunity to stop the CPU, check the contents of the memory and registers, and make changes in test values without affecting the real-time environment of the test.

In order to observe its operation, CRT display messages were inserted in important sections of the program. In addition, the Trace Message which prints the TASKID, PARMID, STEPID and the time (Appendix A) verified the sequence of processing for each task. With respect to the task cycle and delay times, the efficiency of the program proved to be sensitive to the amount of time required for the Trace and display messages. For this reason two additional features were added so that message time could be reduced while the CPU was in operation: a console sense switch was used to turn the Trace Message "on" or "off" and a console A/D knob was used

to control the display time of a CRT message so that when the knob was turned to zero, the display was eliminated. About one minute of delay time was used for stabilization of any ion specific electrode, and during this time, the electrode amplifier was unavailable for use by the other electrodes. Thus, processing of the other ion parameters was also delayed. This appeared to be the primary limitation to system performance since, as the cycle times were decreased, the lower priority parameters were processed to completion less often.

DISCUSSION

Successful operation of the program was confirmed by the final off-line tests, during which additional messages and service routines were incorporated into the program. The ease with which these additions were made illustrated the utility of the program design and implementation of the test experiment did not require any basic changes in the original design.

Several factors concerning the use of the Monitor Control Table may not be immediately apparent. As described, the task processing is controlled by a time scheduler, but initiation of a given task can be accomplished via a signal from the experimenter by setting the task's cycle time to zero and using an initial step to test a sense switch. Or if the event signal is to come from the external hardware, the interrupt system can be used in place of the sense switch. For the latter situation, the scheduler may require redesign which is facilitated by the block structure of the program since task scheduling is performed by the Scheduler Block.

It may appear from the test experiment that the processing of each parameter is meant to be a single task with the sequence of steps moving across the MCT. Actually, an experiment involving many parameters can be designed as a single task if parameter priorities are not required; and the sequence can be vertical as is the case if the data samples for all parameters are required before any of the computations can be accomplished. Furthermore, the entire experiment can be programmed as

a single task with a single step if the identification of the parameters is maintained by the processing program illustrating that, although the MCT size is fixed, the functions performed by two or more steps can be combined into one step if a time test is not required between these functions. Thus, the design of the MCB provides considerable flexibility in adapting the program to the experiment.

BIBLIOGRAPHY

1. Chorafas, Dimitris N. Control systems functions and programming approaches. Vol. A. New York, New York, Academic Press, Inc. 1966.

2. Desmonde, William H. Real-time data processing systems: introductory concepts. Englewood Cliffs, N. J., Prentice-Hall, Inc. 1964.

3. Digital Equipment Corporation. PDP-12 User Handbook. Maynard, Massachusetts, Author. 1969.

4. Quinn, L. Y. Continuous culture of ruminal microorganisms in chemical medium. I. design of continuous culture apparatus. Applied Microbiology 10: 580-582. 1962.

5. Rothstein, Michael F. Guide to the design of real-time systems. New York, New York, Wiley-Interscience. 1970.

6. Vosseller, George V. and George E. Moore. The trophocell, an installation for large scale mammalian cell culture. Research/ Development 20: 20-24. 1969.

7. Watson, Richard W. Timesharing system design concepts. New York, New York, McGraw-Hill, Inc. 1970.

8. Wilkes, M. V. Time-sharing computer systems. New York, New York, American Elsevier Publishing Company, Inc. 1968.

9. Yourdan, Edward, ed. Real-time systems design. Cambridge, Massachusetts, Information and Systems Institute, Inc. 1967.

10. Zimmerli, D. W. An electronic counter for population and size distribution of microscopic particle suspensions. Unpublished M.S. thesis. Ames, Iowa, Library, Iowa State University of Science and Technology. 1967.

ACKNOWLEDGMENTS

I wish to thank Dr. David L. Carlson for suggesting the project and for his patient encouragement and guidance during the development of the program and the preparation of the thesis. To Dr. Richard L. Engen goes my appreciation for his suggesting valuable improvements in the manuscript and for serving, with Dr. Carlson, as a co-major professor. Special thanks go to Dr. Loyd Y. Quinn whose dedication to the study of immunology and cell cultures made this project possible.

APPENDIX A

Test Output

```
LO ASSEMBL4,0
TRACE:  TØ      P1      SØ      141     SEC  1    MIN
TRACE:  TØ      P1      S2      233     SEC  1    MIN
TRACE:  TØ      P1      S3      326     SEC  1    MIN
TRACE:  TØ      P1      S4      424     SEC  1    MIN
TRACE:  TØ      P1      SØ      527     SEC  1    MIN
TRACE:  TØ      P1      S2      621     SEC  1    MIN
TRACE:  TØ      P1      S3      714     SEC  1    MIN
TRACE:  TØ      P1      S4      1012    SEC  1    MIN
TRACE:  TØ      P1      SØ      1117    SEC  1    MIN
TRACE:  TØ      P1      S2      31      SEC  2    MIN
TRACE:  TØ      P1      S3      123     SEC  2    MIN
TRACE:  TØ      P1      S4      221     SEC  2    MIN
TRACE:  TØ      P1      SØ      325     SEC  2    MIN
TRACE:  TØ      P1      S2      416     SEC  2    MIN
TRACE:  TØ      P1      S3      511     SEC  2    MIN
TRACE:  TØ      P1      S4      607     SEC  2    MIN
TRACE:  TØ      P1      SØ      713     SEC  2    MIN
TRACE:  TØ      P1      S2      1005    SEC  2    MIN
TRACE:  TØ      P1      S15     1100    SEC  2    MIN
TRACE:  TØ      PØ      SØ      42      SEC  3    MIN
TRACE:  TØ      PØ      S2      133     SEC  3    MIN
TRACE:  TØ      PØ      S3      225     SEC  3    MIN
```

```
 OUTPUT VALUE = -1      PARMID = 1
       TIME=Ø      HOURS  1      MINUTES  424     SECONDS.

TRACE:  TØ       PØ       S4          324     SEC  3     MIN

TRACE:  TØ       PØ       SØ          573     SEC  3     MIN

TRACE:  TØ       PØ       S2          665     SEC  3     MIN

TRACE:  TØ       PØ       S3          760     SEC  3     MIN

 OUTPUT VALUE = -1      PARMID = 1
       TIME=Ø      HOURS  1      MINUTES  1Ø12    SECONDS.

TRACE:  TØ       PØ       S4          1056    SEC  3     MIN

TRACE:  TØ       PØ       SØ          147     SEC  4     MIN

TRACE:  TØ       PØ       S2          240     SEC  4     MIN

TRACE:  TØ       PØ       S15         333     SEC  4     MIN

 OUTPUT VALUE = -1      PARMID = 1
       TIME=Ø      HOURS  2      MINUTES  221     SECONDS.

 OUTPUT VALUE = -1      PARMID = 1
       TIME=Ø      HOURS  2      MINUTES  6Ø7     SECONDS.

 OUTPUT VALUE = 1       PARMID = Ø
       TIME=Ø      HOURS  3      MINUTES  324     SECONDS.

 OUTPUT VALUE = 1       PARMID = Ø
       TIME=Ø      HOURS  3      MINUTES  1Ø56    SECONDS.

TRACE:  T1       P2       SØ          514     SEC  5     MIN

TRACE:  T1       P2       S1          514     SEC  5     MIN

TRACE:  T1       P2       S2          514     SEC  5     MIN

TRACE:  T1       P2       S15         514     SEC  5     MIN
```

```
TRACE:  T1      P2      S14       514    SEC  5    MIN
TRACE:  T2      P3      SØ        75     SEC  7    MIN
TRACE:  TØ      P1      SØ        243    SEC  7    MIN
TRACE:  TØ      P1      S2        335    SEC  7    MIN
TRACE:  TØ      P1      S3        43Ø    SEC  7    MIN
TRACE:  TØ      P1      S4        526    SEC  7    MIN
TRACE:  TØ      P1      SØ        631    SEC  7    MIN
TRACE:  TØ      P1      S2        723    SEC  7    MIN
TRACE:  TØ      P1      S15       1016   SEC  7    MIN
TRACE:  TØ      PØ      SØ        112Ø   SEC  7    MIN
TRACE:  TØ      PØ      S2        32     SEC  1Ø   MIN
TRACE:  TØ      PØ      S15       125    SEC  1Ø   MIN
TRACE:  T2      P3      S1        222    SEC  1Ø   MIN
TRACE:  T2      P3      S2        434    SEC  1Ø   MIN
TRACE:  T2      P3      S14       53Ø    SEC  1Ø   MIN
TRACE:  T2      P3      S15       625    SEC  1Ø   MIN

 OUTPUT VALUE = 1      PARMID = 1
       TIME=Ø     HOURS  7     MINUTES  526    SECONDS.

TRACE:  T3      P4      SØ        1Ø35   SEC  11   MIN
TRACE:  T3      P4      S1        113Ø   SEC  11   MIN
TRACE:  T3      P4      S2        1Ø5    SEC  12   MIN
TRACE:  T3      P4      S3        2ØØ    SEC  12   MIN
TRACE:  T3      P4      S4        3ØØ    SEC  12   MIN
TRACE:  T3      P4      S16       377    SEC  12   MIN
```

APPENDIX B

Table 2. Block Summary

Table 2. Block Summary

Block	Function	Main Component Routines
Supervisor Block	Provides primary program control and timing functions needed by the program	Idler Routine Clock Routine Interrupt Time Test Routine Timer Set and Test Routine
Scheduler Block	Implements the initiation and delay of processing tasks on a time interval basis	Delay List Test, Delay List Load, Task List Test, Priority Queue Test, Priority Queue Load, Task Starter
Monitor Control Block	Initiates and monitors the execution of each step of each task	Monitor Control Loop Interpreter Monitor Delay Processor Monitor Message Processor
Monitor Service Routine Block	Implements the execution of each step	Subroutine Call and Load Test Sample Data Sample Data Save
Interrupt and Trap Block	Services the external program interrupt and instruction trap	Interrupt Processor Trap Processor
Message Processing Block	Processes message requests for the teletype printer and for the CRT	Message Queue Loader Message Queue Processor

Table 2 (Continued)

Tables and Data Areas Used	Major Data and Control Registers	Block
none	Total Experiment Time Registers Elapsed Time Register Timer Test Registers	Supervisor Block
Task Delay List Task Cycle Time List Task Definition Table Priority Queue	TASKID Register, STEPID Register, POSID Register Monitor Delay Register Task Interrupt Flag	Scheduler Block
Monitor Control Table Service Routine Address Table Monitor Message Table	TASKID, PARMID, STEPID, POSID Registers, Task Interrupt Flag, MCT Pointer, Monitor Message Register Monitor Delay Register	Monitor Control Block
Tape Address Table Subroutine Mask Table Sample Data Buffers	Core Contents Word	Monitor Service Routine Block
Register Save Areas	Interrupt Mode Flag	Interrupt and Trap Block
Message Queue and Buffer	Queue Load Pointer, Buffer Load Pointer, Value List Pointer, Format Character Pointer CRT Scan Counter	Message Processing Block

APPENDIX C

Program Listing

```
        SEGMNT 0
        *451
/ SUPERVISOR BLOCK
SIDLER, JMP TESTIC
        LDA I
        XSIDLE!4000
        LIF 1
        JMP OLLOAD
        LIF 1
        JMP OCNTL1
        JMP SIDLER
SSCALL, STC SSCACC
        ADD 2
        BSE I
        6000
        STC SSCJMP
        ROR I 1
        STC SSSAVL
        ADD 0000
        STC SSCRTN
        IOB
        6234
        ROR 3
        STH
        SSLIF!4000
        SCR 7
        BCL I
        7774
        BSE I
        0640
        STC SSLDF
        ADD SSCACC
SSCJMP, NOP
        STC SSCACC
SSCRST, LDA I
SSSAVL, NOP
        ROL I 1
        LDA I
SSCACC, NOP
SSLDF,  NOP
SSLIF,  0600
SSCRTN, NOP
READSC, IOB
        6002
        LDA
        0000
        STC CLKRTN
        IOB
        6302
        AZE I
        JMP CLKEND
        ADM I
SECCLK, 0000
        SET I 6
        MININC
        JMP CLKDIV
        STC SECCLK
/
        IOB
        6304
        SET I 6
        HRSINC
        ADD 7
        ADM I
EXPCLK, 0000
        LDA
        0007
        ADA I
MINCLK, 0000
        JMP CLKDIV
/
        STC MINCLK
/
        ADD 7
        ADM I
HRSCLK, 0000
/
        LDA I
        INTINC
        STC 6
        ADD EXPCLK
        JMP CLKDIV
        STC EXPCLK
        ADD 7
        ADM I
INTCLK, 0000
CLKEND, SRO
        IMFLAG
        JMP .+3
        IOB
```

```
        6001
CLKRTN, NOP
MININC, -1160
HRSINC, -0074
INTINC, -0001
/
CLKDIV, STC 5
        LDA
        0
        STC 4
        CLR
        STC 7
        ADD 5
        JMP .+2
DVLOOP, XSK I 7
        STA
        5
        ADA 6
        FLO
        APO I
        JMP DVLOOP
        AZE
      JMP .+4
        CLR
        XSK I 7
        STC 5
        LDA
        4
        STC 0
        LDA
        5
        JMP 0
/
/
TIMERS, STC SAVETS
        ADD 0
        STC 2
        JMP READSC
        SET I 6
        MININC
        LDA I
SAVETS, 0000
TRSCON, ADD SECCLK
        JMP CLKDIV
        STC TTSECV
        SET I 6
        HRSINC
        ADD 7
        ADD MINCLK
        JMP CLKDIV
        STC TTMINV
        ADD 7
        ADD HRSCLK
        STC TTHRSV
        ADD 2
        STC 0
        JMP 0
/
TIMERT, LDA
        0
        STC 2
        JMP TMTCON
TTLOOP, SRO I
TMTSW,  0000
        JMP TMTCON
        LDA I
        XTIMET!4000
        LDF 0
        LIF 1
        JMP OLLOAD
        LDF 1
        LIF 1
        NOP
        JMP OCNTL1
TMTCON, JMP READSC
        LDA
        HRSCLK
        COM
      ADA I
TTHRSV, 0000
        APO I
        JMP TTLOOP
        AZE
        JMP TMTEND
        LDA
        MINCLK
        COM
        ADA I
TTMINV, 0000
        APO I
        JMP TTLOOP
        AZE
        JMP TMTEND
        LDA
        SECCLK
        COM
        ADA I
TTSECV, 0000
        APO I
        JMP TTLOOP
TMTEND, LDA
        2
```

```
        STC 0
        JMP 0
/
TESTIC, LDA
        0
        STC TICRTN
        JMP READSC
TTCON1, LDA
        INTCLK
        AZE I
TICRTN, NOP
/
        STC ICOUNT
        STC INTCLK
AAZSSB, JMP DQTST1
/END OF SSB
        SEGMNT 0
        *764
/SCHEDULER BLOCK, "SCB"
DQTST1, SET I 11
        1777
/
        JMP READSC
        JMP DQCON1
DQTST2, SET I 11
        0000
DQCON1, LDA I
        DTIMEQ-1!2000
        STA
        0002
        STC 0003
      LDA I
DTQEND, DCODEQ-1!2000
        STA
        0004
        STC 0005
        ADD ICOUNT
        COM
        STA I
ICCOM,  NOP
        CLR
        STC ICOUNT
DQLOOP, LDA I 2
        APO
        JMP DQCON3
        ADD ICCOM
        APO
        JMP DQCON2
        STA I 3
        LDA I 4
        STA I 5
        JMP DQLOOP
/
DQCON2, LDA I 4
        BCL I
        0377
        ROR 8
        STA
        PQTASK
        LDA 4
        STC PQCODE
/
        SET I 12
        7777
        JMP PQLOAD
        JMP DQLOOP
DQCON3, STA I 3
        XSK 11
/
        JMP DQLOAD
        JMP TQTEST
TQTEST, LDA I
        TTIMEQ-1!2000
        STC 5
        STC 6
        JMP TQCON1
TQLOOP, STA 5
        XSK I 6
TQCON1, LDA I 5
        APO
        JMP TFETCH
        ADD ICCOM
        APO I
        JMP TQLOOP
        LDA
      0006
        STA
        PQTASK
        ADA I
        TDTAB!2000
        STC 7
        LDA 7
        STC PQCODE
        STC 12
        JMP PQLOAD
TQCON2, LDA
        0006
        ADA I
        TTTAB!2000
        STC 7
        LDA 7
        JMP TQLOOP
```

```
DQLOAD, LDA
        0003
        SAE I
        DCODE0-2!2000
        JMP .+2
        JMP DQLCON
        LDA
        DVALUE
        STA 3
        LDA
        TASKID
        ROL 8
        BSE
        POSID
        STA I 5
        CLR
        COM
        STA I 3
        STC TASKID
DQLCON, CLR
        STC DVALUE
        JMP TQTEST
PQLOAD, LDA
        0000
        STC PQLRTN
        LDA I
PQTASK, NOP
        ADA I
        PQUEUE!2000
        STC 15
        LDA 15
        SAE I
        7777
      JMP .+2
        JMP .+5
        LDA
        12
        APO I
        JMP PQCON1
        LDA I
PQCODE, NOP
        STA 15
        LDA I
        XPQLOD!4000
        LDF 0
        LIF 1
        JMP OLLOAD
        LDF 1
/
PQCON1, LDA I
PQLRTN, NOP
        STC 0
        JMP 0
TFETCH, LDA I
        PQUEUE-1!2000
        STC 2
        STC 3
        LDA I
POSIZE, -14
        STC 4
        JMP TFCON1
TFLOOP, XSK I 3
TFCON1, LDA I 2
        SAE I
        7777
        JMP TFCON2
        XSK I 4
        JMP TFLOOP
        LDA
        TASKID
        APO
        JMP SIDLER
        JMP MCLOOP
TFCON2, LDA
        TASKID
        APO
        JMP TSTRT2
        COM
        ADD 3
        AZE
        APO I
        JMP MCLOOP
        JMP TSTRT1
TSTRT1, LDA I
        PQUEUE!2000
      ADA
        TASKID
        STC 4
        ADD POSID
        STA 4
        LDA I
        XPINT!4000
        LDF 0
        LIF 1
        JMP OLLOAD
        LDF 1
/
TSTRT2, LDA 2
        BCL I
        7400
        STC POSID
        ADD 3
```

```
        STC TASKID
        COM
        STA 2
        LDA I
        TDTAB!2000
        ADD 3
        STC 4
        LDA 4
        SCR 13
        STC TIFLAG
        QAC
        SCR 12
        AZE
        NOP
        QAC
        SCR 11
        STC TLEVEL
        LDA I
        XSTART!4000
        LDF 0
        LIF 1
        JMP OLLOAD
        LDF 1
AAZSCR, JMP MCLOOP
        SEGMNT 0
        *1300
/MONITOR CONTROL LOOP
MITEST, JMP TESTIC
MCLOOP, LDA I
POSID,  NOP
        NOP
        ADA I
        MONTAB!2000
        STC MTABPT
        JMP MONINT
MRESET, LDA
        MTABPT
     STC 3
        LDA 3
        BCL I
        7400
        STC POSID
/
MCON1,  LDA I
MOCODE, 0000
        AZE
        JMP MONOUT
MCON2,  LDA I
DVALUE, 0000
        APO I
        AZE
        JMP MDELAY
/
MCON3,  SRO I
TIFLAG, 0000
        JMP MCLOOP
        JMP MITEST
/MONITOR INTERPRETER
MONINT, LDA
MTABPT, NOP
        NOP
        SCR 12
        BCL I
        7774
        AZE
        JMP LTEST
MICON1, QAC
        SCR 11
        STC MOCODE
        ADD POSID
        SCR 4
        STA I
PARMID, NOP
        QAC
        SCR 7
        STA I
STEPID, NOP
        ADA I
        STEPTB!2000
        STC 14
        LDA 14
        STC INTJMP
INTJMP, NOP
MICON2, JMP MRESET
/END OF "MONINT"
/MONITOR OUTPUT ROUTINE
MONOUT, ADA I
        MONMSG-1
        STC 7
        STC MOCODE
        SNS I 4
        JMP MCON2
        LDA 7
     LIF 1
        LDF 0
        JMP OLLOAD
MORST,  LDF 1
        JMP MCON2
MONMSG, TRACE&5777!4000
        NOP
        NOP
/END OF MONOUT
```

```
/LEVEL TEST ROUTINE:
LTEST,  COM
        ADA I
TLEVEL, 0000
        AZE
        APO I
        JMP MICON1
        JMP MRESET
MDELAY, APO
        JMP PAUSE
        LDA I
        XDELAY!4000
        LDF 0
        LIF 1
        JMP OLLOAD
        LDF 1
        JMP DOTST2
PAUSE,  JMP TIMERS
PAUSE1, CLR
        STC DVALUE
        JMP TIMERT
PAUSE2, JMP MCON3
AAZMCB, NOP
        SEGMNT 0
        *1440
SUBCLL, JMP SUBLD
        NOP
        NOP
SCCON1, LIF 2
SUBJMP, NOP
        NOP
        NOP
SBRTN1, JMP MRESET
        NOP
        NOP
SBRTN2, JMP MCON1
/END OF SUBCALL
SUBLD,  LDA I
        SBLTAB!2000
        NOP
        ADD STEPID
        STC 2
        LDA I
        ROL 0
        ADD STEPID
        STC ROLCCW
    ADD CORECW
ROLCCW, NOP
        APO
        JMP SSBJMP
        STC CORECW
        LDA 2
        STC RCGBON
/LDA I DELETED
        NOP
        NOP
        LDF 2
        RCG 0
RCGBON, NOP
        LDF 1
        NOP
        LDA I
        ROR 0
        ADD STEPID
        STC RORCCW
        LDA I
        SBMTAB!2000
        ADD STEPID
        STC 3
        LDA I
CORECW, 0000
        BSE I
        4000
RORCCW, NOP
        BCL 3
        STC CORECW
SSBJMP, LDA 2
        BCL I
        7774
        MUL I
        0400
        BSE I
        6000
        STC SUBJMP
        JMP SCCON1
/END OF SUBLOAD
/DATA SAVE ROUTINE:
SDSAVE, LDA I
SCOUNT, 105
        APO
        JMP SASWAP
SDCON1, ADA I
        -0001
        STC SCOUNT
        LDA I
SAVEPT, SAREA1!2000
        STC 0010
        ADD PARMID
        LDF 3
        STA I 10
        LDF 1
        ROL 1
```

```
      ADA I
          SDBUFF!2000
          STC 11
          LDA 11
          LDF 3
          STA I 10
          LDF 1
          LDA I 11
          LDF 3
          STA I 10
          LDF 1
          LDA
          0010
          STC SAVEPT
          JMP MRESET
SASWAP,   LDA I
          0010
          AXO
          STD
          JMP .-1
          WRI
DWLIST,   6001
          LDA
          DWLIST
          SRO I
SCHAIN,   6314
          ADD NM2000
          ADA I
          1001
          STC DWLIST
          LDA I
          SAREA1!2000
          SRO
          SCHAIN
          ADD NP256
          STC SAVEPT
          LDA I
MAXCNT,   0105
          JMP SDCON1
NM2000,   -2000
NP256,    256
/END OF SDSAVE,
/TEST SAMPLE DATA
TESTSD,   LDA
          PARMID
          ROL 1
          ADA I
          SDBUFF!2000
          STC 2
          LDA
          PARMID
          ROL 1
          ADA I
          LIMTAB!2000
          STC 3
          LDA 2
      COM
          ADA 3
          APO I
          JMP MRESET
          LDA I 3
          COM
          ADA 2
          APO I
          JMP MRESET
          LDA
          PARMID
          ADA I
          SKIPTB!2000
          STC 2
          LDA 2
          STC POSID
          JMP MCON1
/DELAY SERVICE ROUTINE
DELAY,    LDA
          PARMID
          ADA I
          DTAB1!2000
          STC 14
          LDA 14
          STC DVALUE
          JMP MRESET
/INITC1,
INITC1,   LDA I
          CABUFF!2000
          ADD PARMID
          STC 2
          STA 2
          JMP MRESET
INITRC,   LDA I
          INRTAB!2000
          ADD PARMID
          STC 2
          LDA I
          RCTAB!2000
          ADD PARMID
          STC 3
          LDA 2
          STA 3
          JMP MRESET
/END OF INITRC,
REPEAT,   SET I 2
```

```
         3
         LDA I
         RPTAB!2000
         ADD PARMID
         STA 2
         ADA I
PARMNO,  10
         STA I 2
         ADD PARMNO
         STA I 2
     ADD PARMNO
         STC 6
         LDA 4
         AZE I
         JMP RPCON1
         LDA I
         -1
         ADM 4
         APO I
         JMP RPCON1
         LDA 5
         STA 4
         JMP MRESET
RPCON1,  LDA 3
         STC POSID
         LDA 6
         STC DVALUE
         JMP MCON1
LEND,    LDA I
         XEND!6000
         LIF 1
         LDF 0
         JMP OLLOAD
         LDF 1
         CLR
         COM
         STC TASKID
         JMP TFETCH
XEND,    XFEND!2000
         0000
         XVCOM2
XFEND,   TEXT Z(END TASK )V;Z  /
         SAREA1=7000
/END OF MSB, FILED AS MSB
TASKID,  7777
ICOUNT,  0000
AAZMSB,  NOP
/INTERRUPT-TRAP BLOCK
         SEGMNT 0
         *40
         NOP
         IOB
         6002
         STC HISAVA
         ROR I 1
         STC HISAVL
         COM
         STA I
IMFLAG,  0000
         /
         CLR
         IOB
         6234
         ROR 3
         STH
         HILIF!4000
     SCR 7
         BCL I
         7774
         BSE I
         0640
         STC HILDF
         ADD 40
         BSE I
         6000
         STC HIRTN
         ADD 0
         STC HISAV0
         ADD 1
         STC HISAV0+1
         ADD 2
         STC HISAV0+2
         ADD 3
         STC HISAV0+3
         ADD 4
         STC HISAV0+4
         ADD 5
         STC HISAV0+5
         ADD 6
         STC HISAV0+6
         ADD 7
         STC HISAV0+7
         /
         HLT
         NOP
         CLR
         ADD HISAV0
         STC 0
         ADD HISAV0+1
         STC 1
         ADD HISAV0+2
         STC 2
```

```
        ADD HISAV0+3
        STC 3
        ADD HISAV0+4
        JMP TPEND+1
        *140
TRAP,   NOP
        STC TPSAVA
        ROR I 1
        STC TPSAVL
        IOR
        6234
        ROR 3
        STH
        TPLIF!4000
        BSE I
        0040
        STH
        TRAPDF!4000
        SCR 7
        BCL I
    7774
        BSE I
        0640
        STC TPLDF
        ADD 140
        BSE I
        6000
        STC TPRTN
        ADD 0
        STC TPSAV0
        ADD 1
        STC TPSAV0+1
        ADD 2
        STC TPSAV0+2
        ADD 3
        STC TPSAV0+3
        ADD 4
        STC TPSAV0+4
        ADD 5
        STC TPSAV0+5
        ADD 6
        STC TPSAV0+6
        ADD 7
        STC TPSAV0+7
        ADD 0140
        ADA I
        -1
        BSE I
        2000
        STC 2
TRAPDF, 0600
        LDA 2
        STA I
TRAPCD, NOP
/
        JMP SIM
TPCON1, CLR
        ADD TPSAV0
        STC 0
        ADD TPSAV0+1
        STC 1
        ADD TPSAV0+2
        STC 2
        ADD TPSAV0+3
        STC 3
        ADD TPSAV0+4
        STC 4
        ADD TPSAV0+5
        STC 5
        ADD TPSAV0+6
        STC 6
        ADD TPSAV0+7
        STC 7
        ADD TPSAVL
        ROL I 1
        ADD TPSAVA
    DJR
TPLIF,  0600
TPLDF,  NOP
TPRTN,  NOP
TPSAVL, NOP
TPSAVA, NOP
TPSAV0, NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
TPEND,  NOP
/
        STC 4
        ADD HISAV0+5
        STC 5
        ADD HISAV0+6
        STC 6
        ADD HISAV0+7
        STC 7
        STC IMFLAG
        ADD HISAVL
        ROL I 1
        ADD HISAVA
```

```
          DJR
HILIF,    0600
HILDF,    NOP
          IOB
          6001
HIRTN,    NOP
HISAVL,   NOP
HISAVA,   NOP
HISAV0,   NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
HIEND,    NOP
/HARDWARE SIMULATOR,
SIM,      IOB
          6002
          CLR
          COM
          STC IMFLAG
          JMP READSC
          LDF 0
SIML1,    LDA I
          MESSS1!6000
          LIF 1
          JMP OLLOAD
          SNS 1
          JMP SIML1
SIML2,        LDA I
          MESSS2!6000
          LIF 1
          JMP OLLOAD
          SNS I 1
          JMP SIML2
          LSW
          STC TPSAVA
          STC IMFLAG
          IOB
          6304
          IOB
          6001
SIMRTN,   JMP TPCON1
/
MESSS1,   SIMF1!2000
          0000
          SIMVL1!2000
/
MESSS2,   SIMF2!2000
          0000
```

```
        SIMVL2!2000
SIMF1,  TEXT Z(INTO L S  PUT SAMPLE Z
L2,     TEXT Z FOR PARM )B1,VB1,( Z
        TEXT Z RAISE SENSE SWITCH 1); Z
/
SIMVL1, PARMID!2000
        TEXT Z0;Z
/
SIMF2,  TEXT Z(LOWER SENSEZ
        TEXT Z SWITCH 1);Z
/
SIMVL2, TEXT Z0;Z
AAZITB, NOP
        SEGMNT 1
        *2020
OLLOAD, STC 1
        IOB
        6234
        SCR 3
        BCL I
        7774
        ADA I
N600,   0600
        STC OELIF
        ADD 0
        STC OENTN
        ADD 1
        APO I
        JMP OSLOAD
/
OBLOAD, SET I 2
        OSBUFF-1!4000&5777
        JMP SBLOAD
OSLOAD,     SET I 2
OSTCKI, OSTACK-1!4000&5777
SBLOAD, LDA 1
        STA I 2
        LDA I 1
        STA I 2
        LDA I 1
        ADA I
        -1
        STC 5
LSLOOP, CLR
        LDA I 5
        SAE I
        TEXT Z0;Z
        JMP .+2
        JMP LSCON2
        STA
        6
        APO
        JMP BITSTR
        LDA 6
        APO I
        JMP LSCON1
        LDA I
        TEXT Z0-Z
        STH I 2
        LDA 6
        COM
LSCON1, SCR 1
        SET I 7
        -4
        SCR 10
        XSK I 7
        AZE
        JMP .+3
        LZE I
        JMP .+5
        BSE I
        0060
        STH I 2
        ROL I 7
        OAC
        XSK 7
        JMP .-14
/
        LDA I
COMMA,  TEXT Z0,Z
        STH I 2
        JMP LSLOOP
LSCON2, LDA
        2
        APO I
        JMP .+4
        LDA I
        TEXT Z0 Z
        STH I 2
```

```
     LDA I
          TEXT ZØ;Z
          STH I 2
          LDA
          1
          APO I
          JMP LSCON3
          LDA I
          OSBUFF!4000&5777
          STC 2
          JMP OCNTL3
/
LSCON3,   LDA
          2
          STA
          OSTCKI&5777
          COM
          ADA I
          OSBUFF!4000&5777
          APO
          JMP OCNTL2
          JMP OECON2
/
BITSTR,   LDA 6
          RCL I 5
          STC BITSAV
          LDA 5
LOOP2,    ROR I 1
          LZE I
          JMP LOOP3
          SRO
          BITSAV
N0016,    NOP
          JMP LOOP2
LOOP3,    LDA I
BITSAV,   NOP
          JMP LSCON1
/
/OUTPUT CONTROL ROUTINE
OCNTL1,   CLR
          IOB
          6234
          SCR 3
          RCL I
          7774
          ADD N600
          STC OELIF
          ADD Ø
          STC OERTN
          LDA
          OSTCKI&5777
          SAE I
          OSTACK-1!4000&5777
          JMP .+2
          JMP OECON2
OCNTL2,        LDA I
          OSTACK!4000&5777
          STC 2
OCNTL3,   LDA 2
          ADA I
          -1
          BSE I
          4000
          STC 3
          SRO I 2
          JMP .+2
          JMP OCRT
OTT,      LDA I
          OCNTL5!6000
          STC OUTRTN
          JMP TTYPE2
/
OCRT,     SET I 4
          -50
OCRT1,    SAM 7
          ADA I
          -1000
          STC 13
OCRT2,    XSK I 13
          JMP OCNTL4
          XSK I 4
          JMP OCRT1
          JMP OUTEND
OCNTL4,   LDA I
          114
          STC 1
          LDA I
          20
          STC 10
OCNTL5,   LDA
          3
          STC 5
          LDA
          2
          STC 6
/
SEARCH,   LDH I 5
          SHD I
          5000
          JMP STRING
          SHD I
          2600
```

```
        JMP VARVAL
        SHD I
        0200
        JMP BLANKS
        SHD I
        3000
        JMP XCOOR
/
        SHD I
    3100
        JMP YCOOR
        SHD I
        7300
        JMP ENDM
/
        SHD I
        1400
        JMP TTLINE
        SHD I
        0300
        JMP TTCARR
        JMP SEARCH
OUTRTN, NOP
STRING, LDA I
        6000!.+2
        STC OUTRTN
        LDH I 5
        SHD I
        5100
        JMP SEARCH
        SRO 2
        JMP TTYPE1
        JMP CRT
/
VARVAL, LDA I
        6000!.+2
        STC OUTRTN
        SET I 12
        -6
VLOOP,  XSK I 12
        LDH I 6
        SHD I
        5400
        JMP .+4
        SRO 2
        JMP TTYPE1
        JMP CRT
        XSK I 12
        JMP BLANKS+3
        JMP SEARCH
BLANKS, JMP NUMBER
        COM
        STC 12
        LDA I
        6000!.+7
        STC OUTRTN
BLOOP,  LDA I
        0040
        SRO 2
        JMP TTYPE1
        JMP CRT
        XSK I 12
        JMP BLOOP
        JMP SEARCH
XCOOR,  JMP NUMBER
    STC 1
        JMP SEARCH
/
YCOOR,  LDH I 5
        ROR 1
        STC SAVSIN
        JMP NUMBER
        SRO I
SAVSIN, NOP
        JMP .+2
        COM
        STC 10
TTLINE, SRO 2
        JMP .+2
        JMP SEARCH
        LDA I
        0212
        JMP TTOUT
        JMP SEARCH
TTCARR, SRO 2
        JMP .+2
        JMP SEARCH
        LDA I
        0215
        JMP TTOUT
        SET I 7
        -120
        JMP SEARCH
ENDM,   SRO 2
        JMP OUTEND
        JMP OCRT2
NUMBER, LDA
        0000
        STC NUMRTN
NUMLP,  ROL 3
        STC NUMSAV
        LDH I 5
```

```
          SAE I
          0054
          JMP .+5
          LDA
          NUMSAV
          ROR 3
          JMP NUMRTN
          BCL I
          7770
          ADA I
NUMSAV,   NOP
          JMP NUMLP
NUMRTN,   NOP
/TELETYPE ROUTINE
TTYPE1,   ADA I
          -37
          APO
          ADD N100
          ADA I
     37
          JMP TTOUT
          XSK I 7
          JMP OUTRTN
N100,     100
TTYPE2,   LDA I
          0212
          JMP TTOUT
          LDA I
          0215
          JMP TTOUT
          SET I 7
          -120
          JMP OUTRTN
/
TTOUT,    IOR
          6002
          PDP
          PMODE
          TSF
          JMP .-1
          TLS
          LINC
          LMODE
          IOR
          6001
TTRTN,    JMP 0
/CRT OUTPUT ROUTINE
CRT,      ROL 1
          ADA I
          CRTTAB&5777
          STC 11
          ADD 10
          DSC 11
          LDA
          10
          DSC I 11
          LDA I
          2
          ADA
          0001
          LDA
          1
          COM
          ADA I
          656
          APO I
          JMP OUTRTN
          LDA I
          114
          STC 1
          LDA I
          -20
          ADA
          10
          JMP OUTRTN
/
OUTEND,   LDA
          2
          COM
          ADA I
          OSBUFF+1!4000&5777
          APO
          JMP OECON2
          LDA I
          OSTACK-1!4000&5777
          STC 2
/
          LDB 6
          JMP .+2
          LDB I 6
          SAE I
          0073
          JMP .-3
OELOOP,   LDA
          6
          SAE
          OSTCKI&5777
          JMP .+2
          JMP OECON1
          LDA I 6
          STA I 2
          JMP OELOOP
```

```
/
OECON1, LDA
        2
        STC OSTCKI&5777
OECON2, DJR
OELIF,  NOP
OERTN,  NOP
CRTTAB, 0000            /NONE
        0000
        4477            / A
        7744
        5177            / B
        2651
        4136            / C
        2241
        4177            / D
        3641
        4577            / E
        4145
        4477            / F
        4044
        4136            / G
        2645
        1077            / H
        7710
        7741            / I
        0041
    4142            / J
        4076
        1077            / K
        4324
        0177            / L
        0301
        3077            / M
        7730
        3077            / N
        7706
        4177            / O
        7741
        4477            / P
        3044
        4276            / Q
        0376
        4477            / R
        3146
        5121            / S
        4651
        4040            / T
        4077
        0177            / U
        7701
        0176            / V
        7402
        0677            / W
        7701
        1463            / X
        6314
        0770            / Y
        7007
        4535            / Z
        6151
/
TRACE,  TRACEF&5777
        7777
        TRACVL&5777
TRACVL, TASKID!2000
        PARMID!2000
        STEPID!2000
        SECCLK!2000
        MINCLK!2000
        TEXT Z@;Z
        *CRTTAB+100
SPACE,  0000
        0000
TRACEF, TEXT Z(TRACE:)Z
LINET2, TEXT ZR2,(T)VR2,(P)VR2,Z
LINET3, TEXT Z(S)VB4,VZ
LINET4, TEXT Z(SEC)B2,V(MIN)CLZ
LINET5, TEXT Z; Z
/
        *CRTTAB+140
        4136            /0
        3641
    2101            /1
        0177
        4523            /2
        2151
        4122            /3
        2651
        2414            /4
        0477
        5172            /5
        0651
        1506            /6
        4225
        4443            /7
        6050
        5126            /8
        2651
        5120            /9
        3651
OSTACK, NOP
```

```
        *.+50
OSBUFF, NOP
        *.+30
ODFORM, TEXT Z( OUTPUT VALUE = )VZ
        TEXT Z( PARMID = )VCLB7,(TIME=Z
        TEXT Z)V( HOURS  )V( MINUTES  )Z
        TEXT ZV(SECONDS.)CL;Z
/
XSIDLE, XIDLEF
        0000
        XIDLEV
XIDLEF, TEXT Z(IDLE );Z
XIDLEV, TEXT Z0;Z
XTIMET, XFTIMT
        0000
        XVCOM1
XFTIMT, TEXT Z(TEST TIMER P)V;Z
XVCOM1, PARMID!2000
        TEXT Z0;Z
XSTART, XFSTRT
        0000
        XVCOM2
/
XFSTRT, TEXT Z(START T )V( AT POS )VB2,Z
        TEXT ZV;Z
XVCOM2, TASKID!2000
        POSID!6000
        7417
        POSID!6000
        7760
        TEXT Z0;Z
/
XPINT,  XFPINT
        0000
        XVCOM2
XFPINT, TEXT Z(PROG INT T )VZ
XFPIN2, TEXT Z( AT POS )VB2,V;Z
XPQLOD,     XFPQLD
        0000
        XVPQLD
XFPQLD, TEXT Z(SCHEDULE TASK )V;Z
XVPQLD, PQTASK!2000
        TEXT Z0;Z
/
XDELAY, XFDLAY
        0000
        XVCOM2
XFDLAY, TEXT Z(DELAY TASK )V;Z
AAZSOB, NOP
DTIMEQ, 7777
        *.+13
```

```
DCODEQ, NOP
        *.+13
PQUEUE, 7777
        7777
        7777
        7777
        7777
        7777
        7777
        7777
        7777
        7777
        7777
/ TASK TIME QUEUE
TTIMEQ, 1
        3
        5
        7
        11
        13
        7777
        NOP
TTTAB,  6
        10
        10
        10
        10
        10
        NOP
        NOP
TDTAB,  0020
        4040
        0060
        0100
        0140
        0160
        NOP
        NOP
/
MONTAB, 0402
        0
    0403
        0404
        0400
        0
        0
        0
        0
        0
        0
```

```
        0
        0417
        0
        0000
PARM2,  0422
        0
        0423
        0424
        0420
        0
        0
        0
        0
        0
        0
        0437
        0
        2400
PARM3,  0441
        0442
        0443
        0444
        0456
        0
        0
        0
        0
        0
        0
        0457
        0454
        0455
        0000
PARM4,  0461
        0462
        0463
        0464
        0476
        0
        0
        0
        0
        0
        0475
        0477
```

```
          0474
          0000
PARM5,    0501
          0502
          0503
          0504
          0516
          0500
          0
          0
          0
          0
          0
          0
          0517
          0514
          0515
          2520
PARM6,    0521
          0522
          0523
          0524
          0525
          0536
          0
          0
          0
          0
          0
          0
          0537
          0534
          0535
          0000
PARM7,    0541
          0542
          0543
          0544
          0545
          0540
          0
          0
          0
          0
          0
          0
          0
          0557
          0
          0000
PARM8,    0561
          0562
          0563
          0564
          0565
          0560
          0
          0
          0
          0
          0
          0
          0
          0440
          0
          0
STEPTB,   6000!SUBCLL
          6000!SDSAVE
          6000!TESTSD
          6000!SUBCLL
          6000!SUBCLL
          6000!DELAY
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          6000!INITRC
          6000!INITC1
          6000!REPEAT
          6000!LEND
SBLTAB,   0275
          NOP
          NOP
          0276
          0277
          /
SBMTAB,   7700
          0000
          0000
          7700
          7700
SDBUFF,   NOP
          *.+20
LIMTAB,   1731
          1733
          1000
          1020
          1310
          1322
          0670
```

```
          0702
          0100
          0105
          0100
          0105
          -105
     -100
          0670
          0702
SKIPTB,   0015
          0035
          0055
          0074
          0115
          0135
          0155
          0175
/DELAY ROUTINE DELAY TABLE
DTAB1,    0
          0
          0
          0
          2
          2
          2
          2
RPTAB,    NOP
          NOP
          0040
          0060
          0105
          0120
          NOP
          NOP
RCTAB,    0
          0
          10
          10
          10
          10
          0
          0
RLSTAB,   0
          0
          10
          10
          10
          10
          0
          0
RLDTAB,   0
          0
          2
          2
          2
          2
          0
          0
INRTAB,   0
          0
          10
          10
     10
          10
          0
          0
CVBUFF,   NOP
          *.+7
CABUFF,   0000
          0
          0
          0
          0
          0
          0
          0
AAZTAB,   NOP
/
          TSOUT=7777
          SEGMNT2
          *20
HSCREG,   NOP
          CLR
          COM
          STC HSCREG
          IOB
          6002
          LDA I
          0215
          PDP
          PMODE
          TLS
          KCC
          LINC
          LMODE
          SNS 3
          JMP .-1
          CLR
          SCR 14
          LDF 0
          IOB
          6304
```