DESIGN AND IMPLEMENTATION OF AN INPUT/OUTPUT SCHEDULER FOR THE TIME-SHARING SYSTEM OF THE GENERAL ELECTRIC CORPORATE RESEARCH AND DEVELOPMENT CENTER

by

MICHAEL B. RUBENS

Bachelor of Engineering Project Report

June 1972

THAYER SCHOOL OF ENGINEERING
DARTMOUTH COLLEGE
HANOVER, NEW HAMPSHIRE

APPROVED: Miles V. Hayes
Miles V. Hayes

This research, supported by the Advanced Research Projects Agency of the Department of Defense, was monitored by the Air Force Office of Scientific Research under Contract No. F4Y620-68-C-0015.

THAYER SCHOOL OF ENGINEERING
DARTMOUTH COLLEGE

DESIGN AND IMPLEMENTATION OF AN INPUT/OUTPUT SCHEDULER FOR THE TIME-SHARING SYSTEM OF THE GENERAL ELECTRIC CORPORATE RESEARCH AND DEVELOPMENT CENTER

by

MICHAEL B. RUBENS

BACHELOR OF ENGINEERING

June 1972

ABSTRACT

The problem is to design and implement an Input/Output Scheduler for the General Electric Corporate Research and Development Center. Given the Center's current time-sharing environment of master and slave modes, a slave mode scheduling system is proposed. This system is composed of two distinct levels: a monitor, which handles all external input/output and scheduling, comprises the upper level; the lower level contains all the peripheral driver modules, which, while also operating in slave mode, transfer the data to/from such peripheral devices as line printers and card punches. Just such a system has been successfully written and is operating on the Center's computer system.

ACKNOWLEDGMENTS

I would like to express my thanks to my advisor, Professor Miles V. Hayes, who has been most generous with his help and patience during the past two years. His interest and inspiration have contributed greatly to the success of the project. The author wishes to thank Dan Berry, who proved to be a constant source of suggestions, inspirations, and proofs.

I also wish to thank the General Electric Research and Development Center for providing the facilities and the generous support necessary to carry on this type of project report.

The support of this project by the Advanced Research Projects Agency of the Department of Defense is gratefully acknowledged. The research was monitored by the Air Force Office of Scientific Research under Contract No. F4Y620-68-C-0015.

TABLE OF CONTENTS

TABLE OF ILLUSTRATIONS

INTRODUCTION

The purpose of this paper is to provide an overview of an Input/Output scheduling system which was designed and implemented at the General Electric Research and Development Center during the period June 1971 to September 1971. The object of this work was to build a software operating system which while running under the Center's current time-sharing system would:

(1) decrease the amount of code that must be core resident;
(2) increase absolute number and types of I/O devices that could be brought on-line simultaneously; (3) use the existing interfaces; (4) allow the operator via a command language to bring I/O devices arbitrarily on/off line; (5) and allow for experimentation and debugging of new I/O devices.

This project report is divided into three major sections. The first provides an overview of the hardware, the current operating executive system, and the old Listener structure. The second section deals with the overall design of the new Listener structure (i.e. the Input/Output Scheduler). Lastly, the third part gives a detailed view of the internal structure of the Monitor and an I/O driver prototype.

CHAPTER I

SYSTEM OVERVIEW

I.1 Hardware

The hardware configuration is as shown in Figure 1. The basic system consists of a single processor, a real time - I/O controller, and two 64K memory modules. The central processor is a GE 605, which has four base address registers, (BAR), instead of the one as in the GE 635. These provide memory protection, automatic relocation, and optional write inhibit. Each register has two fields, one which denotes the origin of the current program in core and the other which denotes its length. The hardware automatically checks all logical addresses produced by the processor (when operating in slave mode). A process (program in execution) can thus consist of up to 4 physically disjoint segments (which all must be in core when the process is in execution). Since the address field is 18 bits long and the 2 high order bits are used to designate the BAR, a segment has a maximum length of 2↑16 words (65,536). The write inhibit bit, which is associated with each base register, can be used to prevent modification of a segment. This is particularly useful in implementation of pure procedure. The processor operates in either of two modes: master or slave. In addition, certain privileged instructions (for performing I/O and manipulating BAR's) can only be executed in master mode.

The RT-IOC serves as the input-output interface for the system. This device is capable of transmitting data in an asynchronous manner between core memory and up to 32 peripheral devices.

The memory is composed of two memory controllers and 64 K of 36-bit (plus 1 parity bit) magnetic core storage with a 1.0 microsecond read-restore memory cycle time per controller.

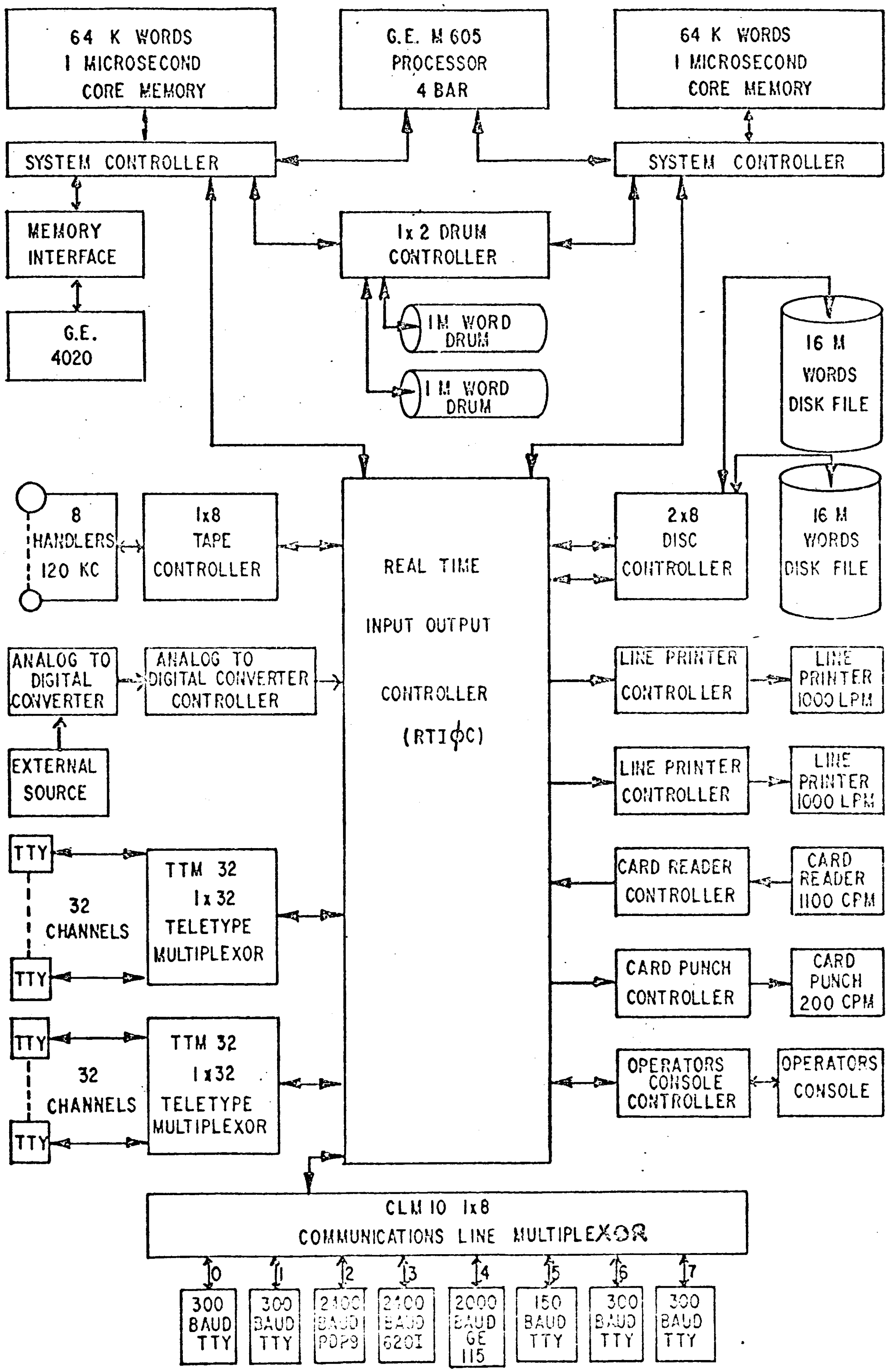

64 K WORDS
1 MICROSECOND
CORE MEMORY
G.E. M 605
PROCESSOR
4 BAR
64 K WORDS
1 MICROSECOND
CORE MEMORY
SYSTEM CONTROLLER
SYSTEM CONTROLLER
MEMORY
INTERFACE
G.E.
4020
1x2 DRUM
CONTROLLER
1M WORD
DRUM
1 M WORD
DRUM
16 M
WORDS
DISK FILE
8
HANDLERS
120 KC
1x8
TAPE
CONTROLLER
REAL TIME
INPUT OUTPUT
CONTROLLER
(RTIOC)
2x8
DISC
CONTROLLER
16 M
WORDS
DISK FILE
ANALOG TO
DIGITAL
CONVERTER
ANALOG TO
DIGITAL CONVERTER
CONTROLLER
LINE PRINTER
CONTROLLER
LINE
PRINTER
1000 LPM
EXTERNAL
SOURCE
LINE PRINTER
CONTROLLER
LINE
PRINTER
1000 LPM
TTY
32
CHANNELS
TTY
TTM 32
1x32
TELETYPE
MULTIPLEXOR
CARD READER
CONTROLLER
CARD
READER
1100 CPM
CARD PUNCH
CONTROLLER
CARD
PUNCH
200 CPM
TTY
32
CHANNELS
TTY
TTM 32
1x32
TELETYPE
MULTIPLEXOR
OPERATORS
CONSOLE
CONTROLLER
OPERATORS
CONSOLE
CLM 10 1x8
COMMUNICATIONS LINE MULTIPLEXOR
0
1
2
3
4
5
6
7
300
BAUD
TTY
300
BAUD
TTY
2400
BAUD
PDP9
2400
BAUD
6201
2000
BAUD
GE
115
150
BAUD
TTY
300
BAUD
TTY
300
BAUD
TTY

Figure 1

I.2 Executive system

The exec is designed to perform two major functions (see Appendix I) [6]. First of all, it provides a set of primitives accessible to all users which serve to enhance the hardware instruction repertoire of the machine. Thus the user sees a pseudo machine with extended capabilities. It is convenient to divide the primitive commands into the following three categories:

1. File manipulation (I/O commands) -- primitives for the reading, writing, appending, and scratching of either sequential or random access files. These enable the user to perform input/output operations which are expressed in terms of logical file parameters. Thus the user is shielded from the intricacies of the actual hardware device being dealt with and the actual physical location of the file.

2. Process manipulation (control commands) -- primitives for creating, terminating, and blocking processes, setting up fault handling modes, for making memory requests and readjusting base address register settings.

3. Directory manipulation (file and event commands) -- primitives for creating and destroying directories and entries within directories and for modifying and reading information within entries. The file commands provide the user with an interface to the file system. They allow him to open cataloged and scratch files, to catalog a file or directory, to unsave a file or directory. The user is thus allowed to create a file structure and to manipulate it within this structure.

The second major function performed by the exec is the allocation of resources to the active processes in the system demanding service. These resources include central processor time, core space, space on mass storage devices and use of peripherals. The exec takes into account the current utilization of the system resources as well as the priority and history of the various processes demanding service in making decisions in this area.

The exec itself is divided into two sections: the Master Mode Executive and the Slave Mode Executive (see Figure 2). The Master Mode Executive is the only portion of the entire Executive that executes code in master mode. The Slave Mode Executive operates in slave mode. The Master Mode Executive can be categorized by its three principal functions:

1. The execution of system input/output primitive commands -- the code for initiating data transfers in contained in the master exec. It translates these primitive commands, which are expressed in terms of logical parameters, into commands recognized by the hardware. The major reason for this is that I/O transactions must use absolute core addresses. Such information is invisible to code which is executed in slave mode where addresses are logical, relative to some BAR. A further reason is that the instruction for initiating a transfer can only be executed in master mode. This prevents slave processes from directly initiating their own transactions, a prohibition which is indispensible when storage on peripheral devices is shared by many users. In addition to servicing the interrupts generated, the Master Mode Executive returns to the user the physical and logical status of the operation.

2. The running of processes -- code in the Master Exec is also responsible for setting up and transferring control to slave processes. The reason for this is that the instructions for loading BAR's can only be executed in master mode. Such a restriction is necessary in a system in which several processes must coexist in core.

3. Fault handling, disk and drum allocation, and hardware malfunction servicing -- master mode is entered in one of two ways: either by interrupts or by faults. In contrast to interrupts, which are caused by signals from peripheral devices, faults are caused by the process in control when the fault occurred (e.g. accumulator overflow, illegal instruction, illegal memory reference, etc.). If the process has previously indicated its desire to handle such faults,

control is returned by the master exec to the fault handling code of the slave process responsible for the fault. If the process has not set up its own fault handling mechanisms, the process is terminated when such a fault occurs.

A fault may also be caused by a slave process by executing a master mode entry (MME) instruction. Such an occurrence indicates that the slave process wishes the exec to execute a primitive on its behalf. The exec determines the identity of the primitive as well as its parameter by examining the contents of the registers at the time the MME occurred. Those primitives dealing with file manipulations are handled by the exec's primitive handler. All other primitives are passed on to the slave exec.

The Slave Mode Executive can likewise be categorized by its three principal functions:

1. The allocation of system resources -- it is its responsibility to allocate the central processor and memory to the slave processes. The slave exec must also make decisions concerning the allocation of certain peripheral devices (e.g. mangetic tapes) to slave processes. The only exception to the allocation of system resources is the space on the shared mass storage device, which is allocated by the master exec.

2. Scheduling and swapping -- the slave exec determines core utilization and process swapping. This also includes (process termination).

3. Directory and process primitives -- directory and process manipulation primitives are passed on to the slave exec by the master exec. The slave exec then executes the process and file system primitives and it maintains the file system catalog (directory). It is the slave exec that has the real control over the operating system.

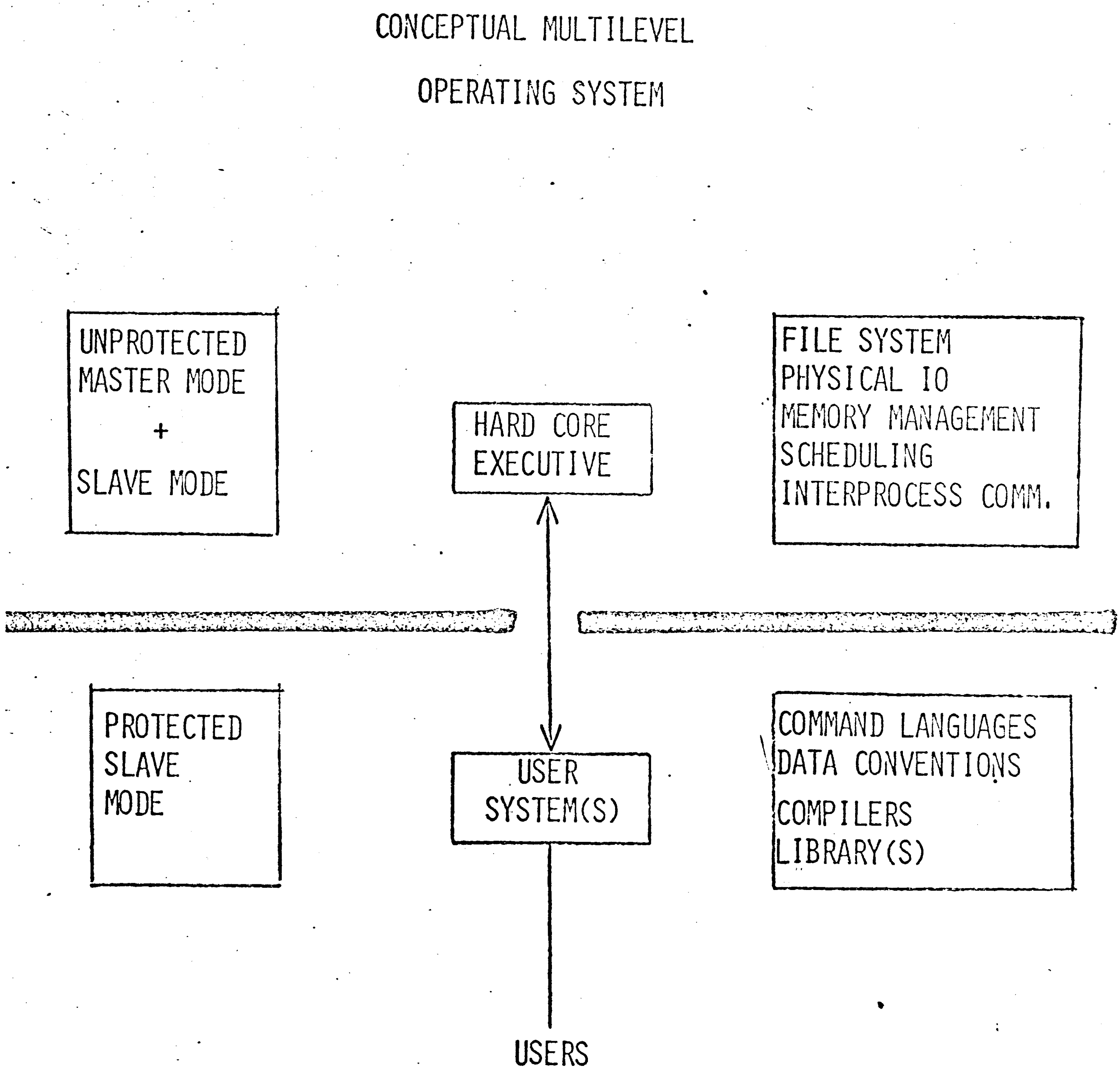

CONCEPTUAL MULTILEVEL
OPERATING SYSTEM
UNPROTECTED
MASTER MODE
+
SLAVE MODE
HARD CORE
EXECUTIVE
FILE SYSTEM
PHYSICAL IO
MEMORY MANAGEMENT
SCHEDULING
INTERPROCESS COMM.
PROTECTED
SLAVE
MODE
USER
SYSTEM(S)
COMMAND LANGUAGES
DATA CONVENTIONS
COMPILERS
LIBRARY(S)
USERS

Figure 2

I.3 Old Listener Structure

The Listener is the one slave process that is always in the system and it plays a rather unique role [8]. As can be seen from Figure 3, the Listener is the most important slave process of the multi-level job structure. In some senses, it is an extension of the Executive. Because of its functional responsibilities, which include the handling of all teletypes and communications files, the validation of all user numbers, the spawning of all other slave processes, the collection of system statistics, and the handling of the system output to such devices as the line printer and card punch, the Listener must be "booted-in" with the Executive as part of the system start-up deck.

In order to better describe the relative importance of the Listener, it is most convenient to discuss the Listener by dividing it into the following four functional categories:

1. The handling of teletypes and communications files -- this category or functional area is itself composed of three sub-areas. On the physical level, the Listener has the responsibilities of answering a ringing phone; hanging-up the phone after a sign-off; and taking care of a disconnect. On the message level, the Listener handles all panic stops (i.e. breaks) such editing as character or line deletions for teletype input; and the transferring of data between teletype and the slave process wishing to communicate via the teletype.

   On the operator level, the Listener is in charge of sending all operator messages and warnings to the teletype users. Due to the Listener's control overall teletype, the operator can selectively enable or disable any and all teletypes.

2. System statistics and accounting -- by the very nature of its position in the job structure hierarchy, the Listener is the most logical process to collect system statistics (e.g. the number of teletype connections, the number of characters input and output,

the maximum number of users, the number of communication file reads and writes, and the number of processes spawned). Since the Listener handles the validation of user numbers, it likewise creates the accounting blocks for user billing.

3. Spawns other slave processes -- it spawns other slave processes as they are required and passes on to them the parameters they need. Hence all processes are descendents of the Listener.

4. SYSOUT processing -- SYSOUT (i.e. System Output) processing includes the printing and punching of output generated by any slave process. A slave process which has created data to be outputted, can do so by copying that data into a cataloged file (or by copying the data into a scratch file and then cataloging the file [2]. Next the slave process opens the system file 'PRINT-FILE-QUEUE' if the data is to be printed on the high-speed printer or opens the system file 'PUNCH-FILE-QUEUE' if the data are to be punched. After a successful open, the slave process appends a 64 word descriptor to the appropriate file. The descriptor contains the complete tree-name of the file to be outputted and identifier bits (e.g. the bits describe the format of the data and whether or not a header has been supplied). Via the event structure [1], the slave process 'causes' the appropriate system event (either PRINT-FILE-EVENT or PUNCH-FILE-EVENT). The Listener is then 'notified' via the event mechanism. It responds by reading the appropriate 'FILE-QUEUE', opening the data file, and outputting it to the proper device. This assumes that the device is currently inactive. If the peripheral device is busy, the Listener simply does nothing for the moment. When it finishes the current data file, it will check to see if there are any more des-

criptors since it last looked. If so, it will start the outputting of the next data file. Although it is possible for any slave process to have a data file punched or printed, the slave process may never access the peripheral device which receives the data. The Listener in addition to the one line printer and card punch also owns the card reader.

OLD LISTENER STRUCTURE

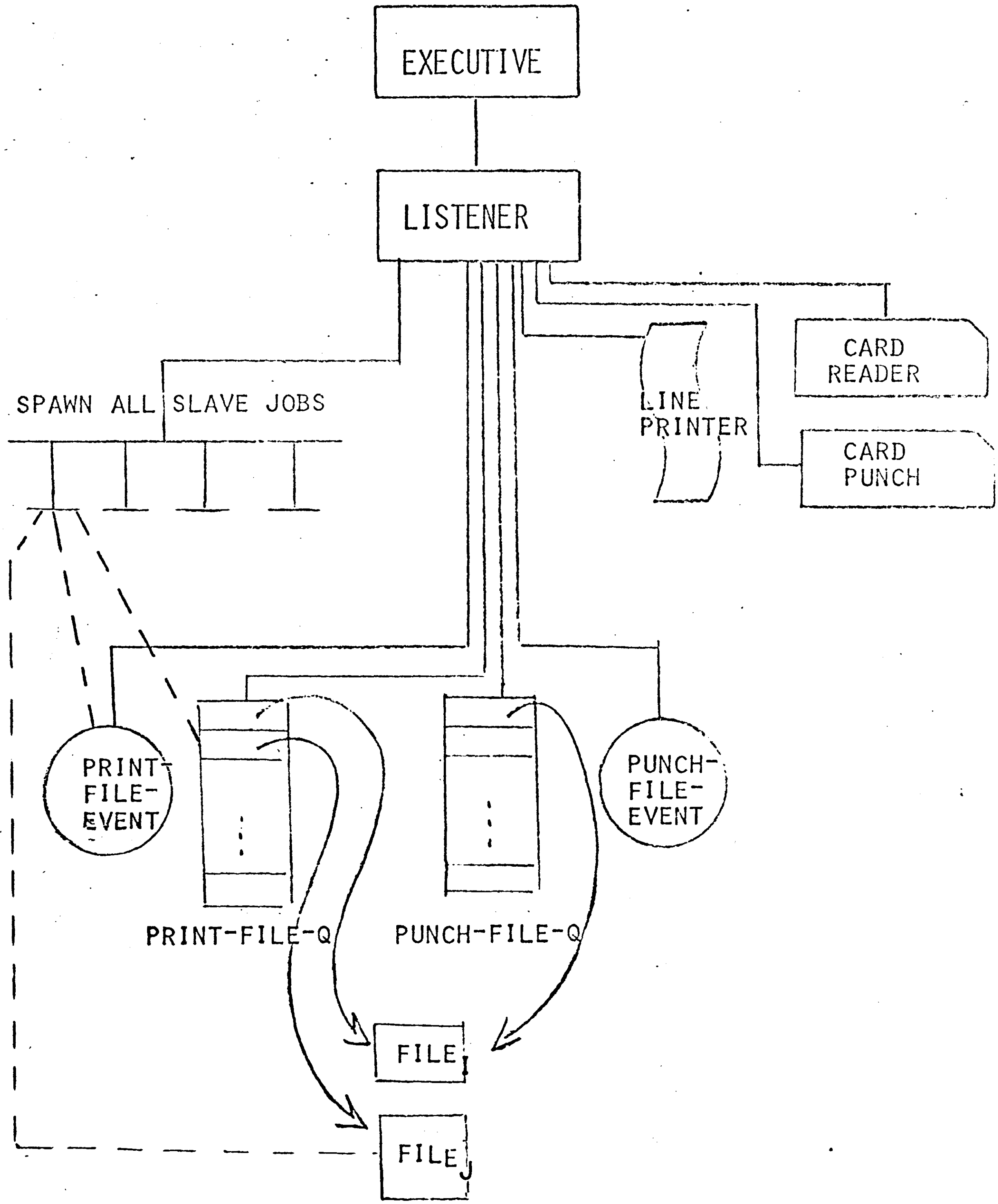


Figure 3

CHAPTER II
NEW LISTENER DESIGN

II.1 Problem Statement

II.2 Limitations of the Old Listener Design

II.3 Design Criteria

II.4 System Design

II.1 Problem Statement

Given the General Electric Research and Development Center's time-sharing system with its hierarchial job and file structure as the programming environment, the problem is to design and then implement in assembly language an Input/Output Scheduler system, which would assume the Listener's current functional duties of card punching, card reading and line printing, and which would increase the utilization of the system resources (e.g. line printer, card punch, core) while leaving as much of the existing support interfaces unaffected.

II.2 Limitations of the Old Listener Design

The Listener, as originally designed, probably served well in its apprentice years. But just as the Executive system has grown to expand its capabilities, the Listener has been forced to grow. Growth in an assembly language program usually means modifications of the existing code and especially additional code to handle the new or unthought of situations. Thus growth (or attempted extra added responsibilities) points out the Listener's three major inter-related limitations with regard to peripheral device I/O scheduling:

1. Core resident -- the Listener, as mentioned earlier, is a core resident module. It is currently 18 1/2 K in size (45000 octal words). Any additions which would obviously increase its size would do so only at the expense of the core designated for slave processes since the additional code would also become core resident. Since the Listener must be core resident it must be made as small as possible by removing non-essential code. The code which handles the line printer, card punch and reader is non-essential.

2. Coded for one peripheral device of each type -- the Listener was coded to handle only one line printer and one card reader and one card punch. Although the Center has two line printers, only one can truly be on-line. This is an obvious waste of system resources. The Listener could be rewritten to handle the second line printer, but this would just increase the amount of code that would then become core resident. This would not answer the problem if yet another line printer was purchased or another card punch.

3. Debugging and experimentation extremely limited -- since the Listener controls all teletypes and spawns all slave processes, any debugging must be done on a dedicated system.

This necessarily limits the amount of time that can be spent trying out modifications. For to crash the Listener is to crash the entire system.

II.3 Design Criteria

The software system to be designed is defined by the problem statement. The assembly language program(s) must run under the General Electric R & D time-sharing environment. The new system must handle Input/Output scheduling formerly handled by the Listener and it should utilize the existing interfaces as much as possible. The criteria that this system must meet are the following:

1. By decreasing a functional responsibility of the Listener, the amount of code that must be core resident must be likewise decreased. And if possible, the total amount of code to handle the peripheral devices should be kept to a minimum, thus freeing core for the other slave processes.
2. The number and type of I/O devices that could be brought on-line simultaneously should be considered arbitrary. The code must be general enough to handle such additions as a third line printer or graphics plotter.
3. The existing software interfaces should be used as much as possible in order to minimize the amount of new code that need be written. This also minimizes the amount of time spent debugging the new system.
4. The operator through some command language must be able to bring I/O devices arbitrarily on/off line. This is necessary in order to handle hardware failures. Also through this command language, the operator must be allowed to re-start or stop the output on any of the I/O devices.
5. The new system must allow for experimentation and debugging of new I/O devices without danger of crashing the entire system. If possible all debugging should be able to be done on-line in order to give the programmer the greatest amount of on time.

II.4 System Design

The development of this Input/Output Scheduling system was evolutionary (rather than a selection of one from a number of alternatives), and only the final result will be described here (see Figure 4), although justifications will be given where possible. To satisfy the first criterion stated above, it was decided that the system should be separated into two major functional areas. The first included such functional duties as the handling of all external input/output, which is composed of slave process and operator requests, the pre- and post-processing of a request, billing, and the scheduling of requests. The second was comprised largely of the transferring of data from the file to the appropriate I/O device. This separation was accomplished by coding the two areas as two distinct prototype modules -- a Monitor and a peripheral driver prototype. This method minimizes at all times the amount of code that must be in core at any given time. For the Monitor need be in core only for a few milleseconds to handle a request. On the other hand, a particular peripheral driver, such as the line printer module, may be transferring data for a length of time as small as a few seconds to as much as an hour or more. Since the two functional areas are independent, the amount of core that is tied up is minimized if they are coded separately. Since any module that is not busy is legible to be swapped out of core.

The second criterion is easily met. The fact that the entire system was modularized allows for an arbitrary number and type of I/O devices that could be brought simultaneously on/off line. The Monitor has a complete set of assembly time parameters (i.e. MACROS) which allow for the definition of an arbitrary number and type of I/O devices. It has a command language via which the operator may selectively bring a particular device on or off line. Also the peripheral driver prototype may be modified (e.g. character conversion tables) to handle the new device or through an assembly time parameter handle an additional device of an already defined type or new type.

The third criterion simply requires that the old interfaces and conventions are retained. Since the author had no objections to or improvements for the old methods of interfacing, these were retained unmodified. The fourth criterion has already been explained. There is a command language via which the operator is able to bring I/O devices arbitrarily on/off line.

It is the fifth criterion that sums up the appropriateness of the modular system design. Since the Monitor is just another slave process spawned by the Listener, the programmer is allowed to spawn more than one of these at a time. Through the use of the command language, the programmer can arbitrarily assign the various I/O devices to the currently running Monitors. Thus while the current version of the Monitor is running, an experimental version may be spawned to test out some new feature. If the regular or experimental version of the Monitor should crash, the rest of the system and users would be unaffected. Likewise new peripheral driver prototypes may be tested in a similar fashion.

NEW LISTENER STRUCTURE

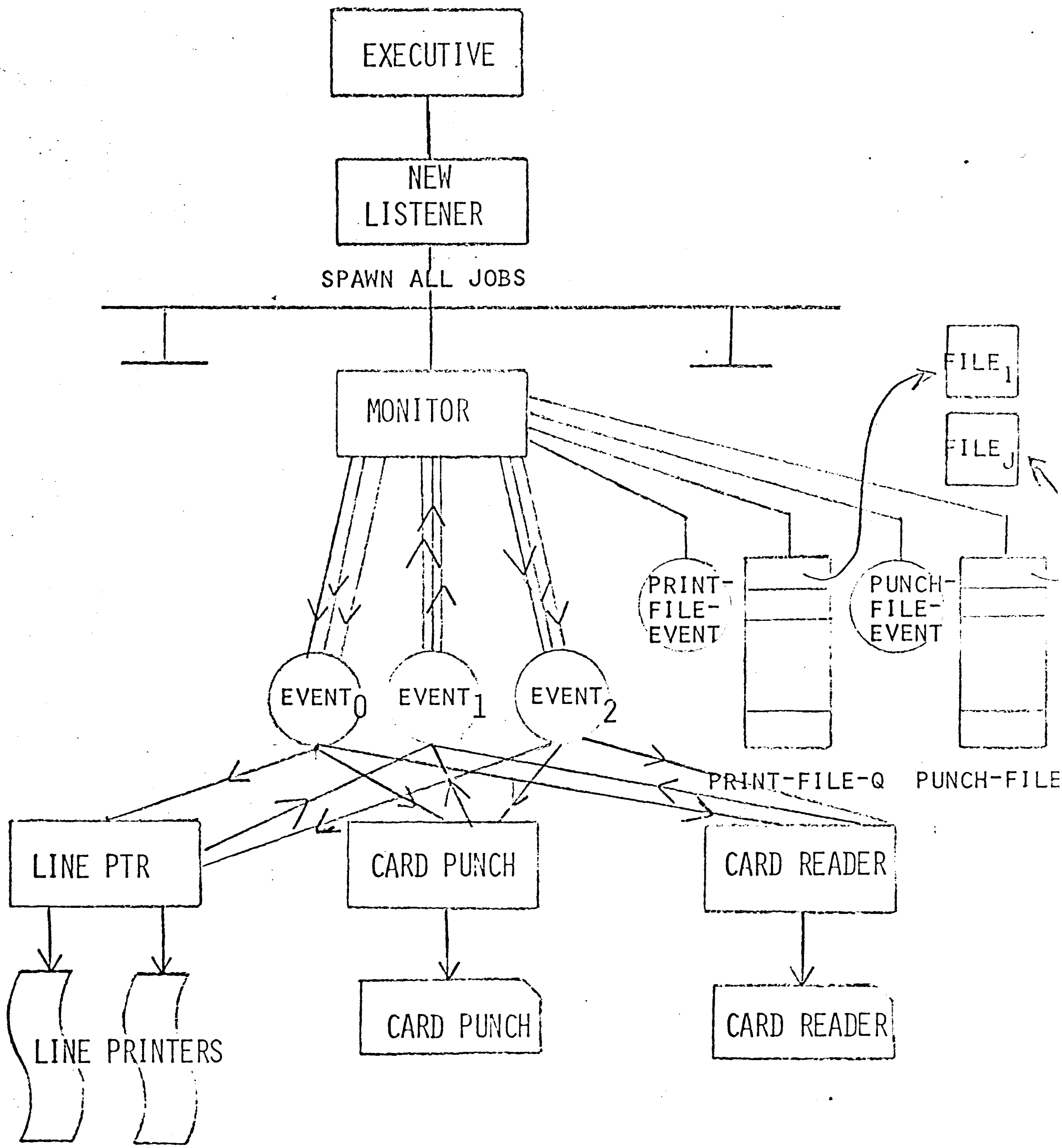


Figure 4

Before discussing the coding in detail of the new Listener structure, it is felt that a more general discussion of the organization of both the Monitor and the peripheral driver prototype would be most enlightening. Although not a design criterion, all modules written were designed to be multi-programmed [3]. Each module is therefore internally organized around a set of queues. Generally speaking, each resource has associated with it a queue on which are put all tasks currently requesting that resource. In a more general sense, the queues are used for synchronizing and communicating between routines.

These routines are in essence a set of algorithms. The Monitor is composed of five major algorithms (see Figure 5); the peripheral driver prototype, four algorithms (see Figure 11). An algorithm, as defined by Trakhtenbrot in Algorithms and Automatic Computing Machines, is a list of instructions specifying a sequence of operations which will give the answer to any problem of a given type. The important point to note is that the list of instructions is never modified. Hence two or more persons or processes can be executing the same algorithm simultaneously (i.e. multiprogramming). The only thing that can be modified is the individual data area of each process. This now allows for the definition of a task (sometimes referred to as a process) [5]. A task is represented by a pair of words. The first is called the instruction pointer, or IP, which points to the next instruction in the algorithm list to be executed. The second word points to the task's data area. It is called the environment pointer, or EP.

What follows is now that generalized discussion of the major algorithms first in the Monitor and then in the peripheral driver prototype. The algorithms are described in a pseudo-algol-like language to help the reader conceptualize the process.

MAJOR ALGORITHMS OF MONITOR

1) EXTERNAL INPUT
2) SCHEDULER
3) RUN-SERVICE
4) NOTIFY-SERVICE
5) TERMINATION

Figure 5

MONITOR

1) External input algorithm

The External Input algorithm is executed on every interrupt from either a slave process requesting a data file to be outputted or the operator via the command language. At initialization time for the Monitor, a task is created for each system event (e.g. PRINT-FILE-EVENT, PUNCH-FILE-EVENT, or OPERATOR-EVENT) to execute this code. The normal state of each task is to be blocked waiting for an interrupt, or wake-up, signal (see Figure 6, L1). Upon receipt of a signal, the task receiving the interrupt resumes at line L2. The conditional statement tests whether or not the Monitor owns any resources associated with the interrupt request. If not, there is no need to read the request because it cannot be handled now -- no resources at all -- so the task goes blocked again. Hence there is no reason to start the processing of the request, which would only tie up core. This is a safety feature to handle mechanical failures.

If the Monitor does own at least one of the resources requested, the execution proceeds. A job control block is allocated, as well as a read buffer, to hold the slave process 64 word descriptor. The descriptor is then read into core. If the file containing the descriptor was not empty (not End-Of-File), then the job control block is filled in as described by the descriptor. A statistics counter is incremented. The read buffer is released, and the request is then placed on the appropriate waiting, or input, queue. Since a single slave process may place more than one request at a time, the whole list of instructions is repeated until the end of the request file is reached. When this condition occurs, both the read buffer and the job control block are released.

The BRANCH statement is the most interesting statement of the entire algorithm. It is through this instruction that the currently executing task creates a second task. The second task has its IP pointing to the Scheduler algorithm and its EP pointing to some allocated memory. This

1) EXTERNAL INPUT (A PERPETUAL PROCESS)

```
L1:  BLOCK(EVENT_I,______)
L2:  IF RESOURCE_I = 0 THEN L1 ELSE
        BEGIN
            ALOC(JOB CONTROL BLOCK)
            ALOC(READ BUFFER)
            READ(FILE_I , (ALOC(READ BUFFER)), STATUS)
            IF STATUS = EOF THEN
            BEGIN
                  DALOC(READ BUFFER)
                  DALOC(JOB TASK BLOCK)
                  BRANCH(SCHEDULER)
                  GO TO L1
            END
            ELSE
            BEGIN
                  CREATE(JCB)
                  COUNTER_I := COUNTER_I + 1
                  DALOC(READ BUFFER)
                  QUEUE(JCB,INPUT-Q_I)
            END
            GO TO L2
        END
```

Figure 6

task is then placed on the queue associated with the processor. When the currently executing task blocks, the next waiting task on the processor queue is then started. After creating the new task, the current task blocks waiting for another interrupt signal.

2) Scheduler algorithm

The scheduler algorithm is executed by a created task (see Figure 7). The task simply checks each waiting task in all input queues to determine if that waiting task may be started. If not, it merely steps to the next item. If so, the peripheral resources required by that task are allocated to it; the task is removed from the queue; its IP is set to the Run-service algorithm; and it is placed on the processor queue. After stepping through all input queues, this task simply terminates (i. e. transfers control of the processor to the next waiting task without scheduling itself for a restart).

3) Run-service algorithm

Likewise the run-service algorithm is executed by a created task (see Figure 8). All this task need do is pass the proper information to the correct peripheral driver submodule. It is accomplished via the event mechanism. The conditional tests whether or not the submodule received the message. If not, it is re-transmitted. This created task in turn creates another task to wait for the reply from the submodule (IP points to notify-service algorithm). Hence the current task terminates after the BRANCH statement.

4) Notify-service algorithm

This algorithm specifies that the executing task alter its restart address (future IP) to point to the termination algorithm and to wait for an interrupt signal from the submodule which just got passed the information mentioned above (see Figure 9). Thus when the submodule signals a completion of the data transferring, the task will then execute the termination code for that request.

2) SCHEDULER (A CREATED PROCESS)

```
FOR I = 1 STEP 1 UNTIL (# OF QUEUES) DO
BEGIN
     FOR J = 1 STEP 1 UNTIL LEN(Q-LIST_I) DO
     BEGIN
          IF RUNABLE(Q-LIST_I, ELEMENT_J) THEN
          BEGIN
               ALOCRES(Q-LIST_I, ELEMENT_J )
               DEQ(Q-LIST_I, ELEMENT_J)
               RESTART = RUN-SERVICE
               QUEUE(Q-LIST_I, Q$TASK)
          END
     END
END
EXIT
```

Figure 7

3) RUN-SERVICE (A CREATED PROCESS)

```
L1: CAUSE(EVENT_I,STATE_J,MESSAGE_K,FILE_L,NUMBER,ACCESS,STATUS)
    IF STATUS(NUMBER) = 0 THEN L1 ELSE
    BEGIN
    BRANCH(NOTIFY-SERVICE,EVENT_I,STATE_J,TERMINATION)
    END
    EXIT
```

Figure 8

5) Termination algorithm

The execution of this algorithm is contingent upon the interrupt signal sent from a submodule via the event mechanism. (See Figure 10). The signal contains coded information. If the status of the signal is bad, then some error processing must proceed. Otherwise a read buffer is allocated; the disposition as specified in the 64 word descriptor is then acted upon; the buffer released. The counter is decremented by one and tested. If zero, then a task is created to attempt to scratch the request file. The resources allocated to this task are then released; another task is created. Since some resources have been released, a task is created to execute the scheduler algorithm. And finally the current task terminates.

4) NOTIFY-SERVICE (A CREATED PROCESS)

RESTART = TERMINATION

$\text{NOTIFY}(\text{EVENT}_I, \text{STATE}_J)$

EXIT

Figure 9

5) TERMINATION (A CREATED PROCESS)

```
IF STATUS = BAD THEN ERRORCHECK ELSE
BEGIN
ALOC(READ BUFFER)
DISPOSITION(FILE_I ,JCB)
DALOC(READ BUFFER)
IF (COUNTER_I := COUNTER_I - 1) = 0 THEN BRANCH(SCRATCH,FILE_I)
DALOCRES(JCB)
BRANCH(SCHEDULER)
EXIT
END
```

Figure 10

PERIPHERAL DRIVER PROTOTYPE

1) External input algorithm (see Figure 12)

This algorithm is not to be confused with the external input algorithm of the Monitor. It is called external because the inputs come external to the module, but they come from the Monitor only. At initialization time for the submodule, a task is created to execute this code. Its normal state is blocked waiting for a message from the Monitor. Upon receipt of any message, it must differentiate the information between a true message and a command. Commands deal with the acquiring and relinquishing of a peripheral device. Messages announce that new data are to be handled. If it is new data, a job control block is allocated and filled in. Then two asynchronous tasks are created to read the data from the data file and transfer it to the I/O device. Then the current task goes blocked waiting for another message from the Monitor.

2) Read task algorithm (see Figure 13)

The algorithm is composed of two other algorithms -- P and V (see Figure 15)[3]. It locates an empty buffer; fills it from the data file; and marks it full and ready to be written out to the I/O device. The filling of empty buffers continues until the data file has been exhausted. The task then simply terminates (EXIT).

3) Write task algorithm (see Figure 14)

This algorithm is the complement of the read task algorithm. It locates full buffers; empties them to the peripheral device; marks them empty and continues until there are no more buffers to empty. At that point it releases all resources allocated to this task and creates a task to send a termination message back up to the Monitor. The task itself terminates.

MAJOR ALGORITHMS OF PERIPHERAL DRIVER PROTOTYPE

1) EXTERNAL INPUT
2) READ TASK
3) WRITE TASK
4) TERMINATION

Figure 11

```
1)  EXTERNAL INPUT (A PERPETUAL PROCESS)
    BLOCK(EVENT_I,STATE_J)
    IF MESSAGE = COMMAND THEN GO TO COMMAND-SERVICE ELSE
    BEGIN
         ALOC(JCB)
         CREATE(JCB DESCRIPTOR)
         BRANCH(READTASK,JCB)
         BRANCH(WRITETASK,JCB)
    END
    GO TO EXTERNAL INPUT
```

Figure 12

```
2)  READ TASK  (A CREATED PROCESS)
      P(EMPTY,JCB)
      FILL(ALOC(READ BUFFER),READ,STATUS)
      V(FULL,JCB)
      IF STATUS = MORE THEN GO TO READ TASK ELSE EXIT
```

Figure 13

```
3)  WRITE TASK  (A CREATED PROCESS)
    P(FULL,JCB)
    EMPTY(WRITE,DALOC(READ BUFFER),STATUS)
    V(EMPTY,JCB)
    IF STATUS = MORE THEN GO TO WRITE TASK ELSE
    BEGIN
         DALOCRES(JCB)
         BRANCH(TERMINATION,STATE_I,MESSAGE_J)
    END
    EXIT
```

Figure 14

```
P(SEMAPHORE)

        SEMAPHORE = SEMAPHORE - 1
        IF SEMAPHORE < 0 THEN SUSPEND PROCESS
        RETURN


V(SEMAPHORE)

        SEMAPHORE = SEMAPHORE + 1
        IF SEMAPHORE <= 0 THEN AWAKEN PROCESSES
        RETURN
```

Figure 15

4) Termination algorithm (see Figure 16)

The task executing the termination algorithm simply sends a message back up to the Monitor via the event mechanism. The conditional is to test to see whether or not the message was received by the Monitor. The message is re-transmitted until it is received. At this time the task terminates.

4) TERMINATION (A CREATED PROCESS)

```
CAUSE(EVENT_I,STATE_J,MESSAGE_K,,1,1,STATUS)
IF STATUS(NUMBER) = 0 THEN GO TO TERMINATION ELSE EXIT
```

Figure 16

CHAPTER III
CODE & STRUCTURES

III. Introduction

This chapter deals primarily with the internal structures implemented in both the Monitor and the peripheral driver prototype. Since the amount of code written totals close to 400 pages (See APPENDICES II AND III), it would be impractical to detail all of the coding styles, techniques, and structures employed. In order to give the reader the proper flavor and scope of the work, only the most important areas will be discussed. For the super inquisitive, the listings with comments are included in the Appendix.

The chapter is broken down into three sections. The first deals with the universally common structures (e.g. core, queue, and task management). The second and third sections describe particular portions of the Monitor and the peripheral driver prototype, respectively.

III. 1 Common Code & Structures

III.1.1 Memory map

All modules can be separated into four distinct sections (see Figure 17). The first is the code of the algorithms. It is the set of instructions that is never modified. It is followed by the set of constants, lists, and tables that are likewise never modified. The third section contains all the queue heads. And the last section is the dynamic storage, or buffer, area.

These four sections describe the memory map for any module. Since the first two sections are never modified, a simple debugging trick is to compare the first two sections of a module after it has run to its original self. If the two don't match, then the programmer knows that there is a bug in the pure procedure -- modifying an algorithm.

LOCATION COUNTERS **MEMORY MAP**

```
                HEAD
          *
          *
          *                                   LOCATION COUNTERS
          *
          *
          *     THE FOLLOWING PREDEFINES THE ORDER IN WHICH THE LOCATION
          *     COUNTERS WILL OCCUR. INDEPENDENT OF THE ORDER IN WHICH
          *     THEY ARE USED WITHIN THE PROGRAM.
          *
000000          USE     CODE                  MAIN PROGRAM SEGMENT
000000    ZCODE BSS     0
          *
          *
004620          USE     CONST                 STORAGE FOR CONSTANTS. TABLES
004620    ZCONS BSS     0
          *
          *
005140          USE     QSTOR                 FOR ALL QUEUES
005140    ZQSTR BSS     0
          *
          *
005300          USE     STORE                 FOR ALL DYNAMIC STORAGE
005300    ZSTOR BSS     0
          *
          *
000000          USE     CODE                  CODE LOCATION COUNTER INITIALLY
          *$*   DISK    MMEDEFS
```

Figure 17

III.1.2 Task control Blocks (TCB)

A task as defined earlier is an EP, IP word pair -- the EP, environment pointer, points to the task's modifiable, or work, area and the IP, instruction pointer, points to the next instruction of some algorithm that the task wishes to execute. As pointed out earlier, this allows for the 'simultaneous' execution of multiple tasks, i.e. multi-programming or parallel processing. In actuality, since there is only one processor on the G.E. system, there can be only one task in execution. All other ready-to-run tasks are queued on the processor queue waiting for the currently executing task to either terminate or block. As soon as this happens, the processor is assigned to the next (top) task of the processor queue. The effect is seemingly parallel processing. While several tasks may be blocked waiting for I/O to complete, another task is executing code.

Since there can be many tasks in various states of execution, the task's EP was standardized (see Figure 18). Each data area is 24 words long with the last half reserved for temporary storage. The first three words deal with Executive status return after the issuance of a system primitive. Control is transferred to word three of the block after the status has been returned. It contains an execute double statement, 'XED', which links this task block onto the processor queue. This moves the task from the blocked state to the ready-to-run state. The remaining four words contain support pointers -- pointers to additional information.

The macro GETT creates a task. The macro will return in symbolic index register T--T for task--a pointer to an allocated block 24 words long. To terminate a task, the RELT macro is called. It returns the 24 word block pointed to by the contents of symbolic index register T to the core management routines where the contents of T are destroyed (i.e. made to point out of core bounds).

```
                 *                    TCB
                 *
                 *
                 *      THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
                 *      IN THE TRAP BLOCK (TBLOCK).
                 *
000000           SRW1   EQU    0      FIRST STATUS RETURN WORD FROM EXEC
000001           SRW2   EQU    1      SECOND STATUS RETURN WORD
000002           RET    EQU    2      SAVED IC/IR WHEN EXEC SPRINGS TRAP
000003           XED    EQU    3      CONTROL IS TRANSFERRED HERE WHEN EXEC
                 *                    SPRINGS THE TRAP.  IT CONTAINS AN XED
                 *                    OF A CHAIN WHICH LINKS THE TRAP TO THE
                 *                    MASTER TASK QUEUE.
000004           TRA    EQU    4      (UPPER) RESTART ADDRESS FOR TASKS ON
                 *                    ON A QUEUE (SUCH AS THE QSTASK)
                 *                    (LOWER) MAY BE USED TO SAVE RETURN
                 *                    FROM A REENTRANT ROUTINE
000005           LINK   EQU    5      (UPPER) LINK TO PREVIOUS TCB
000006           NCB    EQU    6      (UPPER) POINTER TO NCB
000006           JCB    EQU    NCB    (LOWER) POINTER TO JCB
000007           SPARE  EQU    7      SPARE
000030           LEN    EQU    24     LENGTH OF TCB (NICE IF MULTIPLE OF 8)
                 *
                 *
000027           TEMP1  EQU    LEN-1     TEMPORARY STORAGE AT END OF BLOCK
000026           TEMP2  EQU    TEMP1-1   MORE TEMPORARY STORAGE
000025           TEMP3  EQU    TEMP2-1
000024           TEMP4  EQU    TEMP3-1
000023           TEMP5  EQU    TEMP4-1
000022           TEMP6  EQU    TEMP5-1
000021           TEMP7  EQU    TEMP6-1
000020           TEMP8  EQU    TEMP7-1
                 *
                 *      NO ONE EXCEPT R$GETC SHOULD USE TEMP9 - TEM16
                 *
000017           TEM9   EQU    TEMP8-1
000016           TEM10  EQU    TEM9-1
000015           TEM11  EQU    TEM10-1
000014           TEM12  EQU    TEM11-1
000013           TEM13  EQU    TEM12-1
000012           TEM14  EQU    TEM13-1
000011           TEM15  EQU    TEM14-1
```

Figure 18

III.1.3 Queue Control Block (QCB)

All modules are internally organized around a set of queues. Each resource has associated with it a queue on which are put all tasks currently requesting that resource. As seen from the description of the major algorithms, the queues are used for synchronizing and communicating between algorithms. This synchronization and communication is of paramount importance in a system where many tasks are competing for a few limited resources.

To ease coding problems, all queues used in all modules are of the same structure -- a 16 word block (see Figure 19). A queue consists of a possibly empty linked list of task control blocks. The pointers point to word 4 (Q$OFFST) of a block. The link pointers are stored in word 3 (Q$LINK) of a block. The word at location Q$FIRST points to Q$OFFST of the first block of the queue. The location Q$LAST points to Q$OFFST of the last block of the queue. The empty queue is denoted by the word Q$LAST pointing to Q$FIRST+1 (i.e. pointing to itself).

Accompanying the queue structure is a set of macros that will manipulate any queue. For example, task blocks may be enqueued via the ENQ macro and dequeued via the DEQ macro.

QUEUE MANAGEMENT DEFINITIONS

```
                HEAD    Q
       *
       *
       *                                    QCB
       *
       *
       *        THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
       *        IN A QBLOCK GENERATED BY THE QUEUE MACRO.
       *
000000 FIRST    EQU     0               POINTER TO FIRST BLOCK OF QUEUE
000001 LAST     EQU     FIRST+1         POINTER TO LAST BLOCK OF QUEUE
000002 XADD     EQU     LAST+1          INSTRUCTION PAIR FOR ADDINGA BLOCK
000004 XENQ     EQU     XADD+2          INSTRUCTION PAIR FOR ENQUEUEING
000006 XDEQ     EQU     XENQ+2          INSTRUCTION PAIR FOR DEQUEUEING
000010 XINV     EQU     XDEQ+2          INSTRUCTION PAIR FOR INVERTING
000012 BUSY     EQU     XINV+2          RESPONSIBLE BLOCK IF QUEUE IS BUSY
                                        ZERO OTHERWISE
000013 MAX      EQU     BUSY+1          MAXIMUM NUMBER OF ITEMS ASSOCIATED WITH (
000014 AVAIL    EQU     MAX+1           NUMBER OF ITEMS CURRENTLY AVAILABLE
000015 SPAR1    EQU     AVAIL+1         SPAR1
000016 SPAR2    EQU     SPAR1+1         SPAR2
000017 ABBR     EQU     SPAR2+1         ASCII ABBREVIATION OF QUEUE
000020 LEN      EQU     ABBR+1          LENGTH OF QUEUE (WISE TO KEEP EVEN)
       *
       *
000004 OFFST    EQU     4               OFFSET FOR QUEUE POINTER
000003 LINK     EQU     OFFST-1         FORWARD LINK POINTER
```

Figure 19

III.1.4 Core Control Block (CCB)

As mentioned in the memory map discussion, the fourth section of all modules in the dynamic storage area. This area is organized into the free memory list. The list consists of a possibly empty linked list of blocks. The forward/backward pointers of a block point to the first word of the succeeding/preceding blocks, respectively. The link pointers are stored in words 0 and 1, respectively, of the block (see Figure 20). Hence the minimal theoretical size of a block is two words; the practical size is eight. The total length of the block is also kept in word 0. By design conventions, the pointers are upper half quantities and the length is a lower half quantity. The empty list is denoted by the forward link of R$FIRST pointing to R$LAST and the backward link of R$LAST pointing to R$FIRST (see Figure 21).

There is a set of macros, RELC and GETC, at the programmers disposal. RELC releases a block of core back to the free memory list. As a safety feature, each block released is first checked to see if it is out of bounds of the dynamic buffer area and if it has already been released. If either condition holds, the module is halted and copied out to a dump file. GETC returns to the caller a block of memory. The size requested is rounded up to the next multiple of eight. If there is no block big enough on the free list to satisfy the request, a system request is made to expand the size of the dynamic buffer.

CORE MANAGEMENT DEFINITIONS

```
                HEAD      R
         *
         *
         *                          CCB
         *
         *
         *      THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS IN
         *      A BLOCK ON THE FREE MEMORY LIST.
         *
000000   LINKF  EQU       0             POINTER TO SUCCESSOR (UPPER)
000000   LEN    EQU       LINKF         TOTAL LENGTH (IN WORDS) OF BLOCK (LOWER)
000001   LINKB  EQU       LINKF+1       POINTER TO PREDECCESSOR
         *$*    DISK      HEAD
```

Figure 20

```
                      2670 *
                      2671 *                 FREE MEMORY LIST
                      2672 *
        005344        2673             USE      STORE
006400 000000         2674 FIRST       ZERO     $NEXTF,0         FORWARD LINK/ LENGTH OF BLOCK
000000 000000         2675             ZERO     0,               BACKWARD LINK/ <NOT USED>
                      2676 *
                      2677 *
000000 000000         2678 LAST        ZERO     0,0              FORWARD LINK/ LENGTH OF BLOCK
006400 000000         2679             ZERO     $NEXTB,          BACKWARD LINK/ <NOT USED>
        001152        2680             USE      PREVIOUS
```

Figure 21

III.1.5 MACROS

Programming applications frequently involve (1) the coding of a repeated pattern of instructions that within themselves contain variable entries at each iteration of the pattern and (2) basic coding patterns subject to conditional assembly at each occurrence. The macro-operation gives the programmer a shorthand notation for handling (1) and (2) through the use of a special type of pseudo-operation referred to in the GE-625/635 Macro Assembler as a MACRO [4]. Having once determined the iterated pattern, the programmer can, within the MACRO, designate selectable fields of any instruction of the pattern as variable. Thereafter, by coding a single MACRO instruction, the programmer can use the entire pattern as many times as needed, substituting different parameters for the selected subfields on each use.

When the programmer defines the iterated pattern, he gives it a name, and this name then becomes the operation code of the MACRO instruction by which he subsequently uses the macro-operation.

What follows is a discussion of the more representative MACROS employed. Specifically, MACROS dealing with (1) system calls for executive services, such as opening a file or reading/writing an opened file, (2) parallel tasking for multiprogramming capabilities, and (3) diagnostic aids, which greatly facilitated the debigging of the module are treated in detail.

All trapping system calls for executive services are done through MACROS. Essentially they supply a mechanism whereby a slave task can issue requests to the EXECUTIVE for services. Each of these MACROS is coded to the following specific conventions: the first instruction is a 'TSXO', a transfer and save the current Instruction Counter plus one in Index Register XO, to the subroutine with the same name as the MACRO; it is then followed by the argument list for that particular executive service.

The READ macro (See Figure 22) typifies a system call for an executive service. Figure 23 points out the importance of MACROS in the overall design. The READ subroutine, which is called only by the READ macro, in turn employs the three following macros:

1) SETUP (see Figures 24 and 25). This macro and subroutine combination initializes the Task Control Block for the task currently wishing to make a system call for any executive service. It zeroes out the first three status words of the TCB and places the execute double instruction in the link word of the block. This instruction links the task unto the processor queue after the executive service call is completed. SETUP also places the proper IP in the TCB such that the task can continue executing its current algorithm.

2) CKPT (see Figure 26) -- Checkpoint is a diagnostic tool. If the assembly time debugging flag, $DBG, is on, this macro will cause the registers to be stored in 8-word blocks in a circular queue. This is very useful for backtracking of an error.

3) EXIT (see Figure 27) -- The EXIT macro terminates a thread of control by returning to the task distributor.

Thus the READ subroutine works as follows:

It saves the pointer to the caller, but bumps the return past the argument list. It sets up the Task Control Block. Next it picks up the parameters from the argument list and for debugging purposes saves the registers in a circular debugging queue. The MME instruction notifies the Executive of the system call. Instead of waiting in an idle loop for the completion of the call, the task relinquishes control of the processor by executing the EXIT macro. This is how the module is multiprogrammed.

The last major important common macro is the BRANCH macro (see Figure 28). Any task that executes the BRANCH macro creates another asynchronous task. This is the final part of the multi-programming system.

READ MACRO

```
000536              USE     CODE
                    HEAD
               *
               *
               *                                  READ
               *
               READ MACRO   FRN,CORELOC,N,MODE
                    TSX     0,$READ
                    ARG     #1                    FRN ADDRESS
                    ARG     #2                    ADDRESS OF CORE LOC
                    ARG     #3                    NUMBER OF ELEMENTS
                    ARG     #4                    MODE
                    ENDM    READ
```

Figure 22

```
                                1282 *
                                1283 *          READ -- SUBROUTINE
                                1284 *
                                1285 *     THIS SUBROUTINE IS CALLED BY THE READ MACRO.  IT ISSUES THE
                                1286 *     COMMAND TO READ THE NEXT N ELEMENTS OF FRN IN A PARTICULAR MODE
                                1287 *
                                1288 *     CALL WITH
                                1289 *          C(XT) = TBLOCK-ADDRESS
                                1290 *          C(XJ) = JBLOCK-ADDRESS
                                1291 *     ENTER BY
                                1292 *          TSX 0,3READ
                                1293 *          ARG ADDRESS OF FRN
                                1294 *          ARG ADDRESS OF CORELOC
                                1295 *          ARG N
                                1296 *          ARG MODE
                                1297 *     RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                                1298 *     RETURNS WITH
                                1299 *          C(XT) = TBLOCK-ADDRESS
                                1300 *          C(XJ) = JBLOCK-ADDRESS
                                1301 *          C(XL) = RESTART-ADDRESS
                                1302 *     USES LOCAL TEMPORARY ONLY
                                1303 *
                                1304 *
 BINARY CARD IOS00011
10536  005321 7400 00           1305 READ   STX     0,READT        POINTER TO ARGUMENT LIST
00537  000004 0200 03           1306        ADLX    0,4,DU         RESTART ADDRESS
                000540          1307        SETUP
00540  000510 7170 00                       XED     $SETUP
00541  005321 2220 57           1308        LDX     2,READT,IDC    LOAD FRN
00542  005321 2240 57           1309        LDX     4,READT,IDC    LOAD CORE LOC
00543  005321 2250 57           1310        LDX     5,READT,IDC    LOAD N
00544  005321 2260 57           1311        LDX     6,READT,IDC    LOAD MODE
)0545  000004 2200 03           1312        LDX     0,.READ,DU     LOAD MME NUMBER
                000546          1313        CKPT                   CHECKPOINT
00546  000474 7170 00                       XED     X$CKPT
00547  000000 0010 00           1314        MME                    READ
                000550          1315        EXIT
0550   003074 7100 00                       TRA     $EXIT
                                1316 *
                005321          1317        USE     STORE
5321   000000 0000 20           1318 READT  ARG     0,*            POINTER TO ARGUMENT LIST
                000551          1319        USE     PREVIOUS
```

```
                 TRAP SETUP MACRO

000510                  USE     CODE
                        HEAD
         *
         *
         *                                  SETUP MACRO
         *
         SETUP          MACRO               NO ARGUMENTS
                        XED     SSETUP
                        ENDM    SETUP
```

Figure 24

```
                                    1144 *
                                    1145 *                   SETUP -- SUBROUTINE TO SET UP A TRAP
                                    1146 *
                                    1147 *        CALL WITH
                                    1148 *                    C(XT) = TBLOCK-ADDRESS
                                    1149 *                    C(XJ) = JBLOCK ADDRESS
                                    1150 *                    C(XO) = TRANSFER ADDRESS FOR J$TRA
                                    1151 *        ENTER BY
                                    1152 *                    XED T$SETUP
                                    1153 *        DESTROYS C(A), C(Q), C(XO)
                                    1154 *        USES NO TEMPORARIES
                                    1155 *
                                    1156 *
                          000510    1157          EVEN
                          000510    1158 SETUP    BSS      0
  000510  000004 7400 11            1159          STX      0,T$TRA,T          SET T$TRA = RESTART ADDRESS
  000511  000512 7000 00            1160          TSX      0,*+1              BREAK XED
  000512  000000 4310 03            1161          FLD      0,DU               ZERO OUT A AND C
 END OF BINARY CARD IOS00010
  000513  000000 7570 11            1162          STAQ     T$SRW1,T           ZERO STATUS WORDS
  000514  000520 2370 00            1163          LDAQ     TRAP-1             GET ZERO, XED WORDS
  000515  000002 7570 11            1164          STAQ     T$XED-1,T          SAVE ZERO, XED
  000516  000000 7100 10            1165          TRA      0,0                RETURN
                                    1166 *
                                    1167 *
                                    1168 *                   TRAP -- XED SEQUENCE TO PUT BLOCK ON Q$TASK
                                    1169 *
                          000520    1170          EVEN
  000520  000000 000000             1171          ZERO                        CAN BE USED FOR CLEARING RET WORD!
  000521  000522 7170 00            1172 TRAP     XED      *+1                THIS IS EXECUTED FROM THE TBLOCK
  000522  005161 5540 54            1173          STC1     Q$LAST+Q$TASK,DI  *UPDATE PREVIOUS LAST POINTER
  000523  000524 7170 00            1174          XED      *+1                CONTINUE WITHOUT AFFECTING IC
  000524  005161 5540 00            1175          STC1     Q$LAST+Q$TASK      UPDATE POINTER TO LAST
  000  5  777777 6300 04            1176          RET      -1,IC              RETURN TO POINT OF INTE  UPTION
                                    1177 *S*      DISK     SYSCALLS
```

Figure 25

CHECKPOINT MACRO

```
         USE      CODE
         HEAD     X
*
*
*                                       CHECKPOINTS
*
*        THIS MARCRO CAUSES THE REGISTERS TO BE STORED IN 8-WORD
*        BLOCKS IN A CIRCULAR QUEUE FOR DEBUGGING USE.  INFORMATION
*        IS STORED IN THE FOLLOWING FORMAT:
*
*
*        C(0)  =  C(X0) (UPPER)
*                 C(X1) (LOWER)
*        C(1)  =  C(X2) (UPPER)
*                 C(X3) (LOWER)
*        C(2)  =  C(X4) (UPPER)
*                 C(X5) (LOWER)
*        C(3)  =  C(X6) (UPPER)
*                 C(X7) (LOWER)
*        C(4)  =  C(A)
*        C(5)  =  C(Q)
*        C(6)  =  C(E) (0-7 BITS)
*        C(7)  =  C(TR) (0-23 BITS)
*
*
*                                       CKPT
*
CKPT     MACRO    <NO ARGUMENTS>
         IFE      $DBG,$ON,1
         XED      X$CKPT
         ENDM     CKPT
```

Figure 26

EXIT MACRO

```
000473            USE     CODE
                  HEAD
        *
        *
        *                               EXIT
        *
        *         EXIT TERMINATES A THREAD OF CONTROL BY RETURNING TO THE
        *         TASK DISTRIBUTOR.
        *
        EXIT      MACRO                 NO ARGUMENTS
                  TRA     $EXIT
                  ENDM    EXIT
```

Figure 27

```
*
*                                  BRANCH MACRO
*
*       THIS MACRO CREATES AN ASYNCHRONOUS TASK TO BE PREFORMED AT A
*       LATER TIME.  IT CAN GIVE THE CREATED TASK THE CURRENT TBLOCK
*       OR GIVE IT A NEW TBLOCK.  EITHER WAY FOUR PARAMETERS MAY BE
*       BETWEEN THE TWO TBLOCKS.
*
*       PAST INFORMATION IS PLACED IN TEMP1 THRU TEMP4 OF
*       THE TASK PLACED ON THE Q$TASK QUEUE.
*
*       CALLS
*                   T$GETT
*       CLOBBERS C(XX), C(X0)
*
*
BRANCH MACRO    PASS,XFER ADD,C(TEMP1),C(TEMP2),C(TEMP3),C(TEMP4)
       TSX      0,T$GETT          GET A NEW TBLOCK
       EAX      X,0,T             C(XX) POINTS TO NEW TBLOCK
       LDX      T,T$LINK,X        C(XT) POINTS TO OLD TBLOCK
       INE      '#3',,'',2
       LDQ      #3                SAVE FIRST PARAMETER
       STQ      T$TEMP1,X
       INE      '#4',,'',2
       LDQ      #4                SAVE SECOND PARAMETER
       STQ      T$TEMP2,X
       INE      '#5',,'',2
       LDQ      #5                SAVE THIRD PARAMETER
       STQ      T$TEMP3,X
       INE      '#6',,'',2
       LDQ      #6                SAVE FOURTH PARAMETER
       STQ      T$TEMP4,X
       INE      '#1','PASS',1     T IS THE BLOCK THAT IS PASSED
       EAX      T,0,X             PASS NEW BLOCK
       EAX      0,#2              POINT TO TRANSFER ADDRESS
       STX      0,T$TRA,T         INTO QUEUE BLOCK
       EAX      0,Q$OFFST,T       PREPARE TO QUEUE
       XED      Q$XADD+Q$TASK     GET ON THE TASK QUEUE
       IFE      '#1','PASS',1     WHICH BLOCK TO WE GIVE BACK AS CURRENT
       EAX      T,0,X             GIVE BACK NEW BLOCK
       INE      '#1','PASS',1
       LDX      T,T$LINK,X        NO, GIVE BACK OLD BLOCK
       BUGXR    (0,X)
       ENDM     BRANCH
```

Figure 28

III.1.6 Communications Network

There exists a private communications network among the Monitor and all of its submodules (i.e. the peripheral drivers). At start up time for the Monitor, it creates the network by opening three scratch events and passing a file reference number, FRN, of each to each submodule spawned. For the submodules, these events are referenced by canonical numbers:

1) FRNO -- This is the command event for the drivers. Each driver is allowed notify access only. Commands are channeled to the specified submodule by the STATE when caused.

2) FRN1 -- This is the command reply event for the drivers. Each driver is allowed cause access only. In order to inform the Monitor, a peripheral driver simply causes this event with the proper STATE.

3) FRN2 -- As implied above, FRNO and FRN1 are an input/output pair. FRN2, however, is not paired at all. The monitor uses this event as a pass event sending files to be processed and devices down to its sons. The sons (peripheral drivers) never pass anything back to the Monitor. They simply close the files.

See Figure 29 for the message formats used in the communication network.

```
*
*      MESSAGE FORMATS:  RETURNED IN T$SRW2,T (UPPER)
*
*      FOR FRN2 --
*                BITS 0 - 3 = JOB NUMBER (WITH 0 ILLEGAL)
*                BITS 4     = BANNER (ON MEANS SUPPLY BANNER)
*                BITS 5     = OUTPUT MODE:  512/ 320 (ON MEANS 320)
*                BITS 6 -17 = START ADDRESS (IN ELEMENTS)
*
*      FOR FRN1 --
*                BITS 0 - 3 = MUST BE ZERO
*                BITS 4 - 7 = COMMAND
*                BITS 8 -14 = <NOT USED>
*                BITS 15-17 = DEVICE UNIT NUMBER (0-7)
*
*
*      FOR FRN0 --
*                BITS 0 - 3 = JOB NUMBER (MUST BE ZERO)
*                BITS 4 - 7 = COMMAND
*                BITS 8 -14 = <NOT USED>
*                BITS 15-17 = DEVICE UNIT NUMBER (0 THRU 7)
```

Figure 29

III.2 Code and Structures of the Monitor

III.2.1 Job Control Block (JCB)

The Job Control Block (see Figure 30) is merely an extension of the Task Control Block. The JCB contains a coded description of the request submitted by a slave process. The task executing the request examines the JCB for the coded instructions. The JCB also contains additional information (e.g. the user number to bill, a unique number for job identification purposes and lots of debugging aids -- the extra pointers to the TCB, etc.).

JOB CONTROL BLOCK DESCRIPTION

```
                 HEAD    J
        *
        *
        *                                JCB
        *
        *
        *
        *        THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
        *        IN THE JOB CONTROL BLOCK (HERAFTER CALL JCB).
        *
000000  QFRN     EQU     0                (UPPER) FRN OF ASSOCIATED INPUT FILE
000001  QFLOC    EQU     QFRN+1           R/W PTR POSITION OF INPUT FILE IN *QFRN*
000002  FRN      EQU     QFLOC+1          FRN OF FILE TO BE PROCESSED
000003  TYPE     EQU     FRN+1            (UPPER) TYPE
000003  DISP     EQU     TYPE             (LOWER) DISPOSITION
000004  ACODE    EQU     DISP+1           ACODE FOR ACCOUNTING
000005  NCB      EQU     ACODE+1          (UPPER) PTR TO NCB
000005  TCB      EQU     NCB              (LOWER) PTR TO TCB
000006  STATI    EQU     NCB+1            INITIATE STATE FOR COMMUNICATIONS
000007  JOB      EQU     STATI+1          JOB NUMBER
000007  STATT    EQU     JOB              TERMINATE STATE FOR COMMUNICATIONS
000010  MESS     EQU     STATT+1          MESSAGE FOR COMMUNICATIONS
000011  BUF      EQU     MESS+1           WORKING BUFFER
000012  SIZE     EQU     BUF+1            AMOUNT OF DATA TO BE PROCESSED
000013  RES      EQU     SIZE+1           START OF RESOURCE REQUIREMENT LIST
000026  TT       SET     RES+3+1-QFRN+7   ROUND TO MULTIPLE OF 8
000020  TT       SET     TT/8*8           ROUND
000020  LEN      EQU     TT               LENGTH OF JCB (MINIMUM LENGTH = 16. )
        *S*      DISK    NCB
```

Figure 30

III.2.2 Notify Control Block (NCB)

A Notify Control Block (see Figure 31) is nothing more than a TCB with frills. The perpetual tasks which execute the external input algorithms (see Figures 6 and 12) have an NCB instead of just a plain TCB. The extra is for identification of the request and what and how to handle a request after it is received.

```
      *                                    NCB
      *--------------------------------------------------------------------------
      *        THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
      *        IN THE NOTIFY BLOCK (ALIAS NCB).
      *
      *             A NCB IS A TCB WITH EXTRAS.
      *
000000  SRW1   EQU    T$SRW1
000001  SRW2   EQU    T$SRW2
000002  RET    EQU    T$RET
000003  XED    EQU    T$XED
000004  TRA    EQU    T$TRA
000005  LINK   EQU    T$LINK
000005  RLINK  EQU    LINK             (LOWER) RESTART AFTER NOTIFY
000006  NCB    EQU    T$NCB
000006  JCB    EQU    T$JCB
000007  ABBR   EQU    T$SPARE          ASCII ABREVIATION OF NCB
000027  TEMP1  EQU    T$TEMP1
000026  TEMP2  EQU    T$TEMP2
000025  TEMP3  EQU    T$TEMP3
000024  TEMP4  EQU    T$TEMP4
000023  TEMP5  EQU    T$TEMP5
000022  TEMP6  EQU    T$TEMP6
000021  TEMP7  EQU    T$TEMP7
000020  TEMP8  EQU    T$TEMP8
000017  TEM9   EQU    T$TEM9
000016  TEM10  EQU    T$TEM10
000015  TEM11  EQU    T$TEM11
000014  TEM12  EQU    T$TEM12
000013  TEM13  EQU    T$TEM13
000012  TEM14  EQU    T$TEM14
000011  TEM15  EQU    T$TEM15
000010  TEM16  EQU    T$TEM16
000024  ERN    EQU    TEMP4            ERN FOR NOTIFY ***BENE NOTA***
000025  STATE  EQU    TEMP3            STATE FOR NOTIFY ***BENE NOTA***
000030  QFRN   EQU    T$LEN            (UPPER) INPUT Q FILE
000031  QFLOC  EQU    QFRN+1           R/W PTR FOR "QFRN"
000032  BUSY   EQU    QFLOC+1          NO. OF FILES CURRENTLY ACTIVE FROM THIS
                                       INPUT QUEUE FILE
000033  RUN    EQU    BUSY+1           PTR TO R$OMAX FOR THE TYPE REOURCES
                                       NEEDED BY THIS JOB.  IF RSOMAX = 0, THEN
                                       SHOULD IGNORE HIM FOR NOW (SAVVY?)
000034  QUEUE  EQU    RUN+1            PTR TO WAIT Q-LIST TO PLACE JOB ON
000035  RES    EQU    QUEUE+1          RESOURCE LIST  (MUST BE LAST)
                                       THIS WAY WE TEST FIRST TO SEE IF WE
```

III.2.3 Peripheral Management

The peripherals are managed exclusively by the Monitor. It is responsible for the allocation of all resources. The peripheral drivers never have to worry with scheduling and managing per se. The Monitor has all the peripheral I/O devices organized into a hierarchical structure. At the head of this structure is the peripheral type table. It has an entry for each type of system resource (e.g. line printer, mag tape, etc.). If the entry is empty, then there are no resources of that type ever available (e.g. invalid types). If it is not empty, it contains a pointer to peripheral header table.

The peripheral header table is composed of device headers. A device header is the item pointed by an entry in the peripheral type table. A device header contains:

1) a pointer to a device table,

2) information regarding the configuration of the system (e.g. the maximum number of devices of a certain type, the number the Monitor owns, the number currently free, accesses allowed on device, etc.).

The entries of a device table have a one-to-one correspondence between a 'device' and the real physical I/O device in the machine room. The device contains the name of the device, the FRN when opened, flag bits telling of the current status, and if busy, who is responsible.

All of these tables are generated by Macros. There is a pair of macros GETP and RELP that seize and release a peripheral of a given type.

III.2.4 Operator Interface

The operator interface is a set of routines that allows for conversation between the monitor and the operator of the system. The conversation takes form in the set of the following five commands:

1) GET <peripheral-name >; ...; <peripheral-name >
(where < peripheral-name > is the 4 letter generic name for any peripheral device).

It reopens the peripherals identified by the four letter abbreviations. After a successful open, the peripheral is passed down to the appropriate peripheral driver via the communications network.

2) RELEASE <peripheral-name >; ...; < peripheral-name >.

The Monitor sends a command via the communications network to the appropriate submodule telling it to close the specified peripheral as soon as possible. The submodule when finished will send back a corresponding message.

3) KILL < peripheral-name > ; ...; < peripheral-name >

This is an immediate RELEASE command. The output to the peripheral is stopped immediately.

4) START <peripheral-name >; ...; <peripheral-name >

It is a command to restart from the beginning the output to the specified peripheral.

5) EXIT

The command to terminate the conversation with the monitor.

The conversation takes place on the operator's console. Thus there will be a record of the commands for later debugging if necessary. The conversation is initiated by the operator. By running a certain program, the operator can "call up" the Monitor for a conversation.

III.2.5 Initialization

The Monitor initialization routine is a very simple procedure. It initializes the registers and location zero in order to catch wild transfers to the beginning. Through a system primitive, it sets up the fault vector. It then opens the various system files (e.g. PRINT-FILE-QUEUE, PUNCH-FILE-QUEUE, PRINT-FILE-EVENT, etc.); it then opens the peripheral driver modules. The next step creates the internal communications network between Monitor and submodule. It opens three scratch events ($EVENT_0$, $EVENT_1$, $EVENT_2$ (see Figure 4). It completes the system by spawning the peripheral drivers. At spawn time, the Monitor passes a link to the communications network to each submodule. The set of predefined I/O devices are opened and passed to the appropriate peripheral driver. Lastly, the Monitor allows slave process requests to be accepted by putting out notifies on all events. It then goes blocked waiting for any request (see Figure 6).

III.3 Code and Structures of the Peripheral Driver Prototype

III.3.1 Job Control Block (JCB)

There is a one-to-one correspondence between a JCB and a peripheral. Therefore the number of JCBs equals the number of peripheral devices a module can have at the maximum. Hence the JCBs are all pre-allocated for the following reasons:

1) Since a JCB is allocated at job initialization and is not released until the job completes, we don't want memory tied down that long.

2) Also we don't want to create holes in the dynamic buffer area.

3) There are only as many JCBs as real devices (currently 3 -- 2 for the line printer module and 1 for the card punch module).

There is the following pair of macros that manage the JCBs:

GETJ gets a JCB for the calling routine

RELJ releases a JCB (validity checks are made, of course)

III.3.2 Notify Control Block (NCB)

For the sake of symmetry, the NCB's of the peripheral drivers were coded identical to those of the Monitor. For a full description, see the discussion in section III.2.2 and see section III.1.6 discussing the communications network.

III.3.3 Peripheral Management

There is no peripheral management *per se* for a peripheral driver prototype. All managing is done by the Monitor. That is, a driver never has more jobs to process than it has I/O devices to send the data.

III.3.4 Initialization

The peripheral driver initialization routine is very simple. It initializes the registers and location zero similar to the Monitor. Likewise it sets up the fault vector. The last step is to create a set of tasks to talk to the Monitor via the communications network -- event structure. These tasks execute the external input algorithm. The first statement of it is to go blocked waiting for an interrupt signal. (See Figure 12).

CONCLUSION

During the summer of 1971, the following portions of the Input/Output Scheduler system were implemented:

1) The Monitor was fully implemented. All five major algorithms were implemented and have run. Also, the operator's command language was coded to allow the operator control over the I/O peripheral devices.

2) Of the peripheral driver prototypes, the line printer and card punch modules have been fully implemented. Only the card reader module lacks to be coded. The coding and debugging of such a card punch module, given all the standard macros, structures, etc., would take approximately two, maybe three, man-weeks of work.

As currently implemented, the Monitor is roughly 3 1/2 to 4 K of code, including the dynamic buffer area (see Appendix II). The line printer module and card punch module, due to striking similarities of the devices that they control, were coded together in a single module (see Appendix III). Together they are roughly 3 K big, including 1 K of dynamic buffer area. Hence the overall system is approximately 7 K. Yet the Monitor is not core resident. Thus the effective size of the system drops to 3 K. Although the peripheral drivers are allowed to be core resident (in order to increase their efficiency when transferring data), they are swappable. When there is nothing for them to do, the whole I/O system is swapped out of core, unto the drum, waiting for a request.

This new system, when fully implemented, would remove approximately 3 1/2 K of permanently core resident code from the present Listener. This is the same order of magnitude of core that would be used by the new I/O Scheduler system. Yet the new system includes the double-buffering of data. The old Listener was single-buffered. Moreover, the new system handles the two line printers; the Listener handles only one. This IOS system has been running experimentally since September of '71. Although some "bugs" have appeared, they have been corrected. The output efficiency of the system has noticeably increased.

BIBLIOGRAPHY

1. Berstein, A. J., Detlefsen, G. D., and Kerr, R. H., Process Control and Communication in a General Purpose Operating System, General Electric TIS Report 69-C-357 (October, 1969).

2. Berstein, A. J. and Hamm, J. B., The Design and Implementation of a Directory Hierarchy for a General Purpose Operating System, General Electric TIS Report 69-C-356 (October, 1969).

3. Dijkstra, E. W., Cooperating Sequential Processes, Technological University, Eindhoven (September, 1965).

4. Honeywell Information Systems, Inc., Models 625/635 Programming Reference Manual, CPB-1004F, (July, 1969).

5. Johnson, J. B., The Contour Model of Block Structured Processes, General Electric TIS Report 70-C-366 (October, 1970).

6. Kerr, R. H., Berstein, A. J., Detlefsen, G. D. and Johnson, J. B., Overview of the R & DC Operating System, General Electric TIS Report 69-C-355 (October, 1969).

7. Knuth, D.E., The Art of Computer Programming, Volume 1, Addison-Wesley, Reading, Massachusetts, 1968.

8. , "LISTENER", Version V-146, General Electric R & DC (December 17, 1970).

APPENDICES

I. System Programmer's Manual for the R & DC 600 Operating System

II. Listing of Monitor

III. Listing of Peripheral driver prototype - Line Printer & Card Punch Module

APPENDIX I

This is the G.E. R & DC systems programmer's Manual. It was furnished through courtesy of the G.E. Research and Development Center.

SYSTEM PROGRAMMER'S MANUAL
FOR THE R&DC 600 OPERATING SYSTEM

SYSTEM PROGRAMMER'S MANUAL FOR THE R&DC 600 OPERATING SYSTEM

Revised August '70

PREFACE

Who This Manual Is Intended For

This manual is primarily intended as a tutorial and reference document for system programmers who are developing assembly language code which is to be run under the R&DC 600 operating system. A general familiarity with the GMAP assembly language and the fault and interrupt features of 600 machines is assumed.

The overview portions in the manual can also be used to provide an introduction to some of the features of the operating system. Additional information can be found in separate documents.[1]

Structure of the Manual

Section I gives a general overview of the operating system. The structure of the Executive is described together with the basic terminology which is utilized in describing its actions.

Section II gives a general description of the primitive commands which are available with the operating system. The technique for initiating a primitive command and handling the associated trap return is described in some detail.

Sections III through V describe the I/O primitives, the File System and File Primitives, and the Event Primitives.

Section VI provides a detailed description of each of the system primitives.

The Appendices summarize information which is useful for reference purposes.

How To Use This Manual

Programmers should read the introductory and overview sections before attempting to utilize the primitive descriptions in Section VI. The detailed description of the parameters which appear in these overviews can, however, be omitted on a first reading.

---

1. "Overview of the R&DC Operating System", TIS Report No. 69-C-355, October 1969.
   "Process Control and Communication in a General Purpose Operating System." TIS Report No. 69-C-357.
   "The Design and Implementation of a Directory Hierarchy For A General Purpose Operating System." TIS Report No. 69-C-356.

Non-Programmers should concentrate on the first few pages in Section II and Sections I and III-V.

Suggestions and Criticisms

Comments concerning this publication are solicited for use in improving future additions. Please send any recommended additions, deletions, corrections, or other information you deem necessary for improving this manual to: J.E. Kapitula, 4C26, Building K-1, General Electric Research and Development Center; P.O. Box 8; Schenectady, New York 12301.

Table of Contents

Table of Contents (continues)

Description of System Primitive Commands (continued)

APPENDICES

INDEX

Section I

Operating System Overview

OPERATING SYSTEM OVERVIEW

Introduction

The following is a brief description of the 600 Series Time Shared Computer System which has been implemented at the Research and Development Center.

The system provides teletype computing, remote batch and conventional batch processing capabilities. The system has been designed to be sufficiently flexible to allow the incorporation of additional software and hardware developments with minimum effort and disruption of existing service.

The impelemntation is based on multilevel executive structure, a generalized device independent file system, and a set of primitive commands issued to the executive by running programs.

System Capabilities

The computer system design offers flexibility both in terms of new and/ or unusual peripheral devices and in terms of programming ease. The following broad service capabilities are provided:

1) Teletype time sharing for (64) users
2) Large program capability
3) Extensive file system
4) Remote and conventional batch processing
5) An interface to remote and directly coupled computers
6) Ability to utilize unusual peripheral devices.

The teletype time-shared system presents the user with an interface similar to the Desk Side or Mark 2 Computer System. In addition, the teletype user is able to exercise the file system, run machine language programs, and use peripheral I/O devices and remote computers. The initial hardware configuration allows for -64- teletypes. However, the system design will allow this number to be extended with suitable hardware additions.

The system has the capability of running large or small programs intermixed. The running time of a program is generally proportional to the amount of system resources used and to the load on the system.

The system has an extensive file system. Users are able to access files by logical name, rather than by physical device. For most files the user has no control over the physical device allocation. The user is able to access, by name, certain peripheral devices, such as magnetic tapes, in a general manner.

A batch processing capability exists, as a person can initiate the running of a program from a card reader or magnetic tape unit in the same manner as from a teletype.

Several remote computers are connected to the RT-IOC* via high-speed phone lines. A GE115 computer is used for remote entry and output of batch jobs. A PDP-9 computer is used to drive a large cathode ray oscilloscope display unit. A Varian 620-i computer and a GEPAC-30 computer are also connected and are used for special applications.

A directly coupled computer, the GE 4020 is employed for experimental data acquisition, control, and data reduction.

A Varian 622-i computer is being implemented as a disk controller for both the GE 600 and the GE 4020 computers.

The system is able to handle new and unusual peripheral devices which may arrive from time to time with a minimum of reprogramming both in the executive and in the user programs.

* Real Time Input Output Controller

System Hardware Overview

The following is a summary of the hardware configuration. The basic unit of the system is the 4 Base Address Register 600 computer which is composed of three major modules: memory, processor, and real time input/output controller (RT-IOC).

The 600 system consists of a single processor module, an RT-IOC, and two 64K memory modules. The memory is composed of two memory (system) controllers and 64K of 36-bit (plus 1 parity bit) magnetic core storage with a 1.0 microsecond read-restore memory cycle time per controller.

The processor has been modified to include four base address registers for address relocation and protection.

The RT-IOC serves as the input-output interface for the system. This device is capable of transmitting data, in an asynchronous manner, between core memory and up to 32 peripheral devices. The following devices are attached to the RT-IOC:

1) Two -32- channel teletype multiplexors
2) 2 by 8 disk controller with two disk files.
3) 1 by 8 tape controller with eight tape handlers
4) Card reader (1100 CPM)
5) Card Punch (200 CPM)
6) Two line printers (1000 LPM)
7) Console teletypewriter
8) 1 by 8 high speed communications line multiplexor (CLM) (interface for remote computers, etc.)
9) Interface for GE 4020 (Memory to memory interface)
10) High speed analog to digital converter

A diagram of the configuration is shown in Figure I-1.

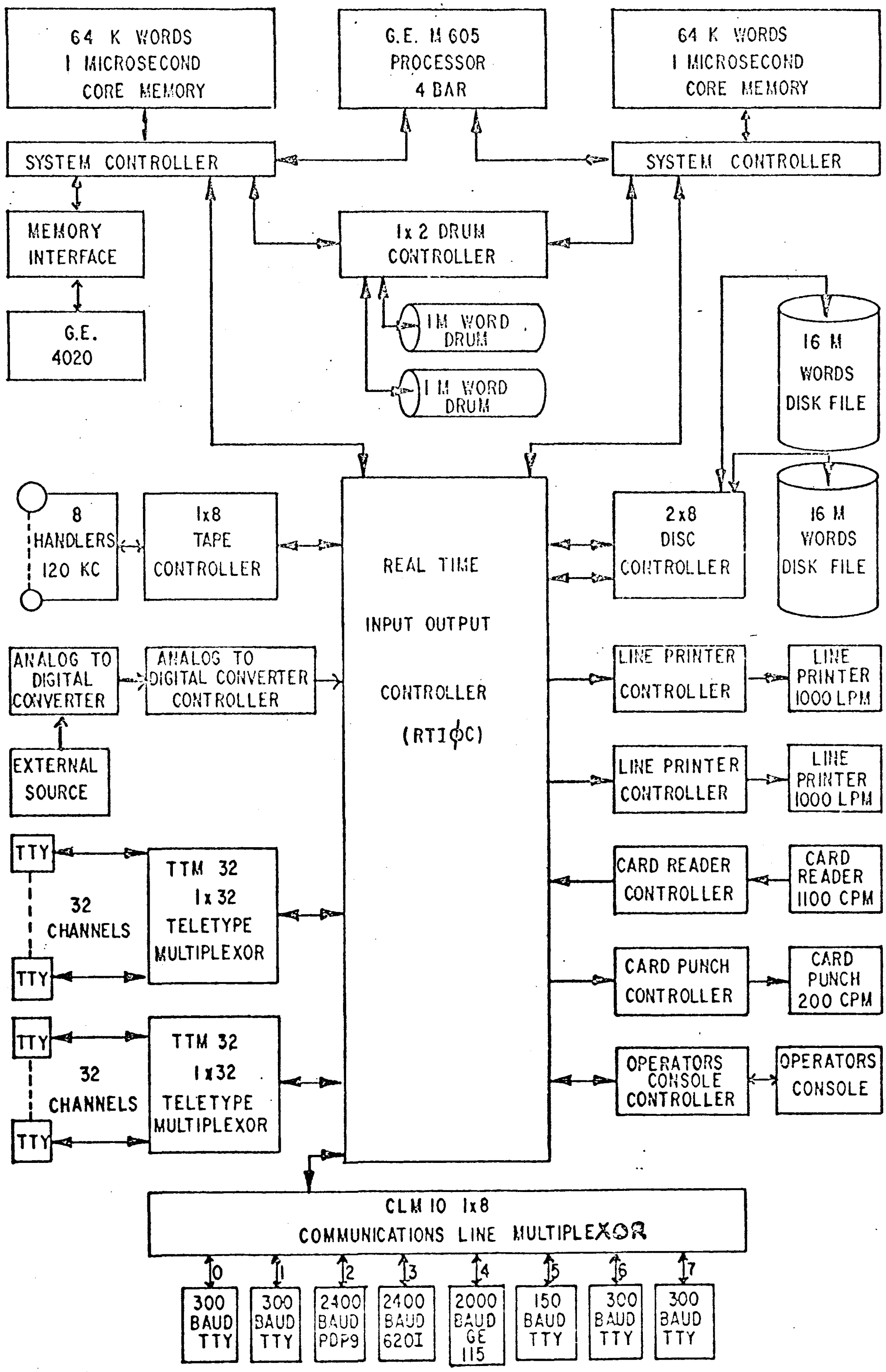

64 K WORDS
1 MICROSECOND
CORE MEMORY
G.E. M 605
PROCESSOR
4 BAR
64 K WORDS
1 MICROSECOND
CORE MEMORY
SYSTEM CONTROLLER
SYSTEM CONTROLLER
MEMORY INTERFACE
G.E. 4020
1x 2 DRUM CONTROLLER
1M WORD DRUM
1 M WORD DRUM
16 M WORDS DISK FILE
16 M WORDS DISK FILE
8 HANDLERS 120 KC
1x8 TAPE CONTROLLER
REAL TIME INPUT OUTPUT CONTROLLER (RTIΦC)
2x8 DISC CONTROLLER
ANALOG TO DIGITAL CONVERTER
ANALOG TO DIGITAL CONVERTER CONTROLLER
EXTERNAL SOURCE
LINE PRINTER CONTROLLER
LINE PRINTER 1000 LPM
LINE PRINTER CONTROLLER
LINE PRINTER 1000 LPM
TTY
32 CHANNELS
TTY
TTM 32 1x 32 TELETYPE MULTIPLEXOR
CARD READER CONTROLLER
CARD READER 1100 CPM
CARD PUNCH CONTROLLER
CARD PUNCH 200 CPM
TTY
32 CHANNELS
TTY
TTM 32 1x32 TELETYPE MULTIPLEXOR
OPERATORS CONSOLE CONTROLLER
OPERATORS CONSOLE
CLM 10 1x8 COMMUNICATIONS LINE MULTIPLEXOR
0 1 2 3 4 5 6 7
300 BAUD TTY
300 BAUD TTY
2400 BAUD PDP9
2400 BAUD 620I
2000 BAUD GE 115
150 BAUD TTY
300 BAUD TTY
300 BAUD TTY

FIG. I-1

Master and Slave Modes of Operation

The R&DC operating system utilizes both the Master and Slave modes of operation which are available with 600 machines.

Programs which run in Master Mode have access to the entire memory, may initiate peripheral and internal control functions, and do not have base address register relocation applied. Only the Master Mode Executive is allowed to execute in Master Mode.

Programs run in Slave Mode have access to a limited portion of memory and cannot generate peripheral control functions. USER PROGRAM'S WILL ONLY BE ALLOWED TO EXECUTE IN SLAVE MODE.

Structure of the Operating System

The operating system can logically be divided into three categories:

- The Multi-Level Executive
- The File System
- The Primitive Commands

A brief discussion of each of these categories is given in the following paragraphs.

The Multi-Level Executive

The System Executive consists of three levels

1) The Master Mode Executive
2) The Slave Mode Executive and Listener
3) Sub-Operating Systems

The Master Mode Executive is that portion of the executive which executes in master mode.

Its principle function is the execution of system input/output primitive commands. The master mode translates these primitive commands, expressed in terms of logical parameters, into commands recognized by the hardware. Master mode services the interrupts generated and returns to the user the physical and logical status of the operation.

Another function of master mode is the running of processes. It sets the BARS to the squeeze or normal settings, returns all traps outstanding to the process, and sets the process into execution at the appropriate location.

Other Master Mode functions are fault handling, disk and drum allocation, and hardware malfunction servicing.

The Multi-Level Executive (Continued)

The Slave Mode Executive directs the allocation of processor(s) and memory to processes and has the real control over the system. The slave mode executive decides process scheduling, determining when a process should be swapped out and the next process to be swapped in. It also executes the control and file system primitives and maintains the file system catalog.

The Sub-Operating Systems provide direct interfaces to the user. These systems will provide the teletype user with interfaces similar to the Desk Side Computer System or Mark 2 and the batch user with a system similar to the GECOS Operating System.

The File System

The File System allows the user to store and retrieve data and programs from a mass storage file by logical name rather than by physical device and address. The File System provides protection against improper use of saved information and also allows simultaneous access to the same file by several users. The File System also provides input/output capabilities with conventional peripheral devices in a similar manner to accessing a mass storage file. The user also is able to access unusual peripherals and remote computers through the file system. Finally, the File System provides a mechanism, the event, by which different processes can communicate with each other.

The Primitive Commands

The Primitive Commands may be viewed as a set of macro calls, available to slave mode processes, which effectively extend the capabilities provided by the hardware. In particular, they relieve the user of the need for dealing directly with peripheral devices.

Processes, Segments, and the State Vector

The process is the basic unit of activity in the system. The notion of a process can perhaps best be described by enumerating its characteristics.

A process (a son) is created by another process (its father) by the system Spawn Primitive. All processes are descendents of a system process called the Listener which is the only process running when the system is first initialized.

A process has the following entities associated with it:

1) From 1 to 4 segments
2) A set of items (files and events) which the process may access.
3) A state segment

When a process is spawned, the father can specify the contents of from 1 to 4 contiguous areas in core, each of which is to have a Base Address Register set around it. The contents of the contiguous areas of core associated with the process in this manner are called the segments of the process. A segment may be either a core image of a father's file or a segment of the father. When a process is in execution, all of its segments must be in core. The manner in which a slave mode program addresses its segments is described in the discussion which follows on Base Address Register usage.

The files and events which are accessible to the spawned process, and the type of access allowable, are also specified by the father when the process is spawned.

The state segment is a unique block of data, associated with each process, which is accessible only to the executive. The state segment is created when the process is spawned and is used to store the information needed by the master and slave executives to run the process.

A more detailed description of the state segment is given in the discussion on Basic System Tables in Section IV.

Base Address Register Usage and Slave Mode Addressing

All addresses for programs running in slave mode are relocated relative to an address which is specified in one of the systems 4 Base Address Registers (BARS). Before this relocation occurs, the slave address is compared with a maximum length setting which is also contained in the BAR. (See Fig. I-2A). If the slave program attempts to address a location which falls outside of this maximum length, a memory fault occurs. A BAR also provides a bit which when set prohibits the slave mode program from doing a write operation in the area defined by the BAR. The BAR selection is determined by bits 0-1 in the slave address as summarized below.

Slave Address (Bits 0-1)	Address Range	Associated BAR	Associated Segment
00	000000-177777	B0	0
01	200000-377777	B1	1
10	400000-577777	B2	2
11	600000-777777	B3	3

The segments associated with a process correspond to the 4 BAR settings. That is, segment zero is associated with the memory bounded by BAR 0, segment 1 with the memory bounded by BAR 1, etc. The slave program can reference its segments by specifying an address associated with the corresponding BAR. For example, to reference segment 3 one would specify an address in the range 600000_8-777777_8.

Note that the 18 bits which are allocated for a slave address enable the slave mode program to address a virtual memory of 2^{18} words (or approximately 262k). Although the slave mode program can utilize addresses in the full range of the virtual memory, the total length of the program will still be restricted by the maximum amount of core memory which the system will allocate to his process.

The interpretation of the 18 bit relative slave address and the BAR format is shown in Fig. I-2A. A further illustration of the mapping of addresses in the user's virtual memory to actual core locations (via the BAR's) is shown in Fig. I-2B. Note that for BAR utilization core is conceptually divided into blocks of 512 words. Both the BAR relocation origin and the maximum length associated with a BAR is specified in terms of these 512 word blocks.

CORRESPONDENCE BETWEEN BASE REGISTER MAPPING AND ADDRESSING

IN THE USER'S SLAVE PROGRAM

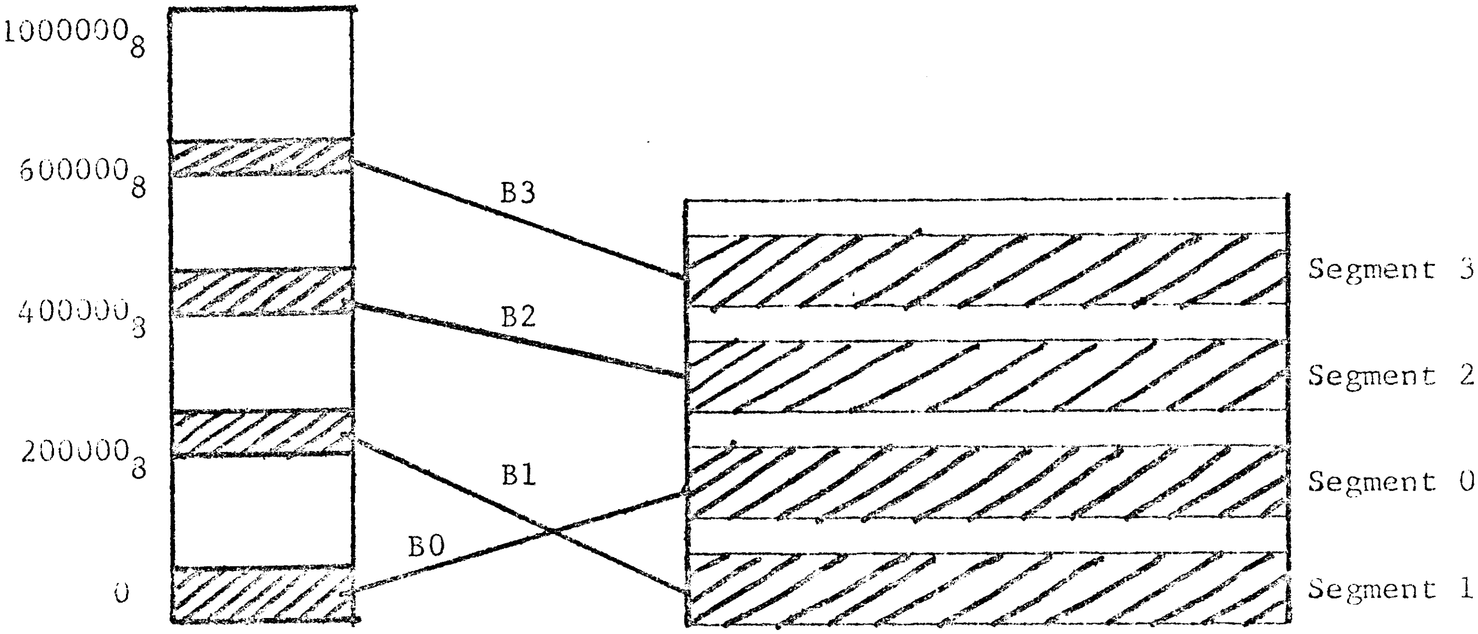


FIG. I-2B

2	7	9
0 Selects the BAR (0,1,2, or 3) 1	2 Block Number (512 Word Blocks) 8	9 Block Index 17

INTERPRETATION OF THE 18 BIT RELATIVE SLAVE MODE ADDRESS

9	1	8
0 Origin of BAR Setting (in 512 word blocks) 8	Write Inh. Bit 9	10 Maximum Length (in 512 word blocks 17

FORMAT FOR A BASE ADDRESS REGISTER (BAR)

FIG. I-2A

FAULT AND INTERRUPT HANDLING

Fault Handling

When a fault occurs, control is automatically transferred by hardware from the process in execution to the appropriate location in the hardware fault vector. An execute double is done on the two instructions at this location. The further action taken by the Master Mode Executive depends upon whether the process in execution was in default, normal, or squeze mode. The base location of the hardware fault vector is defined by a panel setting and consists of 32 consecutive locations in core.

Default Mode

A process is said to run in default mode when it has not declared a fault vector.* For any fault other than Master Mode Entry or Timer Runout, control is passed to an abort routine in the Master Mode Executive which terminates the process.

Normal Mode

A process is said to run in normal (unsquezed) mode if a slave fault vector has been declared and the squeze primitive has not been issued. In normal mode all faults except Master Mode Entry (MME) and Timer Runout (TRO) are passed back to the slave mode process at the fault vector previously designated by the slave mode process.

This fault vector has a two word entry pair for each fault. The instruction counter and indicators are stored in the first word and control is transferred to the second word. The order of the faults is the same as in the hardware fault vector (See Set Fault Vector Primitive).

Handling -MME- and -TRO- Faults

The MME fault is used to call System Primitives, in normal mode it is not returned to the user fault vector. Control is, instead, transferred to a MME Fault Processor within master mode exec. The MME Fault Processor identifies the MME and then transfers control to the appropriate routine in the master or slave mode executive to process the MME. The return mechanism for MME faults is described in the discussion on Primitive Initiation in Section II.

---

* A slave program declares its fault vector by issuing the Set Fault Vector Primitive. For additional information on the slave fault vector see the write-up of the Set Fault Vector Primitive in Section VI.

If the Timer Runout occurs in the slave exec, the timer is reset and control is returned to the slave exec. If the TRO occurs in a normal slave process, control is passed to the slave exec. The slave exec, schedules and runs the next process (which may be the same one).

Interrupt Handling

When an interrupt occurs, control is transferred, by hardware, to an appropriate location in the hardware interrupt vector, where an execute double is performed in Master Mode. The Master Mode Executive then saves the process registers, timer register, instruction counter and indicators, and invokes an appropriate interrupt handling routine. All outstanding interrupts are processed before control is returned to slave mode. Unless an interrupt has occurred for a special priority ("preemptive") process, control is returned to the original slave process. If a trap for this process has occurred, control is returned to the entry point of the trap. Otherwise, control is returned to the point of interrupt. In either case, the process registers, timer register, and indicators at the point of interrupt are restored. (Note that the interrupts which occur, may or may not be related to an activity of the process which was interrupted.) A more detailed discussion of interrupt handling is given in "Trap Routines and Trap Handling" in Section II.

Squeze Mode

Processes which have declared a slave fault vector can enter a special mode known as squeze mode. The addressing space of the squezed process represents a subset of the addressing space of the unsquezed program and is specified by a SETSQUEZE primitive in unsquezed (normal) mode.

Processes are created in the default mode. They enter normal mode by issuing a Set Fault Vector Primitive. After issuing a Set Squeze Primitive to define the BARs of the squezed program, they can issue an "ENTER SQUEZE MODE" primitive to enter squeze mode. The Master Mode Executive resets the BARs to the squezed values and runs the squeze program. If a fault (except Timer Runout) or an interrupt for the process occurs, the Master Mode Executive resets the BARs to the unsquezed values and transfers to the fault vector or trap word in the unsquezed program. All faults (except Timer Runout) which occur in squeze mode, including MMEs, are passed back to the fault vector in the unsquezed mode. However, a MME issued in the unsquezed mode is treated as a system primitive. The unsquezed process can reenter squeze mode by issuing another "ENTER SQUEZE MODE" primitive.

Squeze Mode and Its System Applications

In the discussion that follows, we shall elaborate upon the squeze and normal modes of execution of a slave process, and then indicate applications in which the squeze mode may profitably be employed. As we shall see, these applications are those in which a slave process desires to execute a restricted portion of its segments in a controlled manner.

Recall that a slave process consists of from 1 to 4 segments each having a corresponding BAR set around it. When the process executes in normal (unsquezed) mode, the BAR's are set to their full values, whereas when the process executes in squeze mode some of these BAR settings may be reduced or interchanged (See Fig. I-3). A process goes from normal to squeze mode when it issues the ENTER SQUEZE MODE primitive, having previously specified the new BAR settings for the process with a set squeze primitive.

A squeze mode process returns to normal mode (and the BAR's reset to their unsquezed value) whenever a fault or trap occurs. Note that squeze mode processes are distinguished from processes executing in normal mode in that the MME fault is handled differently. A MME fault which occurs when a process is in squeze mode is returned to the slave fault vector of the unsquezed process(as are all other faults of the squeze mode process) rather than being passed on to the executive as a primitive call. The process may then handle the MME fault in whatever manner it deems appropriate. Traps are also returned to locations in the unsquezed process. The utility for this feature will become more apparent in the examples which follow.

Example 1: Slave Mode DDT

We shall first illustrate the operation of the squeze mode by describing its action for a single base register system in which a Slave Mode Program is to be debugged using a Slave Mode Debugging package (Slave Mode DDT). The DDT program is to intercept all faults which occur in the Slave Mode Program and take an appropriate action.

Let us assume that control is initially in DDT. The single BAR is set around both DDT and the user program and the processor mode is "normal". Before returning control to the user program, DDT issues the SQUEZE primitive which resets the BAR around the user program, and in addition sets a flag which indicates that the process is now running in "squeze" mode. Control is then transferred to the user program which runs in the normal manner until a fault occurs. Whenever a fault occurs, the system fault processing routine notes the flag which indicates that squeze mode is in effect.

The system fault routine, therefore, resets the BAR about both DDT and the user program, clears the squeze mode flag to indicate that the process is now in "normal" mode, and returns control to the appropriate location in the DDT fault vector. The DDT routine proceeds to service the fault. If the fault was not a MME, DDT reissues the squeze primitive and transfers to the fault vector in the squezed program. If the fault was a MME, DDT validates it and converts certain arguments (such as addresses) to their unsquezed values. It then reissues the MME to the exec with the new arguments. When DDT has finished, it reissues the SQUEZE primitive which returns control to a specified location in the user program, resetting the BAR and squeze mode flag. The user program continues in the manner indicated above until the next fault occurs, etc.

The principal steps in the squeze mode procedure are summarized in Fig. I-4.

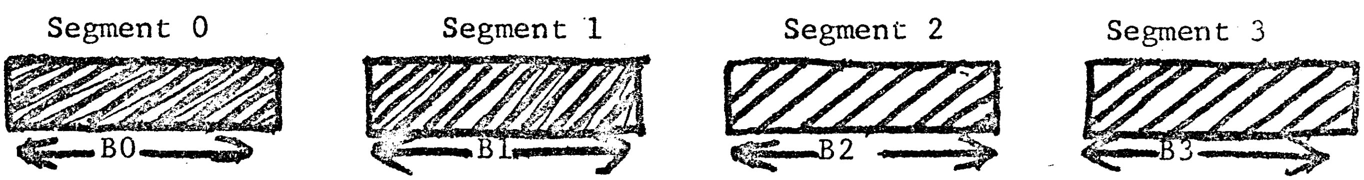
Example 2: Implementation of GECOS Interface

Consider the problem of developing an interface for code, such as GECOS, whose MME conventions do not conform to those employed by the R&DC system. The squeze mode feature enables such already existing code to be run under the R&DC system with a minimum of reprogramming.

Essentially the interface runs GECOS in squeze mode with the BAR's appropriately set about the GECOS system. Whenever a MME fault occurs, control is transferred to the interface which interprets the GECOS MME and then reissues it in the correct format for the operating system. Control is then returned to the GECOS program (by means of the SQUEZE

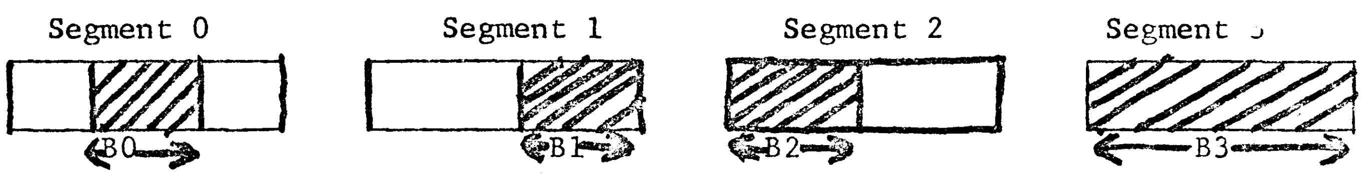
BAR Settings for Normal and Squeze Modes of Operations for a Slave Process

Normal Mode:



1) BAR's set to full value
2) MME's interpreted as primitive commands and passed to the executive

Squeze Mode:



1) BAR's set to reduced value
2) All faults (including the MME) returned to the fault vector of the slave process for handling in unsquezed mode
3) All traps returned to locations in unsquezed mode

Transition from Normal Mode Execution to Squeze Mode Execution

Process is executing in Normal Mode

Process issues SQUEZE Primitive

Process now executes in Squeze Mode

Fault or Trap occurs

Process Resumes execution in normal mode

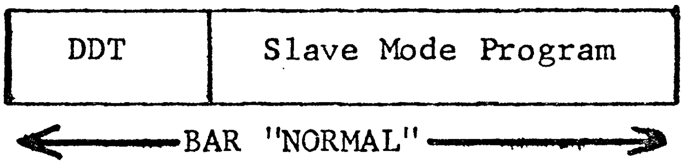
Process Handles Fault or Trap

FIG. I-3

primitive) which resumes execution in squeze mode. The same procedure is repeated for each MME fault which GECOS generates.

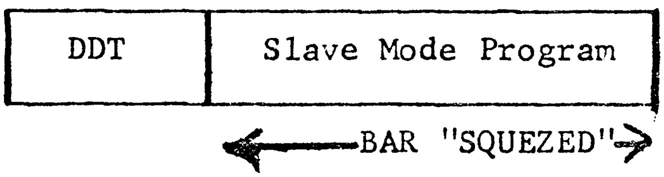
The important point to be observed is that most of the required programming is for the interface. Only a minimal change to the existing code, in this case GECOS, is required.

(a)



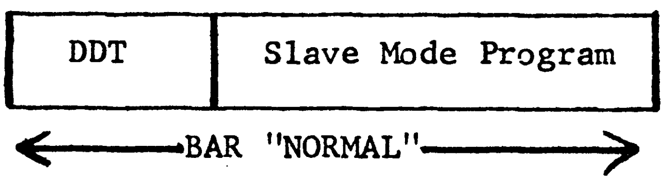
1. Control is in DDT
2. Process Mode in "NORMAL"
3. BAR is set to unsquezed value.

(b) The SQUEZE primitive is issued:



1. Control is in Slave Mode Program
2. Process Mode is "SQUEZE"
3. BAR is set to squezed value.

(c) A fault occurs in slave mode program:



1. Control is returned to DDT
2. Process Mode is reset to "NORMAL"
3. BAR is reset to unsquezed value

(d) After DDT has processed the fault, step (b) is performed and the procedure repeats itself.

Fig. I-4

PRINCIPAL STEPS IN THE SQUEZE MODE PROCEDURE

Section II

Primitive Commands and Trap Handling

PRIMITIVE COMMANDS AND TRAP HANDLING

Introduction

This section describes the considerations involved in using the primitive commands which are available with the R&DC 600 operating system. These commands, which have been implemented as a set of system macros, provide the user with services beyond those provided directly by the hardware. Essentially they supply a mechanism whereby slave programs can issue requests to the executive for services. The macros which initiate the system primitives are available for GMAP assemblies and can be employed with any slave program designed to run under the operating system.

It is convenient to divide the primitive commands into the following categories:

1) I/O Commands

   These commands, such as the reading and writing of sequential and random files, enable the user to perform input/output operations which are expressed in terms of logical file parameters.

2) File and Event Commands

   The file commands provide the user with an interface to the file system. They allow him to open cataloged and scratch files, to catalog a file or directory, to unsave a file or directory, etc. In short, they enable a user to create a file structure and manipulate the files within this structure.

   The Event commands provide a mechanism for interprocess communication. They allow a process to cause an -event-, to be notified when an -event- occurs and to create and catalog -events-.

3) Control Commands

   These commands provide the user with the ability to create, terminate, and block processes, re-adjust base register settings to accommodate non-standard software packages, make memory requests, etc.

A summary of the primitive commands is given at the beginning of Section VI.

Primitive Initiation

A typical macro for a primitive call consists of:

1) A set of instructions which load parameters required by the primitive into appropriate registers. (The primitive is identified by a code number which is loaded into X0).

2) The MME instruction. The address portion of this instruction is ignored by the executive.

The MME (Master Mode Entry) instruction is reserved for primitive initiation and causes control to be transferred from the slave process to the master mode portion of the executive. Associated with each primitive call is generally (but not always) a trap address. The trap address defines the starting location of a corresponding trap routine, which is a block of code that the programmer must specify with the primitive call. Subsequent to the completion of a primitive, the executive stores status in, and returns control to, the associated trap routine. Note that in general, a primitive operation may not be completed when control is returned to the slave process. Hence, the slave program is free to continue its execution while awaiting the primitive completion. When the primitive is eventually completed or aborted due to an error, the slave program is interrupted and the associated trap routine is entered. Upon exiting the trap routine, control may either return to the location at which the slave program was previously interrupted or to an arbitrary location in the slave program. The location to which a trap routine exits depends upon the coding for the trap routine and will be discussed in detail in the following sections.

Illustrations showing the flow of control when a primitive command is issued and when an interrupt occurs are shown in Figs. II-1 and II-2. A detailed discussion of these diagrams is deferred to subsequent paragraphs in this section.

Trap Routines and Trap Handling

The trap routine is located at the trap address specified by the primitive call and consists of a block of code (of at least four words) defined within the slave program. The fourth word is the entry location with the three preceding words set aside for information to be stored by the executive. The format for a trap routine is shown in Fig. II-3, and is summarized in Fig. II-4.

The first two words are reserved for status information which the executive returns. A logical status code, which is returned in the lower half of Word 0, indicates either the success (=0) of the primitive operation or a possible error code. The remainder of the first two words is reserved for optional return information which is relevant to the particular primitive. The logical status code is either specified in the discussion of the primitive in Section VI or in Appendix A.

The third word is used by the executive to store the exit location for the trap routine. By returning through the exit location (e.g., by the instruction RET WORD_3) the trap routine enables the slave program to ultimately resume execution at the point it was interrupted. The location which the executive stores in the exit may be either

a) The instruction counter plus one and indicators at the time the slave program was last interrupted (i.e. the executive effectively does an "STC1 WORD_3" at the point of interrupt).

b) The entry location of another trap routine.

The information so stored is related to the manner in which the operating system handles interrupts, and requires some elaboration. Whenever a primitive operation is completed, the process currently in execution (which is not necessarily the process which issued the primitive) is interrupted and a trap occurs. An entry for the completed primitive is then added to a trap queue which is associated with the process that issued the primitive. The entries on this queue (the outstanding trap queue) consist of all traps which have occurred since the issuing process was last interrupted. Whenever a process with traps outstanding resumes execution, control is first transferred to a trap routine which corresponds to one of the entries on the outstanding trap queue. If there are no traps, control resumes at the interrupt location. The trap routines which correspond to the remaining entries on the outstanding trap queue are linked via the mechanism of the exit location which the executive has stored in Word 3. (See Fig. II-5). The exit location for each trap routine points to the entry location for the next. The exit location for the last trap routine contains the instruction and indicators at the time that the slave program was last interrupted. It should be noted that a trap routine need not exit to the location which the executive has stored. It can for example transfer to any executable location in the slave program. Timing considerations can, however, cause difficulties in the use of this option, especially when there is more than one trap on the outstanding trap queue. It is therefore advised that the programmer carefully read the remaining information in this chapter before employing other than the normal exit.

The fourth word is the first executable instruction of the trap routine and is the entry location to which the executive transfers control when a trap routine is invoked. The remaining words in the trap routine are optional and can be employed to test status returns, set a flag indicating the trap has returned, etc.

It is possible for several primitive commands to share a common trap routine. However, this is only feasible if the traps for these commands do not occur within the same time interval. (otherwise, status and return information stored by the executive for one primitive command would be over-written by the information for the other). A clever programmer may share the same procedure code of a trap routine while allocating distinct four word blocks for status information.

Flow of Control for a Primitive Operation

Let us now summarize the flow of control when a primitive command is issued (See Figs. II-1 and II-2). In order to focus our attention on the essential points we shall first assume that the command is issued after all other primitive commands have been completed. We shall also assume that the primitive command is one with an associated trap routine, and that this trap routine has been coded to exit to the location which the executive stores in Word 3. After a slave process issues a primitive with a MME instruction, the process is interrupted and control is transferred to the Master Mode Executive. The Master Mode Executive may either service the primitive request completely, simply initiate the servicing (by placing the request on an appropriate queue), or pass the request on to the Slave Executive.

Control is returned to the slave process at one of two possible entry points depending upon whether the primitive operation has been completed or not. If the primitive has not been completed, control returns to the location of the MME plus one. If it has been completed, control is returned to the trap routine, with the location of the MME plus one stored as the trap routine exit.

Note the entry point which the executive returns control to is dependent upon timing considerations and also upon the state of the operating system at the time the primitive command is issued. Hence, the slave program should generally be coded to accept either entry location without error.

If control is returned to the slave program before the completion of the primitive (the more usual situation) the slave process has the option of either going blocked or continuing in execution. If the process goes blocked it is taken off the list of processes which are to be scheduled for shots at the processor. The blocked process is reawakened when the trap for the primitive operations occurs with execution resuming at the trap routine.

If a process does not go blocked it will continue execution in parallel with the primitive operation. When the trap for the primitive occurs, the process will be interrupted and control transferred to the trap routine. Upon exiting the trap routine control will return the location of previous interrupt. If the trap occurs when the process under consideration is not in execution, the trap routine will be executed the first time the process regains control. Upon exiting the trap routine, control is again transferred to the location of previous interrupt.

Let us now drop the assumption that the primitive command is issued after all other primitive commands have been completed. In this case several traps may occur during an interval in which the process is interrupted. Control is returned to the process at one of the corresponding trap routines and is transferred successively to the others.

Upon exiting the last of the trap routines control is returned to the location of previous interrupt. Note that this can only occur if each trap routine is coded to exit to the return location which the executive stores in Word 3. (See Fig. II-5).

The fact that the order in which the above trap routines are processed is not mentioned is not an oversight. Generally this order need not correspond to the order in which the corresponding primitive commands are issued. Hence, assumptions relating to the order in which the trap routines are executed should not be built into the program. Programmers who anticipate having multiple traps outstanding at one time should carefully read the remaining paragraphs in this section for additional details in this regard.

Some Considerations in Programming Primitives and Trap Routines

There are several factors that a programmer must bear in mind while programming the primitives and their corresponding trap routines.

A. The contents of all registers are restored after a primitive is initiated by a MME call. Information which is required for later processing can be kept in registers. However, two primitives (request date & time and request elapsed run time) return results in the AQ registers.

B. The primitive may not be completed when control returns to the issuing process. Since several primitives may be issued before any traps return, more than one trap may be outstanding at any one time. A separate trap address is, therefore, required for each trap that may be outstanding during any time interval. Trap routines may, however, share common codes since only the first four words need be distinct.

C. Primitive commands are not necessarily completed in the order in which the primitive commands are issued. For example, successive commands to read the disc and then print from a data area in core might well be completed in the reverse order. There are two basic reasons why this may be so: 1) It may take more time to execute one primitive command than another, 2) Some primitive requests are placed on queues and the queue lengths will generally be different. Assumptions relating to the order of completion of primitive commands should, therefore, not be built into the programming. Note that execution of the trap routine for a primitive command gives a positive indication that the particular primitive command has been successfully or unsuccessfully completed.

D. The order in which trap routines are executed does not necessarily correspond to the order in which the corresponding primitives were completed. (This is a consequence of the trap linking procedure of the Executive and is unavoidable). Hence, one should not assume that the execution of one trap routine presupposes the execution of some other.

E. The programmer should beware of reissuing a primitive within the trap routine for that primitive, since the second trap might occur before the trap routine has been exited. The routine must either be reentrant or must contain a lock to prevent being reentered.

F. The executive stores the exit location and indicators into WORD_3 of a trap. This cannot be zero. Therefore, the programmer can use WORD_3 as a flag to indicate whether the trap has occurred by zeroing WORD_3 before issuing the MME, and then testing it for non-zero.

Conventions for Programming Trap Routines

Recall that several traps may be on the trap queue waiting for processing when an interrupted process resumes execution. In order that all of the corresponding trap routines be executed and control then returned to the process at the point of previous interrupt, the following conventions in writing trap routines must generally be adhered to:

1. Each trap routine must be terminated by a return to the location stored in the third word of the routine (i.e., via a RET WORD_3). (However, see last paragraph on page 9 in case of squeze mode.)

2. All registers which are utilized by a trap routine must be saved in a user data area upon entering the trap routine and restored upon exiting.

Some clarification of these requirements is in order (See Fig. II-5). The third word in a trap routine contains return information and is supplied by the system executive. After a process is interrupted by the occurrence of a trap, the instruction counter and indicators are stored in WORD 3 of the trap routine and control is transferred to the trap routine. The terminating instruction, RET WORD_3, restores control to the process at the point it was interrupted.

If several traps are to be processed, a pointer to the next trap routine in the linking will instead be stored in WORD_3. The terminating instruction, RET WORD_3, then transfers control to the next trap routine in the linking. In order that all such trap routines be executed with control finally returning to the process, ALL trap routines must be terminated with a RET WORD_3. (Recall from the previous section that the trap routines are not executed in an order which the user can predetermine.

Let us now examine briefly why a trap routine must save and restore the registers which it modifies. When a process is interrupted, the system automatically saves the registers. The system ultimately restores these registers when the process resumes its execution. The restored register settings are, however, those which were in effect when the process was last interrupted. However, if any traps have since returned, the process resumes its execution, not at the point it was interrupted, but in a trap routine. It is, therefore, the responsibility of each trap routine to restore the initial settings upon exit, so that ultimately the interrupted process resumes with the correct register settings. (This is not necessary if the trap routine does not modify any registers).

If an interrupt occurred in squeze mode, bit 35 is set to 1 in WORD_3 of the last trap in the chain. A process which utilizes squeze mode must check this bit in every trap before performing a RET WORD_3. If the bit is found on, the process must instead squeze to the location in WORD_3.

FIG. II-1

Flow of Control for a Primitive Command

Slave process is in execution

Arguments for primitive are loaded into appropriate registers, including a code number which identifies the primitive.

MME instruction is issued.

Master Executive takes control and either
a) completes servicing of primitive
b) initiates (but does not complete) servicing
c) passes request to Slave Executive

No Has primitive operating been completed Yes

Return control to slave process at the location of the MME plus one

Yes Does slave process go blocked No

Process is removed from scheduling queue and awaits completion of primitive.

The primitive operation is completed

Slave process is reawakened

Control returned to trap routine.

Slave process continues execution in parallel with primitive operation.

Slave process is interrupted when primitive operation is completed.

Control returned to slave process at trap routine.

Trap routine is executed and exits to interrupt location.

Control returned to trap routine in slave process

Trap routine is executed and exits to location of MME plus one.

Note:

For the flow of control shown it is assumed:

1) that the primitive command is one which has an associated trap routine
2) that the trap routine is coded to exit to the location which the executive stores in the exit word.
3) There are no other primitive commands which have not been completed.

Flow of Control when a Slave Process is Interrupted

Slave Process is in execution

Process is interrupted at instruction L

Registers at point of interrupt saved, and control transferred to the Master Mode Executive.

Interrupt is serviced by the Executive

Registers of slave process restored

Have any traps for the slave process occurred since the slave process was interrupted.

No

Return control to slave process at location L+1.

Yes

Return control to the slave process at a trap routine.

Slave process executes the trap routine for each trap that has returned while it was interrupted.

Last trap routine exits to location L+1.

Note:

1) For the flow of control shown, it is assumed that all trap routines exit through their exit location.

2) The flow of control shown holds equally well for an interrupt caused when a MME instruction is issued.

FIG. II-2

FORMAT FOR A TRAP ROUTINE

	0 — 17	18 — 35	
WORD 1	(Optional) status information returned by executive	Logical Status Code	STATUS WORD1
WORD 2	(Optional) status information returned by executive		STATUS WORD2
WORD 3	Exit Location (returned by executive)	Indicators at point of interrupt (or zero)	EXIT
WORD 4	Entry location - the first instruction of the trap routine		ENTRY
WORD 5 . . . . WORD N	Remaining Instructions (optional)		

where WORD 1 is located at the trap address.

FIG. II-3

Format for a Trap Routine

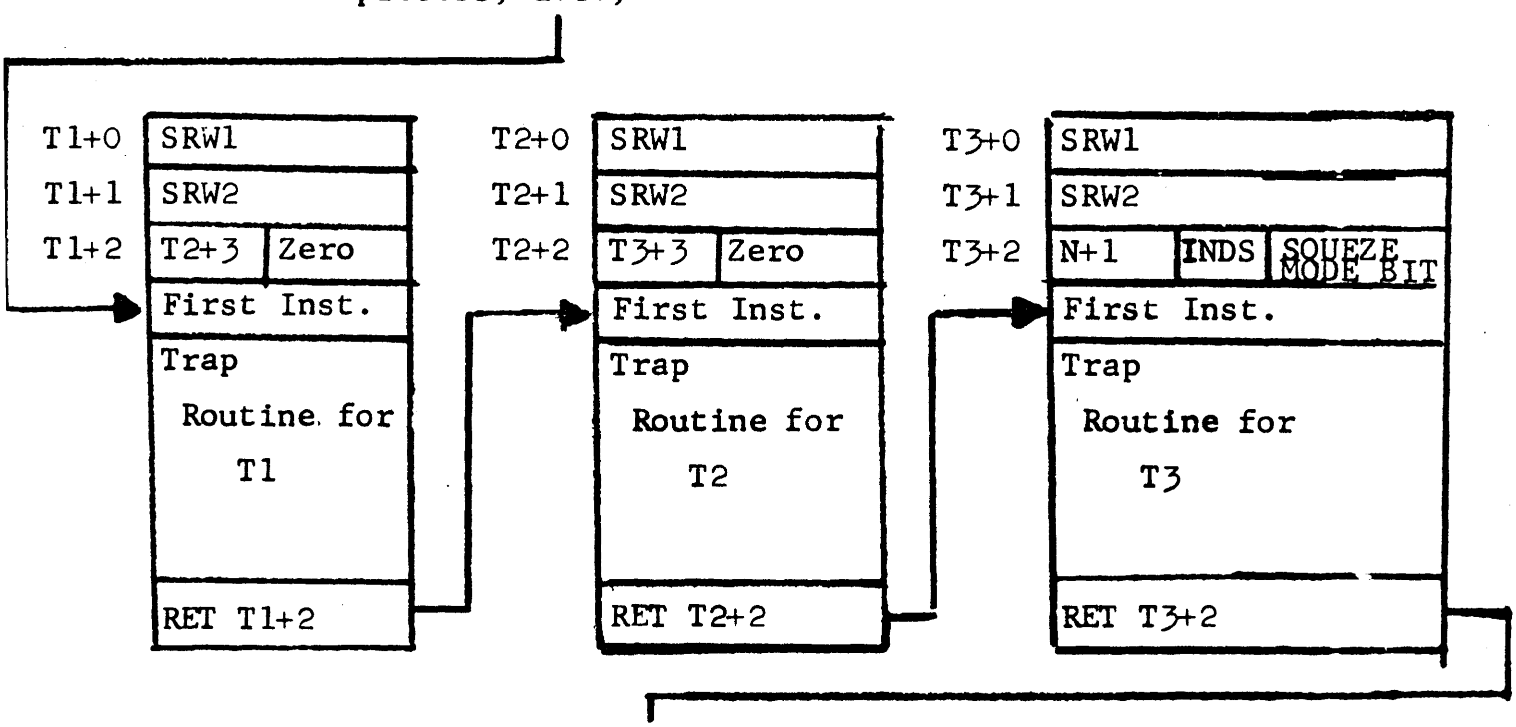
WORD 1		STATUS RETURN WORD 1
	Bits 0 -17	Optional primitive dependent information, returned by the executive.
	Bits 18-35	Logical Status Code - the code is zero if the primitive operation was successful. Otherwise, a primitive dependent error code number is given. The I/O primitives share a common set of error codes, as do the File and Event primitives.
WORD 2		STATUS RETURN WORD 2
	Bits 0 -35	Optional primitive dependent information, returned by the executive.
WORD 3		EXIT LOCATION
		The executive stores an exit location which the trap routine may utilize for returning to the slave program.
	Bits 0 -17	The following exit locations may be stored here: a) The interrupt (or MME) location plus one b) The first instruction of another trap routine
	Bits 18-35	Dependent upon (a) or (b) above a) The indicator settings at the point of interrupt. If the interrupt occurred in squeze mode, the interrupt location is the squezed mode address, and Bit 35 is set on. Otherwise Bit 35 is set off (zero). b) Zero
WORD 4		ENTRY LOCATION
		The executive transfers control to this location when the trap routine is entered. This is the first executable instruction of the trap routine.
WORDS 5-N		REMAINING INSTRUCTIONS OF TRAP ROUTINE (OPTIONAL)
		These instructions constitute the body of the trap routine. The trap routine may return control to the body of the slave program by doing a RET WORD_3.

FIG. II-4

Fig. II-5

TRAP LINKING IN A USERS PROGRAM

User Process in Execution
:
INSTR. N-2
N-1
N
:
Process is interrupted after instruction N (e.g., by a timer runout) and loses control of processor.
:
While process is not in execution, traps with trap routines located at T1, T2, T3 occur.
:
Control is returned to process at entry instruction of T1, the routines for T1, T2, and T3 executed, after which control is returned to instruction N+1 of the interrupted process, i.e.,



Instruction N+1
N+2
N+3
:

Section III

I/O Primitive Overview

I/O PRIMITIVES OVERVIEW

Summary of Macro Calls for I/O Primitives

The primitives classified as I/O, their code identification, and their macro calls are summarized below:

Code	Macro Call		Description
04	READ	TRAP,FRN,CORELOC,N,MODE	Sequential Read
05	APEND	TRAP,FRN,CORELOC,N,MODE	Sequential Write
06	RRF	TRAP,FRN,FILELOC,CORELOC,N	Random File Read
07	WRF	TRAP,FRN,FILELOC,CORELOC,N	Random File Write
08	SCR	TRAP,FRN , FILEOC	Scratch File
09	SPTR	TRAP,FRN,N	Set File Pointer
10	RQST	TRAP,RN	Request Status

These primitives supply a slave program with an interface to both random and sequential devices as indicated.

Random Devices - Drum, Disc

Sequential Devices - Card Reader, Card Punch, Line Printer, Magnetic Tapes, Teletypewriters, Operators Console, Communication Lines.

Element Size and Maximum Transmission for I/O Operations

Each input-output device type has associated with it a unit size, which characterizes the number of bits (or words) employed in an actual data transfer. Primitive commands, however, transfer data in terms of an element size which the slave program specifies for a file at the time it is opened. The element size must be some integral multiple of the unit size.

For each device type there is also a maximum number of data units that can be transferred in response to a single primitive command. This maximum number is determined both by the device and by the software.

The unit sizes and maximum amount of data transmission for each of the I/O device types are summarized in Tables III-A and III-B.

The Addressing Mechanism for Sequential and Random Operations

For all I/O commands, files are referenced by a File Reference Number (FRN) which is returned to the slave program when the file is opened. Associated with each file is an element size which is also specified when the file is opened.

A file can, therefore, be regarded as a linear array of elements, each element of the file having the same specified element size. The file element represents the basic unit in which all input-output operations are expressed. The manner in which an element is referenced depends upon whether the primitive operation is random or sequential.

For random operations, successive elements of a file are associated with consecutive integers, the first element of the file being associated with the integer 0. The starting location for a data transmission can, therefore, be specified by an integer corresponding to the relative position of some element in the file. For example, one can perform a random read of N elements from file "FRN" starting at element M, where M and N are integers.

For sequential operations, the starting location for a read is specified by a current read pointer, while the starting location for an append is specified by an end of file pointer. The read and end of file pointers are discussed further in the write-up of the READ and APEND Primitives.

Treatment of Mass Storage and Physical Device Files

The I/O primitives can be employed with either mass storage or physical device files. The mass storage files are those located on the drum or disc and can be accessed by both the sequential and random commands.

The physical device files are associated with devices such as Card Reader, Card Punch, Line Printer, Magnetic Tapes, and Teletypewriters. Physical device files can also be regarded as consisting of a linear array of elements. The element size will, as previously noted, be some integral multiple of the basic data unit which a device can transmit. Unlike the mass storage files, reads and writes on physical device files can only be done sequentially.

In order to issue an I/O primitive for a physical device file, the device type corresponding to the file must first be opened. This is performed by the OPEN primitive; this primitive will return a file reference number for one of the available devices of the type specified by a treename. All future references to this physical device file will be made in terms of this file reference number. Provision is made to enable a particular unit of a device type to be opened, e.g. a particular line printer or teletype. (See Fig. III-1).

Note that the same sequential I/O primitives are used with both physical devices and mass storage files. For the physical device files, a physical interpretation (such as the current position of a magnetic tape head) is given to the current read and end of file pointers. This interpretation is different for each device type and is summarized in the remarks in Tables III-A and III-B.

Mode Parameter for I/O Primitives

I/O primitives for certain device types require a device dependent mode parameter to further specify the data transaction which is to occur. For example, the Card Reader can read cards in binary, bcd, or mixed, depending upon the setting of the mode. The mode parameter is also used in specifying a particular operation on magnetic tapes.

A summary of the valid mode values for each device type is given in Appendix B.

Status Returns for I/O Primitives

The I/O primitives return device independent logical status (Word 1) and device dependent physical status (Word 2). The same logical status code is employed for all of the I/O primitives and is summarized in Appendix A-1.

The physical status is dependent upon the particular device file which is being referenced and is summarized in the 600 manual for the device in question.

Device Identification Definition

Identification Number	Device Type
1	Disc
2	Drum
3	Operator's Console
4	Card Punch
5	Line Printer
6	Card Reader
7	TTM32-1
8	CLM-10
9	Tape Units
10	Analog to Digital Converter
13	Mem. Interface
14	Events
15	TTM32-2

	27	35
	NUMBER	TYPE
	5	4

FIG. III-1

Device Identification Number

The following information on the device identification number is primarily for the use of programmers working directly on the executive.

Each physical device in the hardware configuration is identified by a 7 bit device identification number.

Device ID

Bits 0-4	Device type unit number
Bits 5-8	Device type identification number

The device type identification number specifies the number to be assigned to a given device type and is summarized above.

The device type unit number specifies the particular physical unit of that device type (i.e. the 22nd teletype).

0 — 4	5 — 8
Device Type Unit Number	Device Type Identification
5	4

MASS STORAGE FILE SUMMARY

Device	Unit Size	Maximum Units per Transmission	Remarks on Starting Location for a Data Transmission
Drum	32 words	64K words	For the drum and the disc, both the sequential and random primitives can be employed.
Disc	32 words	64K words	Random operations can begin at an arbitrary element of the file. The element is specified by an integer index which gives the element's position relative to the beginning of the file. For sequential operations, the starting location for a read is specified by a current read pointer, while the starting location for a write is specified by an end of file pointer.

TABLE III-A

PHYSICAL DEVICE FILE SUMMARY

Device	Access READ	Access APEND	Unit Size	Maximum Units per Transmission	Remarks on Starting Location for a Data Transmission
Card Reader	X		6-bit characters	Binary Mode: 160 characters Hollerith Mode: 80 characters Mixed Mode: 160 characters	The READ starts from the first character of the next card to be read and continues for the number of characters which correspond to the specified number of elements. If less than the maximum number of characters are specified, the remaining characters on the card are ignored.
Card Punch		X	6-bit characters	Binary Mode: 160 characters Hollerith Mode: 80 characters	The APEND starts at the first character of the next card to be punched and continues for the number of characters which correspond to the specified number of elements.
Line Printer		X	6-bit characters	4,096 characters	The APEND starts at the left margin of the current print line. Necessary slew and control characters must be supplied by the user. See Printer Manual for detailed characteristics.
Magnetic Tapes	X	X	1 word	61,440 words	Both the READ and APEND begin at the current position of the magnetic tape head. The set pointer primitive can be employed to modify this starting location.
Teletype-writers Operator's Console	X	X	9-bit characters	128 characters	These devices are not initialized to the left margin so that READ and APEND proceed from the typewriter current carriage position.
Communica-tion Line Multiplex-or	X	X	36 bit words	4096 words	The read starts after the last SYNC character and continues until the subchannel dependent end of record character. The append transmits the words specified.
A to D Converter	X		36-bit words	4096 words	

TABLE III-B

Section IV

File System and File Primitive Overview

FILE SYSTEM AND FILE PRIMITIVE OVERVIEW

Logical Structure

The R and DC file system is, in formal terms, a tree structure of indefinite length whose origin is the system Root Directory. The primary nodes of the tree are the user's highest level directories, referred to as the user's Main Directories. The lower level nodes, if they exist, are subdirectories. The terminal points of the structure are the files (or events). A schematic representation of the file system's hierarchical structure is shown in Figure IV-1.

Files

Files may be classified as mass storage or physical device. The file system views both of these files as linear arrays of elements and is not concerned with the contents. The element may be either a character, word, or block of words depending upon the nature of the file.

Mass storage files are located on the disk and/or drum, and are paged. Storage for a file is allocated dynamically on a -when needed- basis in multiples of either the drum or disc page size. Files located on the disk or drum may be accessed in either a random or a sequential manner.
Physical device files permit access to external storage media such as unit record devices, magnetic tapes, teletypes, and remote computers.

The Directory File

The directory file is a system file which maintains information about all of the catalogued entities. The directory file consists of a hierarchy of directories and is accessible only to the EXECUTIVE.

Directory

A directory consists of a collection of entries called branches. Each branch describes either a file, event, directory, or link. Included in the information for a branch entry is

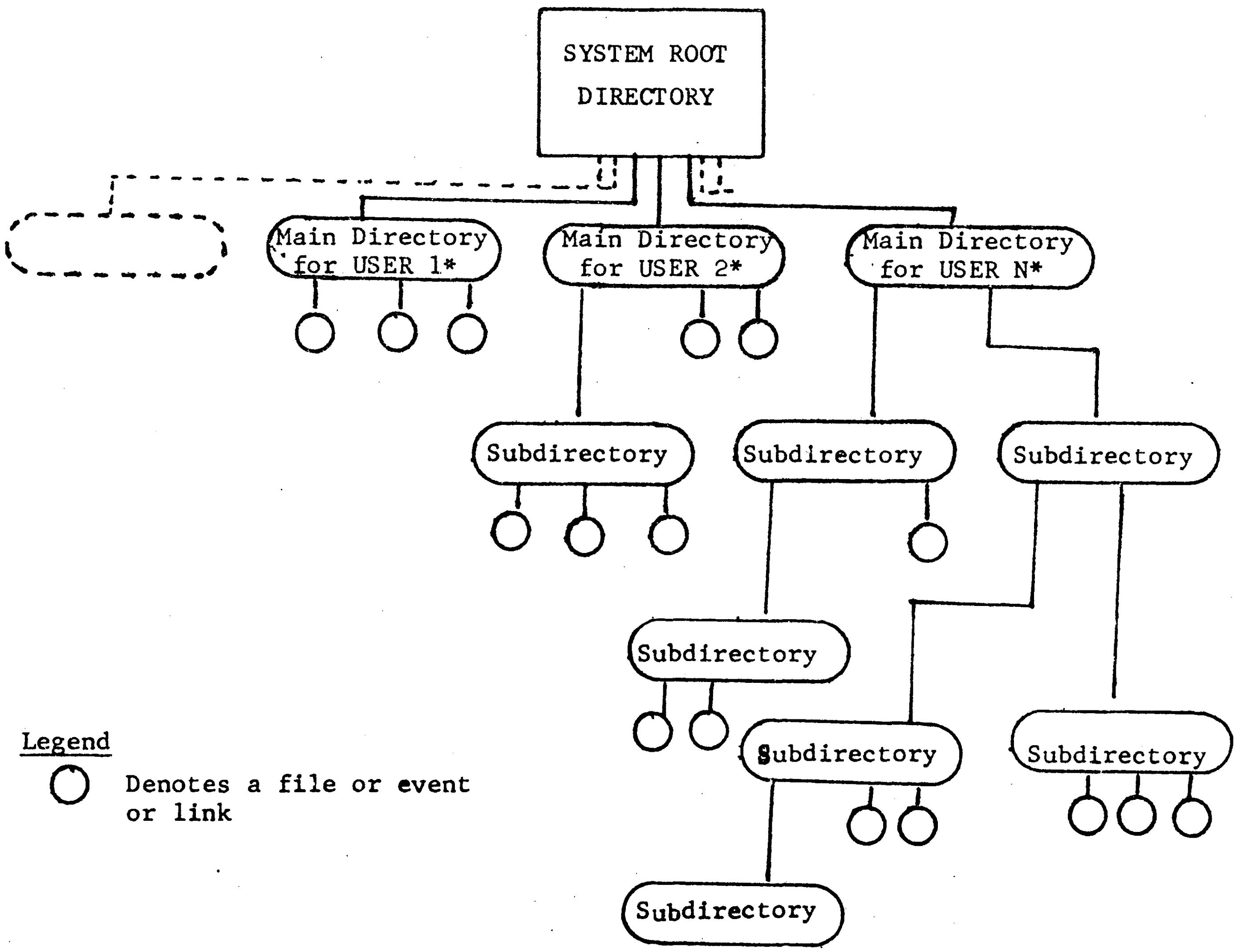
1 - the symbolic name of the entity referred to

2 - the access granted to the entity for system users (except for link)

3 - the password associated with the entity

A detailed description of the information in a directory branch is given in the discussion on the READ BRANCH and READ DIRECTORY primitives.

LOGICAL STRUCTURE FOR THE R&DC FILE SYSTEM



* Identified by the name used for the system root directory entry. All such names must be unique within the system; all subdirectories and file names are qualified by the user's name and the names of any intermediate subdirectories. The system root directory cannot be accessed by the user.

FIG. IV-1

Events

Events are data structures which are used for interprocess communication. They can be catalogued and, in general, handled much in the same manner as files. A more complete description of events will be given in Section V on Event Primitives.

Identifying a File - Tree Name

File or directory names need not be unique in the file system, except that all files emanating directly from a given directory must be uniquely named among themselves. To uniquely identify an individual file in the system, a string of names is given, beginning with the creator's Main Directory name and ending with the file name.

Each successive directory name in the string qualifies the file name, thereby uniquely defining the file. This string is called the tree name. The tree name of any file or directory must include the names of all of the higher-level directories that must be traversed in order to arrive at the desired point.

Note that events and links (to be described) are also referenced by the same tree name conventions as files.

For example FILA in Fig. IV-2 has the tree name USERNO1/DIRA/FILA. In this representation, successive levels of the tree name are separated by a slash. A password can also be associated with the name at a given level. This is represented notationally by appending the dollar symbol and the password to the name component of the level. For example, to associate passwords with USERNO1 and FILA, one would write the tree name as

USERNO1$PASSWORD1/DIRA/FILA$PASSWORD2

Note that this representation which is the same as that used in GECOS 3, is used only for describing tree names. It is similar to, but not the same as, the representation which the programmer will employ in specifying tree names when coding the file primitives.

Links

A link contains the tree name of an entity being pointed to. The entity may be a file, directory, event, or another link. In the latter case, the link chain must eventually terminate with a non-link entry. The tree name specified by the link must begin from the level of a user's main directory. Note that link chains are not allowed to close back upon themselves. For example, the tree name of the link shown in Fig. IV-2, is USERNO2/LINKA. A possible content of this link entry would be the tree name USERNO1/DIRA/FILA.

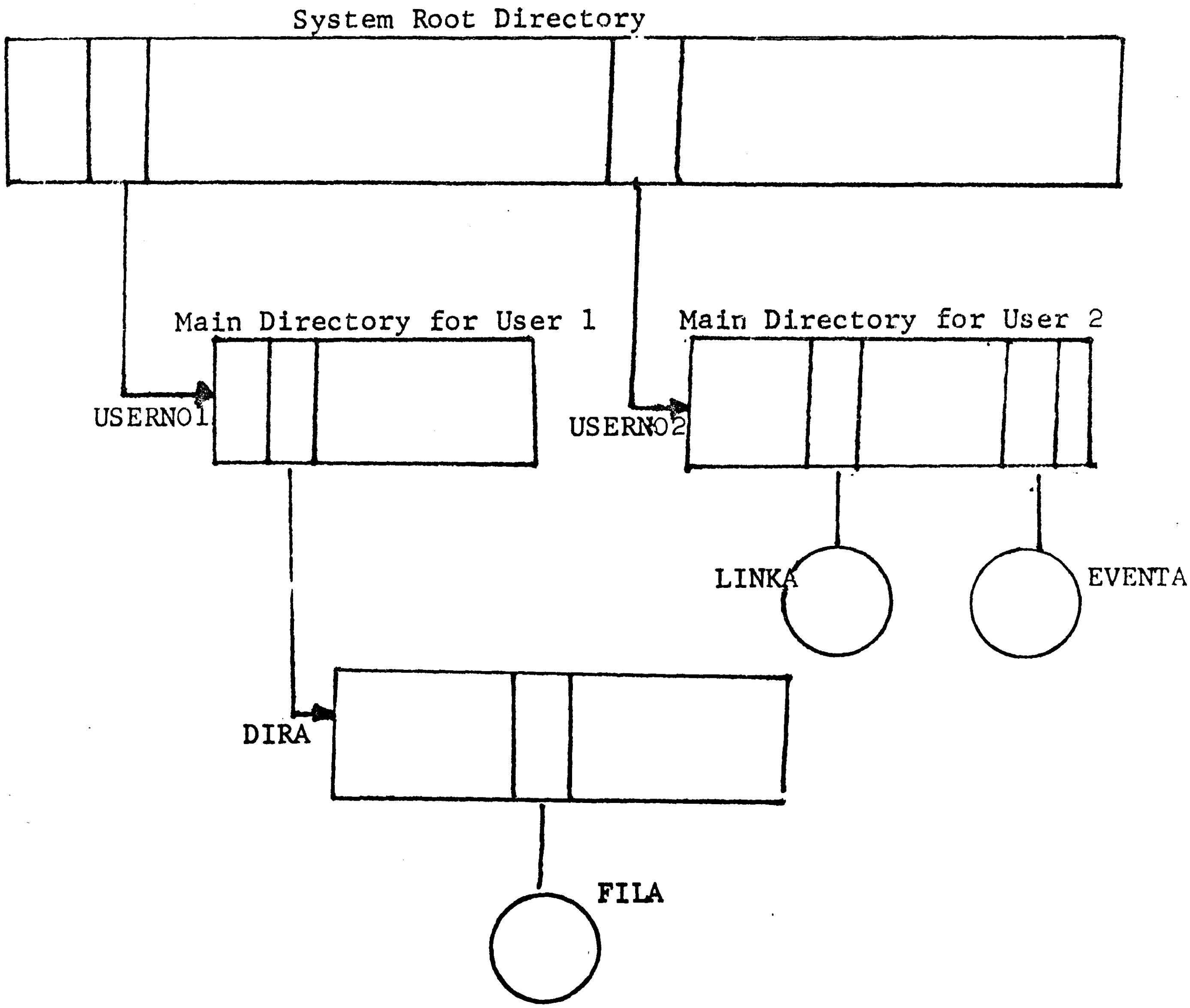
Working Directory

Each process has a current working directory associated with it with respect to which the process can specify tree names. For example, if in Fig. IV-2, the DIRA is the working directory, then the tree name of FILA relative to the working directory is */FILA. (The symbol * denotes that referencing is from the working directory instead of the root directory).

ACCESS

Permission may be granted to other users by the file creator for one or more kinds of access to his catalogued entities. A discussion on access is given in "Parameter Summary for File Primitives", which appears in a subsequent paragraph in this section.

FIG IV-2



Entity in Directory Structure	Tree Name of Entity
Main Directory for User 1	USERNO1
The directory, DIRA, catalogued in the Main Directory for User 1	USERNO1/DIRA
The file, FILA, catalogued in DIRA	USERNO1/DIRA/FILA
The link, LINKA, catalogued in the main directory for User 2	USERNO2/LINKA
The event, EVENTA, catalogued in the main directory for User 2	USERNO2/EVENTA

The Basic System Tables - The AIT, KIT, and State Segment

The following sections describe the basic system tables which the Executive utilizes in performing its functions. These sections are intended primarily for reference and are included to clarify the meaning of the terms 'AIT' and 'KIT' which are employed in the description of certain primitives.

In the discussion that follows, the term 'item' is used to signify a file or event.

The Active Item Table (AIT)

The Active Item Table maintains the global information, required by the executive, for the manipulation of the files, segments, or events which are currently active. There is a single entry in the Active Item Table (AIT) for each such active item.

A file or event becomes active the first time it is opened by a slave process. At this time an entry is made for the file or event in the Active Item Table (AIT). The file or event remains active until it is closed with an attachment count (entry hold count) of zero. At this time the corresponding entry is removed from the AIT.

A file or event may be opened more than once or by more than one process. In this case there is still only one AIT entry, but the attachment count for the entry is updated. In addition, an appropriate pointer to the AIT entry (to be described in discussing the Known Item Table) is set up. (See Fig. IV-3)

A segment becomes active the first time it is specified in the spawning of some process or upon being opened by a process. It remains active until all processes which utilize it have terminated. Segments may also be shared by more than one process. But again there is only a single AIT entry for each segment.

Information Contained in a File or Segment AIT Entry

1) Location of file or segment

   Segment Entry - Core location of segment or the location of the page table for the corresponding segment swap file.

   File Entry - location of page table for file.

2) Maximum and current length (if file)

The Active Item Table (Continued)

3) Number of processes using the segment or file

4) Statistical information on use of segment or file

5) Pointer to a branch in the system directory which points to the page table of the file (catalogued file only).

6) Owner's identification

7) Lock indicator (files only)

Information Contained in an Event AIT Entry

1) Pointer to queue of processes awaiting notification

2) Current and maximum allowable size of queue

3) Number of processes using the event

4) Pointer to corresponding branch (catalogued events only)

5) Owner's identification

6) Lock

7) Maximum time an entry can remain on queue

8) Parameters describing operation of event

State Segment and Known Item Table

The State Segment

Associated with each process is a unique state segment which is created when the process is first spawned. The state segment is used to store the information which is needed by the master and slave executives to run the process. The information in the state segment is local (that is, it applies only to the particular process), as contrasted with the information in the Active Item Table which is global to all processes.

A principal data item contained in the state segment is the Known Item Table (KIT). The KIT contains information concerning the files and events that the process has knowledge of. Because of its significance, the contents of a KIT entry will be discussed separately.

Information in the State Segment

1) Register storage

2) Location of slave process fault vector

3) Timing information

4) Pointer to AIT entry for each of the process' 4 segments

5) Squeze mode information

6) Process ID and father's process ID

7) Count of I/O and file operations underway

8) Count of number of events process is awaiting

9) The Known Items Table (summarized below)

10) Trap information

11) Other Measurement Information

The Known Items Table (KIT)

The Known Items Table has at least one entry for each file and event that a process has knowledge of. Entries are made in the KIT when a process is initially spawned, for each item (if any) passed to it, and subsequently whenever the process opens a file or event (item). The items in the KIT are accessed by means of their -reference number- which has been returned to the process when the item was opened, or which was passed to the process when it was spawned.

The Known Item Table (Continued)

Since a process can open an item more than once, there may be several entries in the KIT which represent the same item. Similarly an item which has a reference number in the KIT of one process may also have a reference number (usually different) in the KIT of some other process. (This occurs, for example, when a father specifies the reference numbers in the father's KIT for items in the son's KIT).

Note, however, that regardless of the number of times an item is referenced by entries from various KIT's, it has but a single AIT entry. Each KIT entry has a pointer to the global entry for the item in the AIT. Hence, different KIT entries which correspond to the same item will all point to a common AIT entry. This mechanism aids in the manipulation and maintenance of items which are common to several processes. (See Fig. IV-3).

Information in a Known Item Table Entry

1) Access permitted to item

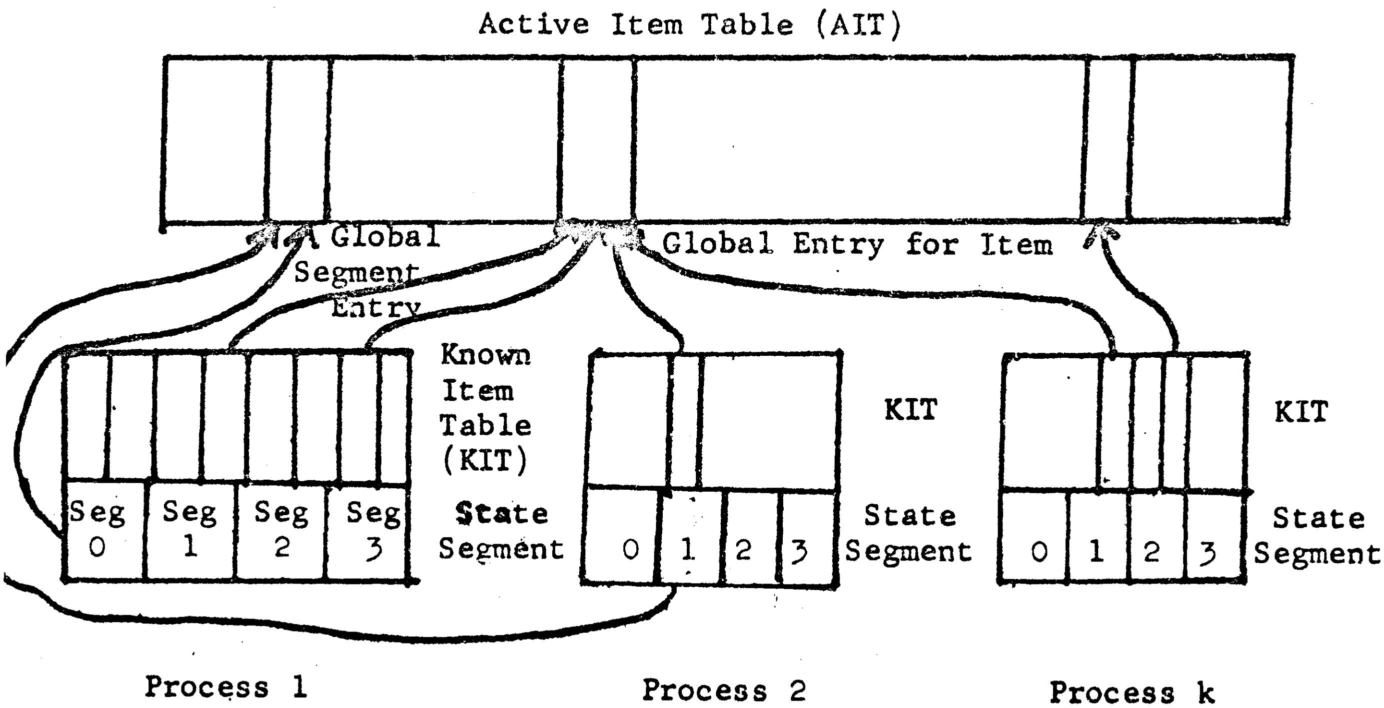
2) Pointer to AIT entry for item

3) Flag to indicate whether process has locked the item

4) Count of number of I/O operations underway (file) or requests for notification outstanding (event).

5) Read pointer (for reading files sequentially).

Relationship Between the Active Item Table,
Known Item Table, and State Segment



* NOTE: Entries in the KIT are referenced by their Item Reference Number, RN, which is used as an index in the KIT. The reference number for the KIT in each process is private to the process.

FIG. IV-3

Summary of Macro Calls for File Primitives

Code	Macro Call	
20	OPEN	TRAP,TREE_NAME,TREE_SIZE,BEHALF,ELSIZE,ACCESS
21	CLOSE	TRAP,RN
22	CATLOG	TRAP,RN,TREE_NAME,TREE_SIZE,SWITCH,UACCESS,OACCESS
23	DESTRO	TRAP,TREE_NAME,TREE_SIZE,BEHALF
24	OPENS	TRAP,DEVID,MAXLEN,ELSIZE
25	UPDATE	TRAP,RN
26	CATDIR	TRAP,TREE_NAME,TREE_SIZE,BEHALF,UACCESS,OACCESS
27	WRACL	TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE
28	RDACL	TRAP,INDEX,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,NUMBER
29	RDDIR	TRAP,INDEX,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE
30	OPENW	TRAP,TREE_NAME,TREE_SIZE,BEHALF
31	RDBRN	TRAP,SYSID,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF
32	RDLNK	TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE
33	WSINF	TRAP,SYSID,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,DELETE
34	CATLK	TRAP,LINK_NAME,LINK_SIZE,TREE_NAME,TREE_SIZE,BEHALF
35	WTBRN	TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF
36	LOCK	TRAP,RN
37	UNLOCK	TRAP,RN

Description of Parameters for File Primitives

Summarized are those parameters which are common to several of the file (and event) primitives. The remaining parameters are described in the individual write up for each primitive.

TREE_NAME

Catalogued items are initially referenced by a tree name which specifies the location of the item in the 600 system directory structure. The parameter 'TREE_NAME' is a pointer to an area in the slave program where the characters in the tree name are stored.

Tree names which consist of more than one level must occupy a contiguous block of storage, with six words reserved for each level. For each level, the first 4 words specify the name and the last two words the password.

The characters of the name and pass word are left justified and all characters must be specified for each level. Trailing blanks (octal 040) must be supplied if the names and passwords are to be compatible with those catalogued by the System Loader from the card reader.

If a password is not desired, a default password of eight ASCII blanks (octal 040) should be specified. Note that the name and password of a given level of the tree name are conveniently defined by the ACI Pseudo Op in the GMAP assembler.

Examples of Tree Name Specification with Passwords

An example showing the default passwords for device files

ACI 6,DEVICE	Name and default password for device
ACI 6,MTAA	Name and default password for any magnetic tape

The above file has the tree name DEVICE/MTAA.

An example of a file with a non default password

ACI 4,ZBROWN	Name of user's main directory
ACI 2,SESAME	Password for main directory
ACI 6,DOCCAT	Name of directory in above with default password
ACI 4,FILE1	Name of file in DOCCAT
ACI 2,SHAZAM	Password for FILE1

The above file has the tree name ZBROWN$SESAME/DOCCAT/FILE1$SHAZAM.

TREE_SIZE — The number of words in the storage area where the characters of the tree name are stored (i.e. the storage area whose starting location is given by 'TREE_NAME'). The number of words specified by 'TREE_SIZE' must be a multiple of 6.

If 'TREE_SIZE'=6 it implies the operation is being performed at the root level of the directory.

RN — The item (file or event) reference number, 'RN', is returned when either a scratch or catalogued item is opened. Most subsequent references to the item will be in terms of this reference number. The reference number, 'RN', is an index to the entry in the Known Item Table of the process which describes the item. The value of 'RN' has meaning only to the process which opened the item. An item which is shared by several processes will generally have a different reference number in each process.

BEHALF — The BEHALF indicator specifies the user identification number which is to be checked in determining whether the requested access to the directory structure will be granted. The BEHALF indicator settings are as follows:

Bit 17 (in behalf word)

1 = ORIGINATOR's ID is to be used (normal setting)

0 = OWNER's ID is to be used

The BEHALF should almost always be set for the ORIGINATOR. The use of OWNER's behalf should be restricted only to system programs accessing proprietary subroutines and then used with CAUTION. An explanation of this terminology is given below.

Each process may have two distinct identification numbers stored in its state vector, one of which is referred to as the ORIGINATOR's ID, the other the OWNER's ID. The ID is a number which the system assigns to each user of the system when the user's main directory is initially catalogued in the system root directory.

Parameter Summary for File Primitives (continued)

The user who signs on via the "HELLO" sequence is called the ORIGINATOR, and the ID corresponding to this user is the ORIGINATOR's ID. When a process is spawned by the LISTENER, the ORIGINATOR's ID is stored in the state vector of the process. All processes which are subsequently spawned by this spawned process will also have the same ORIGINATOR's ID specified. Hence if any such process issues a file or event primitive with the ORIGINATOR's behalf specified, the ID of the user signing on via the "HELLO" sequence is the one which will be checked by the file system in determining whether access is to be granted. Note that a process will always have a non-null entry in its state vector for the ORIGINATOR's ID.

When a process is spawned, a non-null OWNER's ID may also be stored in its state vector. The OWNER's ID is that of the user in whose main directory the file (or segment) spawned as segment zero is cataloged. A NON-NULL OWNER'S ID IS STORED ONLY IF THIS FILE WAS CATALOGED WITH THE OWNER'S ACCESS PRIVILEGE SWITCH SET ON (see catalog primitive). If the owner's access privilege switch is set OFF, or if the file spawned as segment zero is a scratch file, a null OWNER's ID is set in the state vector. A file or event primitive issued with the OWNER's behalf specified will be rejected unless the OWNER's ID is non-null and the access for the non-null ID is valid.

In general files are rarely catalogued with the OWNER's ACCESS PRIVILEGE SWITCH set ON, since any process which can spawn this file as segment zero will be allowed to use OWNER's ID and hence have all access to the directory structure that the owner has. The OWNER's ID is generally useful only for system programs accessing proprietary subroutines.

ACCESS,UACCESS,OACCESS

This parameter specifies the access to be requested or assigned in performing a manipulation of the file structure. The access attributes are represented by a mask, the format of which is summarized in Fig. IV-4.

Whenever a user issues a primitive which involves a tree search on the directory structure, a check is made to determine whether he possesses the requested access. Access privileges are specified for three classes of users:

1) The Owner - the individual in whose main directory the referenced entity is catalogued.
2) A set of users, specifically enumerated, that appear on the Access Control List for the entity.
3) The Universe - all other valid users (excluding the owner and users on access control list)

The OWNER's and UNIVERSAL access (OACCESS and UACCESS) for an entity are defined when the entity is initially catalogued. Entries on the Access Control List are defined by the Write Access Control List Primitive.

The BEHALF indicator (discussed previously) determines whether the ORIGINATOR's or OWNER's ID is to be used for the access check. (Normally the ORIGINATOR's ID will be specified). The ID specified by BEHALF is first checked against the ID of the file owner. If they are the same, then the access used is that granted to the owner. Otherwise, a search for the ID is made on the Access Control List. If the ID is specified on the Access Control List, then the access there defined is the one granted. If the ID is neither that of the owner nor on the Access Control List, then the access is that granted to the Universe.

Note that the above comments on the employment of the 'BEHALF' indicator apply only to those primitive requests which involve a tree search on the entity. Primitive requests which reference an entity by its reference number use the access specified in the associated Known Item Table entry.

A more detailed description of the manner in which the access at a particular level in a tree search is checked is summarized in "Access Checking in the Directory File" on the following pages.

SUMMARY OF USAGE ACCESS ATTRIBUTES

Bit	Attribute	Entity	Description
31	READ	File	Can read the file or spawn it into any base register, except zero
		Directory	Can read a directory to get information about any or all of the entries including access control lists
32	WRITE	File	Can truncate or rewrite existing contents of file without adding to its length
		Directory	Can delete or modify specifically named entry
33	APPEND	File	Can add to a file without changing its original contents
		Directory	Can add entries without changing existing entries
	NOTIFY	Event	Can issue a notify or delete to the event
34	EXECUTE	File	Can spawn the file into base register zero
		Directory	Can use the directory for a tree search on a symbolic name
	CAUSE	Event	Can issue a CAUSE or UNCAUSE to the event
35	LOCK	File	Can prevent other users from accessing the file
		Event	Can prevent other users from accessing the event

The usage attributes are referenced by the macro parameters ACCESS, UACCESS, and OACCESS. The access mask is loaded to the -A- or -Q- register (as specified by the primitive) with a bit set ON if the access attribute is to be granted.

0	31	32	33	34	35	
Not Presently Used	READ	WRITE	APEND	EXEC	LOCK	Mask for files and directories

0	33	34	35	
Not Presently Used	NOTIF	CAUSE	LOCK	Mask for events

Fig. IV-4

Access Checking in the Directory File

By access checking at a particular stage of a tree search we mean both a password check and, if that is successful, a check on the user's access permission at that stage. For the purpose of describing access checking we divide the primitives which involve a tree search into two categories.

Category I		OPEN	requested access
		OPEN WORKING DIR	execute access
		READ DIR	read access
Category II	A	CATLOG	append access
		CATDIR	append access
		CATLINK	append access
	B	DESTROY	write access
		READ BRANCH	read access
		READ LINK	read access
		READ ACL	read access
		WRITE ACL	write access
		WRITE SYSTEM INF	write access
		WRITE BRANCH	write access

If a primitive in category I has as an argument a tree name $N_1/N_2/\ldots N_k$ then the item actually being manipulated is the one whose local name is N_k. As a result access will only be checked at level k and only the password at level k need be supplied. The access permission which the user must have to that item is specified opposite the corresponding primitive.

Primitives of category II are manipulating the items whose local name is N_{k-1} and as a result access will only be checked at level k-1.

In establishing a link the link name may have the form $N_1/N_2/\ldots/N_k$ or $N_1/N_2/\ldots\ldots/N_k\$P_k$. If the user intends to manipulate the item named by the link name the latter form must be used. If, however, the link is to be used as simply a pointer to a directory and the items to be manipulated lie below that directory then the link name need not include P_k.

Access Checking in the Directory File (Continued)

The following conventions will be established with respect to error returns which result from a failure during the tree search.

Error Return	Circumstance
Invalid Name	Name not found in directory being searched and searcher has execute permission in that directory.
Access Denied	Access check on some item in a directory fails and searcher has execute permission in that directory.
No Search	All other cases

PRESENT PASSWORD SYSTEM

A password is always checked on the directory above the item being accessed.

AND on the root directory when the root directory is the item being accessed.

AND on the item being accessed when the command is an OPEN, OPWD or RDIR.

EXCEPT on catalogs in which case the password is checked only on the directory in which the item is to be cataloged.

Section V

Event Primitives Overview

EVENT PRIMITIVES OVERVIEW

Overview of Event Structure

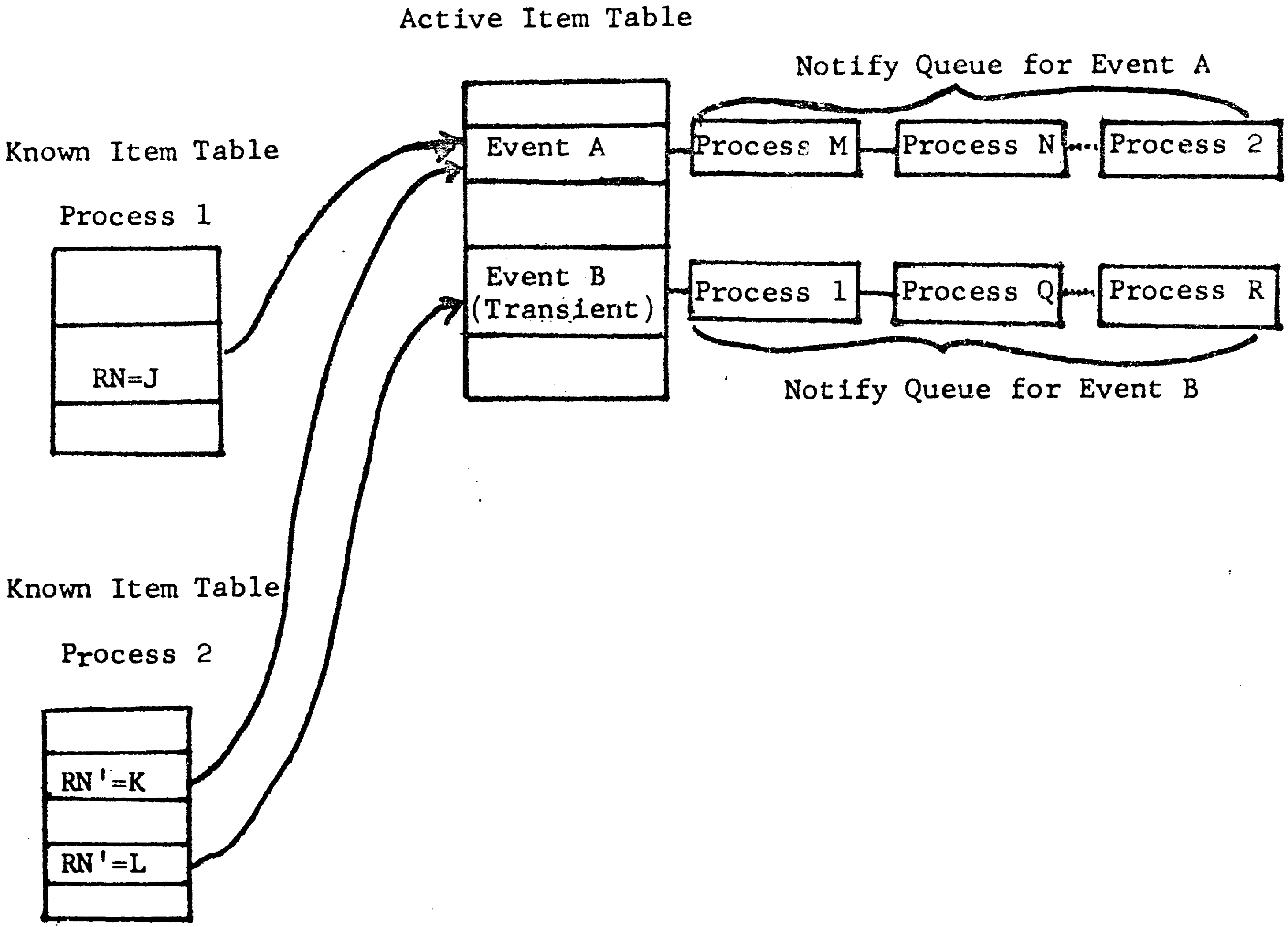
Events are used as a means of interprocess communication, and are handled much in the same manner as files. Events can be either scratch or catalogued (in which case they have a tree name). They have attributes (lock, notify, and cause). They have an owner, and they can be passed on in a spawn.

Events are referenced by an event reference number 'RN' which is either returned when a scratch or cataloged event is first opened, or which is passed to the process when it is spawned.

The system maintains a global event queue for each event which is currently active. A notify primitive to an event 'E' by a process 'P' results in an entry for 'P' being made on E's event queue. The corresponding cause primitive to event 'E' results in the trapping of the process which issued the notify to that event (See fig. V-1). The next time the process regains control, the trap routine (specified by the notify) is executed. If the process which issued the notify is blocked, the occurrence of the cause reawakens it.

Events are catalogued by means of the catlog primitive and a catalogued event is opened by the open primitive. These primitives are described in section IV on File Primitives, and are used for events in a manner similar to that indicated for files.

FIG. V-1



Summary of the action of the principle Event Primitives

OPEN SCRATCH EVENT - A global entry for the event is made in the Active Item Table (AIT) and a private entry for the process is made in its Known Item Table (KIT). For example, if Process 1 opens a scratch event, globally denoted as Event A, the corresponding reference number returned for its KIT entry is RN=J. When the scratch event is opened, its notify queue has length zero.

NOTIFY - An entry for the process issuing the notify is made on the event's notify queue in the AIT. For example, if Process 2 issues a notify with ERN'=K, an entry for Process 2 is made on the event queue for A.

CAUSE - A portion of the entries on the notify queue for the event caused will be trapped. For example, if Process 2 issues a CAUSE with ERN'=L, NUMBER=5, and STATE=3, then the first 5 entries on Event Queue B whose state equals 3 will be trapped.

Summary of Macro Calls for Event Primitives

The following calls are applicable for files and events

Code	Macro Call	
20	OPEN	TRAP,TREE_NAME,TREE_SIZE,BEHALF,ELSIZE,ACCESS
21	CLOSE	TRAP,RN
22	CATLOG	TRAP,RN,TREE_NAME,TREE_SIZE,SWITCH,UACCESS,OACCESS
23	DESTRO	TRAP,TREE_NAME,TREE_SIZE,BEHALF
36	LOCK	TRAP,RN
37	UNLOCK	TRAP,RN

The following calls apply only to events

Code	Macro Call	
38	NOTIF	TRAP,ERN,CTRAP,STATE
39	CAUSE	TRAP,ERN,STATE,MESSAGE,NUMBER
40	DELET	TRAP,ERN,STATE
41	UNCAU	TRAP,ERN,NUMBER
42	OPSCE	TRAP,TIMLIM,MODE,MAXLEN

Parameters Associated with the Event Primitives

MODE

Events have either a transient or steady state mode associated with them. The mode is specified when the event is initially opened by the Open Scratch Event primitive.

For a transient mode event, only those processes which issue a NOTIFY prior to the time that the event is CAUSEed can be trapped as a result of that CAUSE.

For a steady state event, processes which issue a NOTIFY either prior or subsequent to the time that the event is CAUSEed can be trapped as a result of the CAUSE (See also CURRENT COUNT).

CURRENT COUNT

For each event, the system maintains a running count of the number of processes yet to be trapped. This parameter, the CURRENT COUNT, is incremented by the parameter 'NUMBER' whenever a CAUSE to the event is issued. The current count is decremented by one each time a steady state event is CAUSEed, and is reset to zero each time a transient event is CAUSEed. The current count must be either zero or positive.

Note that for a steady state event, the current count may remain non-zero after the occurrence of a CAUSE to that event. This results when the number of entries on the event queue is less than the parameter 'NUMBER' specified by the CAUSE. A process which subsequently issues a notify to a steady state event with non-zero current count will be immediately trapped.

The current count is therefore the implementation mechanism whereby a NOTIFY to a steady state event can be acknowledged subsequent to the CAUSE to that event. The current count and the current length of an event's queue can be obtained by issuing the Request Status primitive with the event of concern specified by its reference number

NUMBER The number of entries on the particular event queue which are to be notified (i.e., trapped) when the CAUSE primitive is issued or which are to be deleted when the UNCAUSE primitive is issued. If 'NUMBER' is set to zero, all of the events on the event queue are notified (or deleted). Note, only those entries with the appropriate 'STATE' (see below) are included in those notified or deleted).

STATE A parameter specified by the NOTIFY and CAUSE primitives which provides a mechanism for distinguishing between processes awaiting notification. Only those processes whose event queue entries have states matching the state issued with the CAUSE will be trapped.

The state 777777777777_8 for transient mode events will match all states: that is, a CAUSE issued with this universal state will cause a process to be notified regardless of the state specified in the NOTIFY.

Note: States can only be defined for transient mode events. The 'STATE' must be set to zero for a steady state event. If an event primitive is issued to a steady state event with state other than zero, it will be rejected.

A fixed state event is an event which has been passed on a spawn or pass event and designated 'fixed state' by the passer. He must specify a state. When the receiver of the event performs a NOTIFY or CAUSE on it, the 'STATE' argument from the MME is ignored and the state designated by the father is used instead. If the receiver passes the event to someone else, it will remain fixed state.

MESSAGE When an event is caused it also transmits a one word message. The message is returned to the notified process in Status Return Word 2.

ACCESS A user may have NOTIFY, CAUSE, or LOCK access to an event. The access to a catalogued event for the owner and the universe is specified when the event is catalogued and checked when the event is opened. Access is granted when the indicated bit is set on.

NOTIFY	Bit 33
CAUSE	Bit 34
LOCK	Bit 35

Access for scratch events is established by the open scratch call.

MAXLEN — The maximum number of entries which can be placed on the queue for a particular event. This number is specified in the Open Scratch Event Primitive and may be from 1 to 16 as presently implemented. If the 'MAXLEN' parameter is set to zero, a default number of 16 is assigned.

TIMLIM — The maximum amount of time a process will be allowed to wait for an event for which it has issued a NOTIFY. This parameter is supplied in the Open Scratch Event Primitive and is expressed in seconds up to a maximum of 1/2 hour. If this parameter is set equal to zero, a default option of 1/2 hour is assumed.

TRAP — The trap address corresponding to the issuing of the event primitive. The logical status (Status Word 1) employs the same error code as the File Primitives and is summarized in Appendix A-2.

TRAPC — The trap address where control is returned to a process when an event is caused. (The process specifies this trap address in the Notify Primitive). A message may be returned in Status Word 2 of this trap.

Status Return Word 1 will contain zero if the trap is for a successful return (i.e., the CAUSE for this NOTIFY has been issued) and will contain an appropriate error code if the trap is the result of the time limit being exceeded.

RN — Reference number of a file or an event

ERN — Reference number of an event

SYSTEM EVENTS

The system events are events catalogued by the system which may be used by any slave process. At present there are three system events, the TIME, UNLOCK, and DEALLOCATE events. System events are catalogued with a root directory name of SYSEVNTS. The tree name of the system events are respectively SYSEVNTS/TIME, EVNTS/UNLOCK, and SYSEVNTS/DEALL. A description of these events and their use follows.

Time Event

A process can request notification (i.e. being trapped) at a time N/64 millisecond after it issues a NOTIFY to the time event, where N is specified as the state of the NOTIFY.

To use the time event:

1) OPEN the time event with the tree name specified below and request notify access.

   ACI 6,SYSEVNTS

   ACI 6,TIME

2) Issue a NOTIFY to the time event with the following parameters

TRAP	Trap associated with issuance of notify
ERN	Reference number of time event which was returned by the open.
TRAPC	Trap location to which the system is to return control when the system causes the time event.
STATE	The number of 64th of a millisecond, N, after the NOTIFY in which the process is to be trapped.

See the OPEN and NOTIFY primitives for additional details.

Unlock Event

A process can request notification when a particular item is unlocked. The item in question is identified by its reference number which is used as the state of the notify.

To use the unlock event:

1) OPEN the unlock event with the tree name specified below and request notify access.

   ACI 6,SYSEVNTS

   ACI 6,UNLOCK

Unlock Event (cont'd)

2) Issue a NOTIFY to the unlock event with the following parameters

TRAP	Trap associated with the issuance of the notify
ERN	Reference number of unlock event which was returned by the open
TRAPC	Trap location to which the system is to return control when the item referenced is unlocked.
STATE	The reference number of the item which is tested for being unlocked.

See the OPEN and NOTIFY primitives for additional details.

Deallocate Event

If a busy status is returned after an OPEN primitive is issued on a device the process can request notification of the device deallocation. When the process is trapped back with logical status 0, the device is then open to it and the status returned is specified below:

Status return word 1:

Bits 0 -17	reference # of open device
18-35	logical status code (See Appendix A-2)

Status return word 2:

Bits 0 -19	not used
20-25	access mask (right justified)
26	not used
27-35	device ID (see fig. III-1).

To use the deallocate event:

1) OPEN the deallocate event with the tree name specified below and request notify access.

```
ACI 6,SYSEVNTS

ACI 6,DEALL
```

Deallocate Event (cont'd)

2) Issue a NOTIFY to the deallocate event with the following parameters

NOTIFY	TRAP,ERN,TRAPC,STATE,MESSAGE
TRAP	Trap associated with the issuance of the notify
ERN	Reference # of deallocate event which is returned by the open
TRAPC	Trap location to which the system is to return control when the device referenced is deallocated.
STATE	The tree name of the device which is tested for being deallocated. ex. ACI 1,MTAA
MESSAGE	The message is to be loaded into the Q register. The upper half of Q must contain the element size, expressed in bits, in which subsequent data transactions involving this device are to be expressed. The lower half must have the access requested for the device: Bit 31 Read Bit 32 Write Bit 33 Append Bit 34 Execute Bit 35 Lock If Q lower is set to zero, the process will receive all the access to the device that is allowed.

Possible bad status on NOTIFY:

DEC	OCT	
2	2	Access denied
12	14	Illegal element size
31	25	Device not allocated - reissue OPEN

SPECIAL EVENTS

Process Event

The process event is a scratch event, associated with each process, which occurs (i.e. is CAUSEd) when the process is terminated. The reference number for this event is returned to the father in the Upper Half of Status Word 1 when the process is spawned. (The process event reference number is also referred to as the son's reference number).

The process event may be passed to other processes or closed by the father. In the latter case the son becomes unknown to the father. A CAUSE issued to this event results in the termination of the process. The event is a steady state mode event with NUMBER = ∞ . The father has full access to the event. There are six ways a process can terminate:

1) execution of a CAUSE on the process event
2) execution of a final CLOSE on the process event
3) execution of terminate primitive
4) commiting a fault, other than MME, while in default mode
5) using all the time allocated to process
6) process becomming to large to fit into core

Pass Event

A pass event is a transient mode event which enables one process (Process A) to pass the reference number of an item to another process (Process B). State differently, Process A can, by CAUSING the pass event, make an item known to Process B.

To create a pass event, bit 16 is set ON in the MODE parameter for the Open Scratch Event Primitive. The MODE parameter, which is loaded into index register -X4- will be interpreted as follows:

Bits 16,17	Interpretation of Mode Parameter
00	Steady state mode, regular event
01	Transient mode, regular event
11	Transient mode, event of type pass
10	Error. Pass events are defined only in transient mode.

CAUSE TRAP,ERN,STATE,MESSAGE,FRN,ACCESS

The parameters for the CAUSE of the pass event have the following meaning:

TRAP	Trap address associated with issuance of CAUSE
ERN	Reference number of the pass event being CAUSEed.

ERN	This reference number is obtained by opening either a scratch or catalogued event of type 'pass'.
STATE	State associated with the CAUSE primitive.
FRN	The reference number of the item being passed.
ACCESS	The access granted on the item being passed. This access is and'ed with the passer's access on the item. (See Fig. IV-4 for format).
MESSAGE	Only 18 bits can be sent as a message with a pass event. QU - Message QL - Ignored
NUMBER	The number of processes to receive the item passed.

The following discussion illustrates how the LISTENER might use the pass event to allow system processes to pass items to it:

1) The LISTENER opens a scratch pass event, and then catalogs it with universal access of CAUSE and owner's access of NOTIFY and CAUSE.

2) The LISTENER issues a NOTIFY to the pass event.

3) To pass an item to the LISTENER, a process must first open the catalogued pass event and then issue a CAUSE to this event.

The format of the status return words for a notify on a pass event is as follows:

TRAP	WORD 1	BITS 0 -17	Not specified
		BITS 18-35	Logical status code (See Appendix A-2)
TRAPC	WORD 1	BITS 0 -17	FRN of passed item
		BITS 18-35	Logical status code (See Appendix A-2)
	WORD 2	BITS 0 -17	Message
		BITS 18-35	ACODE of passer. Accounting ID.

The ACODE is a code set up when a process is created by the LISTENER (via the LOGIN sequence) and passed along to all descendents of that process. It is unique to each family of processes (generally one user.)

Section VI

Description of System Primitive Commands

DESCRIPTION OF SYSTEM PRIMITIVE COMMANDS

Introduction

In this section a description of each of the system primitive commands is given. Each description is given in the following general format.

FORMAT	EXPLANATION
Name of Primitive Command	Self Explanatory
Primitive Command Macro Call	Macro call for system macro which initiates the primitive action. The macro expansions are given in Appendix D.
Registers	Specifies the contents of the registers prior to the issuance of the MME for the primitive command.
Status Return Words	Specifies the information which the executive returns to the first two words at the primitive trap location when the primitive operation is completed.
Remarks	Self explanatory

Information On The Use of Primitive Command Descriptions

The primitive command descriptions should initially be used in conjunction with the overview summaries presented in the previous sections. After the basic terminology is understood the descriptions themselves should suffice. It is suggested that the programmer have a clear understanding of the system trap handling procedure (Section II) before attempting to utilize the primitives.

SUMMARY OF PRIMITIVE COMMANDS AND THEIR CODE NUMBERS

Primitives Executed Directly by the Master Mode Executive

Primitive Code and Name	Primitive Classification
00--Privileged Command From Slave Exec	CONTROL
01--Set Up Fault Vector	CONTROL
02--Set Up Squeze Mode	CONTROL
03--Enter Squeze Mode	CONTROL
04--Read	I/O
05--Append	I/O
06--Random Read	I/O
07--Random Write	I/O
08--Scratch File	I/O
09--Set Pointer	I/O
10--Request Status	I/O
11--Request Date and Time	CONTROL
12--Request Elapsed Run Time	CONTROL
45--System Status Measurements	CONTROL
46--Measure READ Me	CONTROL
48--Write Me	CONTROL
49--Who Am I	CONTROL
51--Request Working Directory	CONTROL

Primitives Executed by the Slave Mode Executive

13--Spawn	CONTROL
14--Terminate	CONTROL
15--Pause	CONTROL
16--Open Segment	CONTROL
17--Close Segment	CONTROL
18--Change Segment Length	CONTROL
19--Exchange Segments	CONTROL
20--Open	FILE AND EVENT
21--Close	FILE AND EVENT
22--Catalog	FILE AND EVENT
23--Destroy	FILE AND EVENT
24--Open Scratch	FILE
25--Update	FILE
26--Catalog Directory	FILE
27--Write Access Control List	FILE
28--Read Access Control List	FILE
29--Read Directory	FILE
30--Open Working Directory	FILE
31--Read Branch	FILE
32--Read Link	FILE
33--Write System Information	FILE
34--Catalog Link	FILE
35--Write Branch	FILE
36--Lock	FILE AND EVENT
37--Unlock	FILE AND EVENT
38--Notify	EVENT

Summary of Primitive Commands and Their Code Numbers (cont'd)

Primitives Executed by the Slave Mode Executive (cont'd)

39--Cause	EVENT
40--Delete Entry	EVENT
41--Uncause	EVENT
42--Open Scratch Event	EVENT
47--Create Segment	CONTROL

Privileged Primitive (Slave Executive Only) Code = 0

This primitive is a privileged command issued to the master mode executive by the slave executive only.

Only the slave executive is authorized to issue this primitive. If other processes attempt to use this primitive, the call will be rejected with a command fault.

There are four micro commands, specified by a micro command number, which are associated with this primitive.

Register	Description
X0	0 = Privileged Primitive code number
X1	Micro command number 00 -- Run micro command 01 -- Set BAR micro command 02 -- Destroy scratch file micro 03 -- Update micro 04 -- I/O clean-up routine micro 05 -- Deallocate Drum File micro

The micro commands are discussed on the following pages.

Run Micro Primitive (Slave Executive only)

The Run Micro Primitive (privileged) starts up the execution of the specified slave mode process. Only the slave mode executive is authorized to issue this primitive. This primitive sets up the Run-A-Program working storage and the indirect fault vector. Control is then transferred to the slave mode process via the R$RETRN routine.

This routine also sets up the BAR1 setting in the Run-A-Program storage for the slave executive to point to the state vector of the current running process. When the slave executive is re-entered for either a timer runout fault or primitive call, BAR1 will be set around the state vector of the current process.

Register	Description
X0	0 = Privileged Primitive
X1	00 = Run Micro Primitive
X2	AIT entry address for state vector
A	Process ID

Set BAR Micro Primitive (Slave Executive Only)

The Set BAR Micro Primitive sets a specified base register around a specified state vector, and returns directly.

This primitive is accessed by the privileged primitive command and is allowed for the slave mode exec only.

Register	Description
X0	0 = Privileged Primitive code number
X1	01 = Set BAR Primitive
X2	AFT pointer for segment
X3	BAR to be set

Destroy Scratch File Micro Primitive (Slave Executive Only)

This Micro Primitive (privileged) releases any disc space or page table space allocated to the file. The file length, allocation and the disc address fields in the AFT are cleared and the disc and page table bits in the flag word are also cleared. The AFT entry is not destroyed by this primitive - that is done by the Slave Exec.

Register	Description
X0	00 = Privileged primitive
X1	02 = Destroy Scratch File Micro Primitive
X2	Slave Trap address
X6	Slave Address of AFT of file

Update Micro Primitive (Slave Executive Only)

The Update Micro Primitive is used in the course of the non-privileged update primitive (25). If the file has a page table and it is in core and has been changed, the page table is written out to the page table file. In any case, at the close of the primitive the file will not have a page table in core. If the file has a new page table and the global switch G$G2 is set, then the bit map of the page table file is written out. If the file has new disc allocation and the global switch G$G1 is set, then the disc bit map is written out. In any case, G$G1 and G$G2 will be clear after the primitive has been completed. Some of the bits of the flag word are cleared but the AFT entry is otherwise untouched.

Register	Description
X0	00 = Privileged Primitive Code Number
X1	03 = Update Micro Primitive
X2	Slave Trap Address
X6	Slave Address of AFT of file

I/O Cleanup Micro Primitive

This routine is called by the I/O Cleanup Routine of the process terminator. It transfers control to the appropriate device cleanup routine.

Register	Description
X0	0 = Privileged Primitive code number
X1	04 = I/O Cleanup Micro Primitive
X2	Pointer to process table entry
X3	Slave address of outstanding operation entry
X4	Slave address of AIT entry

Deallocate Drum File Micro Primitive (Slave Executive Only)

The Deallocate Drum File primitive deallocates a logically contiguous part of a drum file beginning at the specific logical file address and continuing for as many units as specified. If the unit count is negative all of the file past the logical file address will be deallocated.

Register	Description
X0	00 = Privileged Primitive code number
X1	05 = Deallocate Drum File Micro Primitive
X2	Slave trap address
X6	Slave address of AFT of file

Setup Fault Vector Primitive   Code = 1

ˉETFV TRAP,CORELOC

The Setup Fault Vector Primitive is used by a slave process to declare the location of it's slave fault vector. If the declared location, 'CORELOC', is out of bounds then the process is trapped at 'TRAP' with an error return. Otherwise, the new slave fault vector location is established in the state vector and the indirect fault vector is setup to point to the new fault vector location.

Register	Parameter	Description
X0	(Implicit)	1 = Setup Fault Vector Primitive
X1	TRAP	Trap location
X2	CORELOC	Location of slave fault vector in slave program

STATUS RETURN WORDS

Word 1

18 - 35	Logical Status Code
	0 successful operation
	1 location of slave fault vector out of bounds
	2 upper portion of slave fault vector out of bounds

Format for the Slave Process Fault Vector

The slave process fault vector has a word pair for each of the 16 faults. The order of the faults are the same as in the hardware fault vector.

The format for a word pair is:

Word 1	Storage for the IC and indicators which are in effect when the fault occurs.
Word 2	First instruction of the slave process fault handling routine.

Default mode

If the slave process runs in default mode (no slave fault vector declared), the process is aborted if a fault occurs, except for a MME fault or Timer Runout.

Example of Coding for Fault Vector

```
FLOC  NULL

      DEC   0
      TRA   SDFFT          0 = Shutdown Fault
      DEC   0
      TRA   MEMFT          1 = Memory Fault
      DEC   0
      TRA   MME            2 = Master Mode Entry
      DEC   0
      TRA   FTGFT          3 = Fault Tag Fault
      DEC   0
      TRA   TROFT          4 = Timer Runout Fault
      DEC   0
      TRA   CMDFT          5 = Command Fault
      DEC   0
      TRA   DRLFT          6 = Derail Fault
      DEC   0
      TRA   LUPFT          7 = Lockup Fault
      DEC   0
      TRA   CONFT          8 = Connect Fault
      DEC   0
      TRA   PARFT          9 = Parity Fault
      DEC   0
      TRA   IOCFT         10 = Illegal Op Code Fault
      DEC   0
      TRA   ONCFT         11 = Operation not Complete Fault
      DEC   0
      TRA   SUPFT         12 = Startup Fault
      DEC   0
      TRA   OVFFT         13 = Overflow Fault
      DEC   0
      TRA   DCKFT         14 = Divide Check Fault
      DEC   0
      TRA   EXEFT         15 = Execute Fault
```

Note on the Indirect Fault Vector

The Indirect Fault Vector is a block of code maintained by the system executive for passing control from the hardware fault vector to the appropriate entry in the slave process fault vector.

The indirect fault vector is maintained in the Run-A-Program working area and is modified to point to the slave process fault vector when the Set Fault Vector Primitive is issued.

The indirect fault vector contains a word pair for each fault. The first word is a pointer to a corresponding word in the slave process fault vector. The second word is a -TSS- to the corresponding second word in the slave fault vector.

The indirect fault vector, therefore, points to the location in the slave fault vector where the IC and indicators are to be stored by the master mode executive. It also enables control to be returned in slave mode to the user program.

Example:

The system macro which calls the Set Fault Vector Primitive is

```
SETFV   MACRO
        LDXO    1,DU      Set Fault Vector Code Number
        LDX1    #1        TRAP
        LDX2    #2        CORELOC
        MME     0         Issue the Primitive
        ENDM
```

The macro call

```
   SETFV (TRAP,DU),(FLOC,DU)
```

establishes a fault vector at FLOC (see previous page for typical coding at FLOC). The trap routine is located at TRAP.

Setup Squeeze Mode Primitive   Code = 2

SETSQ TRAP,LOCTBL

The Setup Squeze Mode Primitive initializes the Squeze Mode Table in the state vector of the process and computes the effective base register settings for the process when it enters squeze mode. 'LOCTBL' is the location of the squeze mode mapping table. If the parameters are in range, a successful trap to location 'TRAP' is made. Otherwise, an unsuccessful trap is returned. The process remains in normal mode until the squeze primitive is issued.

Register	Parameter	Description
X0	(Implicit)	2 = Set Squeze code number
X1	TRAP	Trap location
X2	LOCTBL	Location of Squeze Mapping Table

STATUS RETURN WORDS

Word 1		
18 - 35	Logical status code	
	0	successful operation
	1	squeze table out of bounds
	1	error in segment specification for squeze mode
Word 2		
0 - 17	Number of segment in error (if this was cause of error)	

The Squeze Mapping Table has the following form

```
        ZERO      0,MAP0
        ZERO      0,MAP1
        ZERO      0,MAP2
        ZERO      0,MAP3

        VFD       18/ORG0,1/WP,1/0,16/LEN0
        VFD       18/ORG1,1/WP,1/0,16/LEN1
        VFD       18/ORG2,1/WP,1/0,16/LEN2
        VFD       18/ORG3,1/WP,1/0,16/LEN3
```

where:

MAPI = Base register mapping for segment I. (Note that in the squeze mode table, the index in the table gives the base register in normal mode and MAPI the base register that it maps into in squeze mode).

Setup Squeze Mode Primitive (Continued)

ORGI = Origin in Segment I for squezed bar (relative to segment starting address)

LENI = Length (in words) or the bar setting for Segment I in squezed mode.

WP = Write protect bit (1 = no writes allowed)

Example of initial settings for squeze mapping table:

```
DEC    2                  Seg 0 normal mode maps to seg 2 squeze mode
DEC    3                  Seg 1 normal mode maps to seg 3 squeze mode
DEC    0                  Seg 2 normal mode maps to seg 0 squeze mode
DEC    1                  Seg 3 normal mode maps to seg 1 squeze mode
```

The remainder of the table sets up the origin and length of each segment in squeze mode.

```
OCT    000000400000    SEG 0, ORG = 0, LEN = 0, WP
OCT    000000400000    SEG 1, ORG = 0, LEN = 0, WP
OCT    000000000000    SEG 2, ORG = 0, LEN = 0, No WP
OCT    000000000000    SEG 3, ORG = 0, LEN = 0, No WP
```

Enter Squeeze Mode Primitive Code = 3

SQUEZ TRAP,REGS,IC

The Enter Squeze Mode Primitive (also called SQUEZE) will reset the base address registers to those previously established by the SETSQ Primitive, load the registers from 'REGS', and transfer to 'IC'. ('IC' contains the IC and indicators to be used on entry to the squezed program).

The process will trap at location 'TRAP' on completion. If squeze mode is not set up, then the process is trapped with error return of four (4). If the register load area is out of bounds, then the process will be trapped with error return of one (1). If the transfer address is out of bounds, then a memory fault will occur.

Register	Parameter	Description
X0	(Implicit)	3 = Enter Squeze Mode Primitive code number
X1	TRAP	Trap location
X2	REGS	Location of register settings
A	IC	Instruction counter and indicators

STATUS RETURN WORDS

Word 1

18 - 35		Logical status code
	0	successful operation
	4	squeze mode not set up
	1	register load area out of bounds

Trap Handling for Squeze Mode Programs

Consider the sequence of events that occurs when a process running in squeze mode is interrupted. If no traps have returned during the time of interrupt, control is returned to the slave process which continues its execution in squeze mode. If, however, a trap has returned (the trap must be for the unsquezed process) a squeze mode bit (bit 35) is set in the indicators which are stored in the trap exit location. In addition, the process mode is changed from squeze to normal and the base address registers set to their unsquezed values. The outstanding traps are linked in the manner described in section II and control is returned to a trap routine in the unsquezed program. The unsquezed program utilizes the squeze bit, which will be set in the exit location of the last trap routine in the linked sequence of trap routines, to identify the return address in the squezed program. The standard return from a trap routine, RET to the exit location, cannot be used in this case. Instead, a SQUEZE to the return address must be issued.

Read Primitive

Code = 4

READ TRAP,FRN,CORELOC,N,MODE

The Read Primitive (Sequential Read) transfers the next 'N' elements from the file 'FRN' to the area in the slave program which starts at 'CORELOC'. Data is transferred to core from the file element located by the Current Read Pointer.

Register	Parameter	Description
X0	(Implicit)	4 = Read Primitive code number
X1	TRAP	Trap location
X2	FRN	File reference number
X4	CORELOC	Core memory location indicating the first word in a user slave area for a data transmission.
X5	N	Number of elements in the referenced file which are to be transmitted. The actual element size is specified when the file is opened. (See I/O Primitive Overview for definition of element).
X6	MODE	An indicator specifying what mode the transmission is to be made in. This parameter varies from device to device and is summarized in Appendix B.

STATUS RETURN WORDS

Word 1

0 - 17	The number of units transferred in a successful I/O operation
18 - 35	Logical Status Code for I/O Primitives (See Appendix A-1).

Word 2

0 - 35	Device dependent physical status. See appropriate device manual for interpretation.

The Updating and Determination of the Current Read Pointer For Disk or Drum Files

The starting location for the (Sequential) Read Primitive is specified by a current read pointer which is initialized to zero whenever a file is opened. If a file is shared by several users, each user has his own read pointer.

The current read pointer is updated each time a successful sequential read is executed. However, the pointer may or may not be updated if the read is in error. (To determine the read pointer setting in this case, issue the Set Pointer Primitive for the file in question with the number parameter set to zero. The read pointer setting will be in the lower half of Status Word 2).

If the number of elements read exceeds the current file length, a successful return will be given with the read pointer being set at the end of the file. (That is, the read pointer is not extended past the end of the file, although the read itself may so extend.) This condition can be detected by comparing the number of units specified for transfer with the number of units actually transferred (returned in the upper half of Status Word 1).

The issuing of a read when the current pointer is set to the end of the file will result in an -end of file- error return.

Note that the read pointer is not modified when a random read is issued. For disc and drum files the read pointer can be shifted by the set pointer primitive. For magnetic tape files the pointer can be shifted by issuing the read with an appropriate mode. (See Appendix B for usage of mode parameter with magnetic tapes).

Append Primitive Code = 5

APPEND TRAP,FRN,CORELOC,N,MODE

The Append Primitive (Sequential Write) writes 'N' elements from core memory starting at 'CORELOC' into the file 'FRN'. The element location in the file at which writing starts is specified by the end of file pointer.

Register	Parameter	Description
X0	(Implicit)	5 = Append Primitive code number
X1	TRAP	Trap location
X2	FRN	File reference number
X4	CORELOC	Core memory location indicating the first word in a user slave area for a data transmission
X5	N	Number of elements in the referenced file which are to be transmitted. The actual element size is specified when the file is opened. (See I/O Primitive Overview for definition of element).
X6	MODE	An indicator specifying what mode the transmission is to be made in. This parameter varies from device to device and is summarized in Appendix B.

STATUS RETURN WORDS

Word 1

0 - 17	The number of units transferred in a successful I/O operation
18 - 35	Logical Status Code for I/O Primitives (See Appendix A-1).

Word 2

0 - 35	Device dependent physical status. See appropriate device manual for interpretation.

End of File Pointer (Current lenght of file)

The -End of File Pointer- locates the starting element in a file for the next Append (Sequential Write). When a scratch file is first opened the End of File Pointer locates the beginning of the file; that is, element zero.

When a catalogued file is opened the End of File Pointer locates the element following the last one which had been written into the file. There is only one End of File Pointer for a given file, so that Appends by several users sharing a file always start at the location determined by the current setting of the End of File Pointer.

The End of File Pointer is updated whenever a successful Append is executed and also by successful random writes if the last element of the random write exceeds the current pointer setting.

The End of File Pointer may also be updated by an unsuccessful Append if a hardware error occurs. The current position of the End of File Pointer may be obtained by issuing the Request Status Primitive, the position in elements being returned in the status word. The End of File Pointer may be reset to zero by issuing the Scratch Primitive.

Read Random File Primitive Code = 6

RRF TRAP,FRN,FILELOC,CORELOC,N

The RRF (Read Random File) Primitive transfers the 'N' elements from location 'FILELOC' in file 'FRN' to the area in core memory which starts at 'CORELOC'. This primitive can only be used with drum or disk files.

Register	Parameter	Description
X0	(Implicit)	6 = Read Random File Primitive code number
X1	TRAP	Trap location
X2	FRN	File reference number
X3	FILELOC	An integer, representing the element number within the file at which a data transmission begins. This parameter is only employed for random operations.
X4	CORELOC	Logical core memory location indicating the first word in a user slave area for a data transmission.
X5	N	Number of elements in the referenced file which are to be transmitted. The actual element size is specified when the file is opened. (See I/O Primitive Overview for definition of element).

STATUS RETURN WORDS

Word 1

0 - 17	The number of units transferred in a successful I/O operation
18 - 35	Logical Status Code for I/O Primitives (See Appendix A-1).

Word 2

0 - 35	Device dependent physical status. See appropriate device manual for interpretation.

Write Random File Primitive Code = 7

WRF TRAP,FRN,FILELOC,CORELOC,N

The WRF (Write Random File) Primitive writes 'N' elements from core memory starting at 'CORELOC' into the file 'FRN'. The element in the file at which writing starts is specified by 'FILELOC'. This primitive can only be used with drum or disk files.

Register	Parameter	Description
X0	(Implicit)	7 = Write Random File Primitive code number
X1	TRAP	Trap location
X2	FRN	File reference number
X3	FILELOC	An integer, representing the element number within the file at which a data transmission begins. This parameter is only employed for random operations.
X4	CORELOC	Logical core memory location indicating the first word in a user slave area for a data transmission.
X5	N	Number of elements in the referenced file which are to be transmitted. The actual element size is specified when the file is opened. (See I/O Primitive Overview for defintion of element).

STATUS RETURN WORDS

Word 1

0 - 17	The number of units transferred in a successful I/O operation.
18 - 35	Logical Status Code for I/O Primitives (See Appendix A-1).

Word 2

0 - 35	Device dependent physical status. See appropriate device manual for interpretation.

Scratch Primitive Code = 8

SCR TRAP,FRN,FILELOC

The Scratch Primitive destroys all data in the file 'FRN' by releasing all storage owned by the file and setting the current Read and End of File Pointers to the start of the file if FILELOC = 0. The file is not closed by this operation. Otherwise, the file is scratched starting from the element specified by FILELOC.

Register	Parameter	Description
X0	(Implicit)	8 = Scratch Primitive code number
X1	TRAP	Trap location
X2	FRN	File reference number
X3	FILELOC	Element in file to scratch from

STATUS RETURN WORDS

Word 1	
0 - 17	Not specified
18 - 35	Logical Status Code for I/O Primitives (See Appendix A-1).
Word 2	
0 - 35	Device dependent physical status. See appropriate device manual for interpretation.

REMARKS

This primitive can be used only on disk or drum files.

Set Pointer Primitive   Code = 9

SPTR TRAP,FRN,N

The Set Pointer Primitive shifts the current Read Pointer by 'N' element for file 'FRN' which is located on the disc or drum. (For files corresponding to magnetic tapes, positioning is accomplished by using the Read Primitive with an appropriate mode).

The process is trapped at location 'TRAP' upon completion with the new setting of the Read Pointer, expressed in elements, in the lower half of Status Word 2. Error returns will be made if the process does not have read access for the file or if the pointer is shifted beyond the maximum file length or file beginning.

Register	Parameter	Description
X0	(Implicit)	9 = Set Pointer Primitive code number
X1	TRAP	Trap location
X2	FRN	File reference number
X3	N	The number of elements which are added to or subtracted from the current setting of the read pointer. (For subtraction the number is expressed in two's complement form).

STATUS RETURN WORDS

Word 1

0 - 17	Not specified
18 - 35	Logical Status Code for I/O Primitives (See Appendix A-1).

Word 2

0 - 17	Not specified
18 - 35	New setting of Read Pointer (in elements)

Request Status Primitive Code = 10

RQST TRAP,RN

The Request Status Primitive returns information relating to file (or event) 'RN' in the Status Return Words. The Process traps at location trap upon completion with the status words containing the information summarized below.

Register	Parameter	Description
X0	(Implicit)	10 = Request Status Primitive code number
X1	TRAP	Trap location
X2	RN	File or event reference number

STATUS RETURN WORDS

Word 1	
0 - 17	Number of bits/element (files) Current count (events)
18 - 35	Logical status code for I/O primitives (See Appendix A-1).
Word 2	
0 - 17	File length in units (e.g. 32 words for disk) (files) Queue length (events)
18 - 20	Not Used
21 - 25	Access mask (files) (RWAEL)
23 - 25	Access mask (events) (NCL)
26	Not Used
27 - 35	Device ID

Request Date and Time Primitive   Code = 11

RQDT   No arguments

The Date and Time Request Primitive returns the date in the A-register, the time of day in the Q-register, and returns control to the location of the MME plus one.

Register	Parameter	Description
X0	(Implicit)	11 = Request Date and Time Primitive code number

STATUS RETURN WORDS

None since no trap routine.

Format for Date and Time

DATE   Expressed as BCD characters (i.e., 0007010606108 for 7/16/68).

TIME   33 bit number representing time since midnight in 64's of a millisecond. (Note: this can exceed 24 hours).

Request Elapsed Run Time Primitive Code = 12

RQERT No arguments

The Request Elapsed Run Time Primitive returns the elapsed run time of the process, right justified in the Q-reg. The units are 1/64 of milliseconds. It returns the count of resources used by the process in the A-reg. This primitive has no trap routine. Upon completion control is returned to the location of the MME plus one.

Register	Parameter	Description
X0	(Implicit)	12 = Request Elapsed Time Primitive Code Number

STATUS RETURN WORDS

None (since no trap routine)

Other Return Information

A	Resource count in resource units
Q	Elapsed run time (1/64 milliseconds), right justified.

Spawn Primitive Code = 13

SPAWN TRAP,PLOC,LENGTH,ORIG

The Spawn Primitive creates a new process (a son) which will be executed in parallel with its creator (the father). The father specifies the information pertaining to the son in a parameter list starting at 'PLOC'. The number of words in the list is given by 'LENGTH'. When the son has been created, the father is trapped at 'TRAP' with the son's reference number in the upper half of Status Return Word 0.

Register	Parameter	Description
X0	(Implicit)	13 = Spawn primitive code number
X1	TRAP	Trap location
X4	PLOC	Location of parameter list
X5	LENGTH	Length of parameter list
Q	ORIG	Originator (for special SPAWN)

STATUS RETURN WORDS

Word 0

0-17	Event reference number for process event just spawned (i.e., son's reference number).
18-35	Logical status code

0 successful operation
12 argument list must be at least 14 words long
4 segment zero (0) must be specified
5 illegal segment specification
6 invalid access on segment specification
7 illegal file reference number
8 only a file or event may be passed
9 resources not available

REMARKS

The number of files and events passed to the spawned process will appear in X-register zero (0) on initial entry to the process.

If a user is trapped with status code = 9, he should retry the command.

The son's reference number is also referred to as the process event number (See 'Special Events' in Section V).

Format of Parameter List

Words	Function	
0-7	Initial register settings for son	
8	Time limit for process in 64's of a msec. A negative number gives an infinite time limit	
9	Option switches	
	Bits 14-17	Priority curve (0=default, same priority as father)
	Bit 32	Indicates core end segments exist
	Bit 33	On if process is to know about itself, as reference number one greater than the number handed on by the father, (N+1).
	Bit 35	If on, process will be given a timer runout fault. If it runs out of time, it will get 16 secs. more.
10	Son segment zero (0) specification	
11	Son segment one (1) specification	
12	Son segment two (2) specification	
13	Son segment three (3) specification	
14-N	Files and events to be passed to the son (i.e., the specification of the son's KIT).	

Segment Specification

One (1) word per segment

Bits 0-17	File Reference or Segment Number
Bit 18	Write Protect
Bit 32	Core end segment
Bits 34-35	=0 for void segment =1 for spawn from parent's segment =2 for spawn from parent's file

KFT Specification for File or Event

One (1) word per file or event or 2 words for fixed state

Bits 0-17	File or event reference number in father's KIT
Bit 20	Fixed state event - if on next word contains state
Bits 31-35	Access granted to son for file (Read, Write, Append, Execute, Lock)
or	
Bits 33-35	Access granted to son for event (Notify, Cause, Lock)

KFT Specification for File or Event (Continued)

The order of the word in the KIT specification will determine the item reference number seen by the son. (i.e., the item specified by Word 14 will be reference number 0 in the son's KIT, that specified by word 15 will be reference number 1, etc.) The access mask will be ANDed with the father's access mask to determine the son's access to the file.

In the case of a special spawn by a privileged process, Q will contain the originator's ID returned by the open working directory primitive. Six words, originator's name, are stored after the buffer, but they are not counted in the length.

Additional Information on SPAWN

1) All RN's which are specified in the parameter list are with respect to the father process.

2) The parameter list must be at least 14 words to specify the initial register settings and the contents of all four segments. The remaining words which specify the son's files and events are optional.

3) The son's process starts execution at location zero of segment zero. Hence, segment zero must be specified and correspond to an executable procedure.

4) Segments other than segment zero may be void. Void segments are specified by setting bits 34-35 zero in the corresponding segment specification word.

5) The father must have execute access on a file which he spawns as segment zero for his son and read access for files which are spawned as segments one, two, or three.

6) The father can make a segment write inhibit by setting the write protect bit in the corresponding segment parameter word. If the segment is spawned from a write-inhibit file or from a father's segment which is write inhibit, the bit will be set on regardless.

7) The access mask for a son's file will be -ANDED- with the father's access to determine the son's resultant access to the file. Hence, the son is not allowed an access to a file that the father does not have.

8) If the file that is spawned in segment zero has the owner's access privilege ON then the son will be allowed to use the owner's ID associated with that file in performing file operations. Note that the owner's access switch is set when the file is catalogued.

Terminate Primitive

Code = 14

TERM   no arguments

The Terminate Primitive terminates the execution of a process and liquidates the resources used by the process. The following things happen:

Event trap blocks are returned to AIT free list
Segments are closed
State vector is closed
Process event is activated

Register	Parameter	Description
XO	(Implicit)	14 = Terminate Primitive identifier

Pause Primitive

Code = 15

PAUSE No arguments

The Pause Primitive blocks the further execution of the current process if:

1) There are no traps on the outstanding trap queue for the process, and

2) If there is at least one trap for an I/O, file, or event primitive which has not returned.

If these conditions are satisfied, the process will remain blocked until one of the I/O, file, or event traps return. At this time, the process resumes execution at the trap location.

If neither condition is satisfied, the corresponding Pause results in the simulation of a timer runout fault. That is, the process is rescheduled. In such a case, the process will resume execution at the instruction following the Pause MME the next time the scheduler runs the process.

Register	Parameter	Description
X0	(Implicit)	15 = Pause Primitive code number

Notes on the Use of 'PAUSE'

When a process no longer has useful work to do (i.e., it is awaiting the completion of some operation by the executive) it should issue the 'PAUSE' Primitive instead of sitting in a wait loop. This places the process on a queue of blocked processes and enables another process to utilize the processor. The process which issued the 'PAUSE' will be rescheduled when the trap which is being awaited is returned.

The following code illustrates how the Pause Primitive can be incorporated in a 'WAIT' macro, which would be issued whenever a process has no useful work to perform.

Pause Primitive (Continued)

```
WAIT    MACRO
        REM
        INHIB    SAVE,ON        Prevent interrupts in middle of test loo
        SZN      TRAP+2         Test exit word to see if trap has return
        TNZ      *+4            Trap has returned-Exit and continue pro-
                                cessing
        LDXO     15,DU          Otherwise issue PAUSE primitive
        MME      0
        TRA      *-4            Go back if control reaches here
        INHIB    RESTORE
        ENDM
        STZ      TRAP+2         Set exit word to zero as a flag
        OPEN     (   ),...,     Issue Open primitive
        WAIT                    Pause and wait for trap to return
        TRA      RESUME         Continue processing
```

A Trap routine, located at 'TRAP' which can be used with the above macro is:

```
TRAP    NULL
        DEC      0              Storage for Status Word 1
        DEC      0              Storage for Status Word 2
        DEC      0              Storage for trap routine exit location
        RET      TRAP+2         Return to process via standard trap retu
```

Further Notes on the Use of PAUSE

It is important to note that the Pause Primitive will cause a process to go in a blocked state only for a subset of the I/O, File, and Event primitives.

For the remaining primitives, which include all of the control primitives, the PAUSE results in the process being rescheduled. If a user desires to go blocked in this case, he should precede the PAUSE by a Notify Event primitive.

Note that when a process is blocked it becomes active again after the occurrence of the I/O or event trap. Control returns to the process at the trap routine and continues in a manner determined by the trap routine coding. If each trap routine is terminated by a -RET WORD3-, control eventually transfers to the instruction following the PAUSE MME. Otherwise, control transfers to the indicated termination loca-
ion.

Open Segment Primitive Code = 16

OPSEG TRAP,SEGNUM,LENGTH

This primitive opens a segment for use by the process. The process specifies the segment number and length desired.

Register	Parameter	Description
X0	(Implicit)	16 = Open Segment Primitive code number
X1	TRAP	Trap location
X2	SEGNUM	Segment number (0, 1, 2, or 3)
X3	LENGTH	Desired segment length, in words. The segment length must be less than 200000 octal or 65536 decimal.

STATUS RETURN WORDS

Word 1

18 - 35	Logical status code
	0 successful operation
	1 segment number out of range
	7 segment length out of range
	8 segment already opened

The logical addresses associated with the starting location of each c the four segments are

Segment 0	Octal 0
Segment 1	Octal 200000
Segment 2	Octal 400000
Segment 3	Octal 600000

Example:

OPSEG (TRAP,DU),(3,DU),(2048,DU)

Since SEGNUM = 3, the logical starting address for the segment is 600000 (octal).

The segment opened contains 2048 (Mod 512) blocks of 512 words.

Close Segment Primitive Code = 17

CLSEG TRAP,SEGNUM

This primitive routine closes a segment for a process. The segment is no longer available to the process, however, it may be available to other processes.

Register	Parameter	Description
X0	(Implicit)	17 = Close Segment Primitive code number
X1	TRAP	Trap location
X2	SEGNUM	Segment number (0, 1, 2, or 3)

STATUS RETURN WORDS

Word 1

18 - 35	Logical status code	
	0	successful operation
	1	segment number out of range
	2	segment not known
	4	I/O activity in progress

Change Segment Length Primitive Code = 18

CHSEG TRAP,SEGNUM,LENGTH

The Change Segment Length Primitive changes the length of the segment specified by 'SEGNUM' to 'LENGTH'.

Register	Parameter	Description
X0	(Implicit)	18 = Change Segment Length Primitive code numb
X1	TRAP	Trap location
X2	SEGNUM	Segment number (0, 1, 2, or 3)
X3	LENGTH	Desired length (words)

STATUS RETURN WORDS

Word 1

18 - 35	Logical status code
	0 successful operation
	1 segment number out of range
	2 segment not known
	7 segment length out of range
	4 I/O activity in progress
	5 segment is write inhibit
	6 requested segment length equals zero

Remarks

Segment lengths are defined in blocks of 512 words. The executive will round off the requested length to the next 512 word block, if this length is not a multiple of 512.

Exchange Two Segments Primitive   Code = 19

EXSEG TRAP,SEG1NUM,SEG2NUM

This primitive exchanges the segment numbers of two specified segments

Register	Parameter	Description
X0	(Implicit)	19 = Exchange Segments Primitive identifier
X1	TRAP	Trap location
X2	SEG1NUM	Segment 1 number (0, 1, 2, or 3)
X3	SEG2NUM	Segment 2 number (0, 1, 2, or 3)

STATUS RETURN WORDS

Word 1

18 - 35	Logical status code 0 successful operation 1 number for segment 1 is out of range 1 number for segment 2 is out of range

Open Primitive Code = 20

OPEN TRAP, TREE_NAME,TREE_SIZE,BEHALF,ELSIZE,ACCESS

The OPEN primitive makes the catalogued item specified by 'TREE_NAME' active and known to the process. The reference number (RN) and other information relevant to the item is returned in the status words. Most subsequent references to the item will be in terms of its reference number.

Register	Parameter	Description
X0	(Implicit)	20 = Open Primitive code number
X1	TRAP	Trap location
X4	TREE_NAME	Location of the tree name of the item being opened
X5	TREE_SIZE	Number of words in the tree name (a multipl of six)
X6	BEHALF	Behalf indicator - Bit 17 1 = Originator (normal setting) 0 = Owner
X7	ELSIZE	The element size, expressed in bits, in which subsequent data transactions involvin this file are to be expressed (not required for events).
Q	ACCESS	Requested access for item Bit 31 - Read (File) Bit 32 - Write (File) Bit 33 - Append (File) Notify (Event) Bit 34 - Execute (File) Cause (Event) Bit 35 - Lock (File,Event) If this word is set to zero, the program will receive all the accesses to the item that it is allowed. The accesses granted are returned in Status Word 2.

Open Primitive (Continued)

STATUS RETURN WORDS

Word 1

0-17	File or Event reference number
18-35	Logical Status Code (See Appendix A-2)

Word 2

0-17	File length in elements (File) Number of queue entries (event)
20-25	Access mask (right justified)
26	Not used
27-35	Device ID (See Fig. III-1).

Tree Name Conventions for Opening Device Files

The tree name of a device file contains two levels. These levels correspond to the identifier 'DEVICE', and the name code of the device type and number of the device. (If the identifier 'AA' is used, the next available free device will be supplied). The two words assigned to the password at each level are filled with ASCII blanks.

Device Name Code	Device Type
DI	Disc
DR	Drum
OP	Operator's Console
CP	Card Punch
LP	Line Printer
CR	Card Reader (see "Communication With Card Reader From the Listener")
TT	Teletypewriter
CL	Communication Line Multiplexer Channel
MT	Magnetic Tape

Example of treename coding for any magnetic tape (tree name = DEVICE/MTAA)

```
ACI 6,DEVICE          Device Identifier
ACI 6,MTAA            Code Name for any Card Reader
```

Example of treename coding for tape number 5 (tree name = DEVICE/MT05)

```
ACI 6,DEVICE          Device Identifier
ACI 6,MT05            Code Name for Tape Handler 5
```

REMARKS

1) Each time the OPEN primitive is issued, a new reference number for the specified file is returned. Hence, a given process may open the same file more than once.

2) Several processes may open the same file during a given time interval. The reference number returned by the OPEN primitive is private to to the process which issued the OPEN; that is, the reference number is an entry in the Known Item Table for the process. Global information about the file is maintained in a single Active Item Table (AIT) entry to which all of the KIT entries point.

3) Each device, except the operator's console, can only be opened by one process at a time. Subsequent processes trying to open it before the first process has closed will receive logical error status 3 (software busy). They can use the deallocate event to wait for its release.

Close Primitive

Code = 21

CLOSE TRAP,RN

The item whose reference number is 'RN' is made unknown to the process. When the item is closed, its attachment count is decremented by one. The global entry for the item (in the Active Item Table) is not removed until the attachment count reaches zero.

Register	Parameter	Description
X0	(Implicit)	21 = Close Primitive code number
X1	TRAP	Trap address
X2	RN	File or event reference number

STATUS RETURN WORDS

Word 1

0-17	Undefined
18-35	Logical Status Code (See Appendix A-2)

REMARKS

If a scratch file is closed with attachment count zero, the storage allocated to the scratch file is returned to the free storage pool.

Catalog Primitive Code = 22

CATLOG TRAP,RN,TREE_NAME,TREE_SIZE,SWITCH,UACCESS,OACCESS

The scratch item whose reference number is 'RN' is catalogued with the tree name specified by 'TREE_NAME'. A segment spawned from a catalogued file will be write inhibit if the write inhibit indicator in 'SWITCH' is set on.

Register	Parameter	Description
X0	(Implicit)	22 = Catalog Primitive code number
X1	TRAP	Trap location
X2	RN	Reference number of the file or event being catalogued
X4	TREE_NAME	Location of the tree name of the catalogued ite
X5	TREE_SIZE	Number of words in tree name (a multiple of six
X6	SWITCH	Bit 17 - Behalf Indicator 1 = Originator's behalf (normal setting) 0 = Owner's behalf Bit 16 - Owner's Access Privilege Indicator (Files Only). See Remarks. 0 = Access to owner's ID not granted (normal setting) 1 = Access granted to owner's ID Bit 15 - Write Inhibit Indicator (Files Only) 0 = Segment spawned from file will not be write inhibit 1 = Segment spawned from file will be write inhibit (See Remarks)
A	UACCESS	Universal Access
Q	OACCESS	Owner's Access Bit 31 - Read (File) Bit 32 - Write (File) Bit 33 - Append (File) Notify (Event) Bit 34 - Execute (File) Cause (Event) Bit 35 - Lock (File,Event)

STATUS RETURN WORDS

Word 1

0-17	Undefined
18-35	Logical Status Code (See Appendix A-2)

REMARKS

1) If the owner's access privilege switch is set on, any user with execute access on this file will be allowed the use of the file owner's id when the user spawns this file as segment zero of a process, (i.e. when the file is spawned as segment zero the user doing the spawn will have all access privileges that the file owner has including access to the owner's entire directory structure). This switch should rarely be set on if the owner desires to protect his files.

   If owner's access is granted for a file, the owner will be protected only if control cannot be transferred outside of the procedure that this file spawns into.

2) If the write inhibit indicator is set on, the segment which is spawned from this file will be write inhibit , and the file itself will have its WRITE and APPEND access bits set off. That is, neither write nor append operations to the file will be allowed.

Destroy Primitive

Code = 23

DESTRO TRAP,TREE_NAME,TREE_SIZE,BEHALF

The item named by 'TREE_NAME' is uncatalogued (i.e. its directory entry is destroyed). If the item is currently active, it is transformed into a scratch item.

Register	Parameter	Description
X0	(Implicit)	23 = Destroy Primitive code number
X1	TRAP	Trap location
X4	TREE_NAME	Location of tree name of item to be destroyed
X5	TREE_SIZE	Number of words in tree name
X6	BEHALF	1 = Originator's behalf (normal setting) 0 = Owner's behalf

STATUS RETURN WORDS

Word 1

0-17	Undefined
18-35	Logical Status Code (See Appendix A-2)

Open Scratch Primitive Code = 24

OPENS TRAP,DEVID,MAXLEN,ELSIZE

The Open Scratch Primitive opens a zero length, scratch (uncatalogued) file on the device specified by 'DEVID' (disc or drum). The File Reference Number of the file is returned in the status word. Subsequent data transactions involving this file will occur in units specified by 'ELSIZE'. The maximum file length is given by 'MAXLEN' and the type of error recovery by 'STATUS'. The scratch file is granted all of the access attributes.

Register	Parameter	Description
X0	(Implicit)	24 = Open Scratch Primitive code number
X1	TRAP	Trap Location
X4	DEVID	Device Identification of device on which scratch file is to be opened 1 = Disc 2 = Drum (access restricted to privileged users)
X5	MAXLEN	The maximum file length in elements. Attempts to access file outside this length will give an error return.
X7	ELSIZE	Element size, expressed in Bits, in which all subsequent data transactions for the file are to be expressed. The number of bits must be an integral multiple of the unit size for the disc or drum (i.e. a multiple of 32*36=1152). If this field is set to zero, a default element size of 1152 bits (32 words) is set up.

STATUS RETURN WORDS

Word 1

0 - 17	File Reference Number
18 - 35	Logical status code (See Appendix A-2).

Update Primitive

Code = 25

UPDATE TRAP, FRN

The Update Primitive writes current information to the branch. It can be called previous to any close. A final close is no longer required to update information in the branch.

Register	Parameter	Description
X0	(Implicit)	25 = Update Primitive code number
X1	TRAP	Trap location
X2	FRN	File Reference Number

STATUS RETURN WORDS

0-17	Undefined
18-35	Logical status code

REMARKS

An invalid file reference number or an attempt to update a scratch file will return with an error message.

Catalog Directory Primitive — Code = 26

CATDIR TRAP,TREE_NAME,TREE_SIZE,BEHALF,UACCESS,OACCESS

An empty directory with name specified by an N level 'TREE_NAME' is catalogued in the directory specified by the previous N-1 levels of 'TREE_NAME'. Only privileged users are allowed to catalog with N=1, since this corresponds to making an entry at the root level of the directory structure.

Register	Parameter	Description
X0	(Implicit)	26 = Catalog Directory Primitive code number
X1	TRAP	Trap location
X4	TREE_NAME	Location of tree name of directory being catalogued
X5	TREE_SIZE	Number of words in tree name
X6	BEHALF	1 = Originator's behalf (normal setting) 0 = Owner's behalf
A	UACCESS	Universal Access
Q	OACCESS	Owner's Access

Bit 31 READ - Read directory entries includi access control lists

Bit 32 WRITE - Delete or modify specifically named entries

Bit 33 APPEND - Add entries without changing existing entries

Bit 34 EXECUTE - Can use directory for a tree search on a symbolic name

STATUS RETURN WORDS

Word 1

0 - 17	Undefined
18 - 35	Logical Status Code (See Appendix A-2)

Write Access Control List   Code = 27

'RACL TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE

The Write Access Control List Primitive modifies the access control lis which is associated with a given file, directory, or event. The primitive either adds a new user entry to the list, modifies an existing entry, or deletes an existing entry.

Several entries may be added with a given command. The name and access associated with each entry is defined in a five word block of informati which is contained in a buffer of size 'BUFSIZE' which starts at 'BUFLO The format for the five word entry is specified below.

The primitive adds a new entry for a user if none exists previously. Otherwise, it modifies the existing entry. The executive stores status in each five word block if the operation is not successful.

Register	Parameter	Description
X0	(Implicit)	27 = Write Access Control List code number
X1	TRAP	Trap location
X3	BUFLOC	Starting location of buffer in slave program which contains the five word blocks defining names and accesses to be written
X4	TREE_NAME	Location of the tree name of the file or event whose access is being modified
X5	TREE_SIZE	Number of words in the tree name
X6	BEHALF	Behalf Indicator - Bit 17 1 = Originator's behalf (normal setting) 0 = Owner's behalf
X7	BUFSIZE	Number of words in the buffer which contains the five word access control list blocks. The maximum size is 50 words and all intermediate sizes must be a multiple of 5.

STATUS RETURN WORDS

Word 1

0 - 17	Not specified
18 - 35	Logical status code (See Appendix A-2)

Format for the 5 Word Access Control List Block

Words 1-4	The name of the user as it appears in the root directory of the system catalog (i.e. the first level of the user's tree name).
Word 5	
Bits 0-17	Reserved for storing additional status by the executive. The executive stores 777777_8 in this field if the access specified in this block has not been successfully written to the access control list. This occurs when the name specified in Words 1-4 is not catalogued in the root directory.
Bit 29	Erase bit. If this bit is set ON the previously defined entry for the named user will be deleted from the access control list.
Bits 30-35	Access to be granted to the named user Bit 30 -- READ (file) Bit 31 -- WRITE (file) Bit 32 -- APPEND (file) or NOTIFY (event) Bit 33 -- EXECUTE (file) or CAUSE (event) Bit 34 -- LOCK (file,event) Bit 35 -- TRAP (not implemented yet)

REMARKS

If the owner's name appears on the list, the owner's access will be changed in the branch. If the delete bit is on, the owner's access will be set equal to the universal access.

Read Access Control List Code = 28

RDACL TRAP,INDEX,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,NUMBER

The Read Access Control List Primitive reads entries from the access control list of a specified file, directory, or event, and writes them to a buffer in the user program. The name and access for each entry is written in a five word block, similar to that employed in the Write Access Control List Primitive.

Reading starts from the entry number in the control list which is specified by 'INDEX'. The number of entries read, up to a maximum of 10, is given by 'NUMBER'.

Register	Parameter	Description
X0	(Implicit)	28 = Read Access Control List Primitive code number
X1	TRAP	Trap location
X2	INDEX	Number of the access control list entry at which reading is to begin (=1 for first entry).
X3	BUFLOC	Location of buffer into which the five word block containing names and accesses are to be written by the executive
X4	TREE_NAME	Location of the tree name of the file, directory, or event whose access is being read
X5	TREE_SIZE	Number of words in the tree name
X6	BEHALF	1 = Originator's behalf (normal setting) 0 = Owner's behalf
X7	NUMBER	Number of ACL entries to be read, up to a maximum of 10. Reading begins from the entry number stored in -X2-.

STATUS RETURN WORDS

Word 1

0 - 17	Not specified
18 - 35	Logical status code (See Appendix A-2)

Read Access Control List (Continued)

Format for the 5 Word Access Control List Block

Words 1 - 4	The name of the user as it appears in the root directory of the system catalog (i.e. the first level of the user's tree name).
Word 5	
Bits 30-35	Access which has been granted to user Bit 30 -- READ (file) Bit 31 -- WRITE (file) Bit 32 -- APPEND (file) or NOTIFY (event) Bit 33 -- EXECUTE (file) or CAUSE (event) Bit 34 -- LOCK (file,event) Bit 35 -- TRAP (not implemented at present)

Read Directory Primitive Code = 29

RDDIR TRAP,INDEX,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE

The Read Directory Primitive reads the contents of one or more branches from the directory specified by 'TREE_NAME' into a buffer area in the slave program which starts at 'BUFLOC'. The branches are read sequentially, the starting branch being specified by the number 'INDEX'. The number of branches read depends upon the buffer size, 'BUFSIZE', there being one branch read for each ten word block which is assigned. The number of branches read is returned in the upper half of Status Word 1.

Read Access is required for the directory specified.

Register	Parameter	Description
X0	(Implicit)	29 = Read Directory Primitive code number
X1	TRAP	Trap location
X2	INDEX	Index in directory of first branch to be read. (The index is =1 for the initial branch in the dir)
X3	BUFLOC	Starting location of slave buffer area where information from the specified branches will be returned
X4	TREE_NAME	Location of tree name of directory being read
X5	TREE_SIZE	Number of words in tree name
X6	BEHALF	1 = Originator's behalf (normal setting) 0 = Owner's behalf
X7	BUFSIZE	Size of slave buffer area where branch information will be returned. Ten words must be specified for each branch to be returned. The buffer starting location is stored in X3.

STATUS RETURN WORDS

Word 1

0 - 17	The number of branches which have been read into the buffer area
18 - 35	Logical status code (See Appendix A-2)

REMARKS

1) For system information, use the Read Branch Primitive.

Format for Each 10 Word Block of Returned Branch Information

Word 1 Upper	
Bits 0-1	Entry Type 00 - File 01 - Directory 10 - Event 11 - Link
Bit 3	Owner's Directory Privilege Switch (file)
Bits 12-17	Owner's Access (all except Link)
Bits 14-17	Number of names in tree name (link)
Word 1 Lower	
Bit 18	Password Bit (=1 if item has password)
Bit 19	Active Bit (file or event) =1 if item is open
Bit 20	Write inhibit (file)
Bit 21	Access control list bit (not link) =1 if item has ACL
Bit 22	System information bit (file) =1 if file has system information in branch
Bits 31-35	Requestor's access (all except link)
Word 2 Upper	
Bits 0-11	Day item last used (file or event)
Bits 12-16	Universal access (all except link)
Word 2 Lower	
Bits 18-23	Not Used
Bits 24-35	Date Last Modified
Word 3 Upper	File Length (file) or Current Count (event) or No. of Blocks used (root directory entry)
Word 3 Lower	Maximum file length in units (file) or Maximum queue length (event) or Maximum number of blocks allowed (root dir. entry)

Format for each 10 Word Block of Returned Branch Information (cont'd)

Word 4 Upper	Units allocated (file) or
Bit 6	Pass event indicator (event)
Bit 7	Unlock event indicator (event)
Bit 8	Mode (event)
Bit 9	Infinity bit (event)
Bit 10	Time event indicator (event)
Word 4 Lower	System ID. (file) =0 since no system information returned (This field may be non-zero for the Read Branch primitive) or Time Limit (event)
Words 5 and 6	Password (returned only for priv. ID)
Words 7-10	Branch name

NOTE: The first six words of returned information has the same format in the Read Branch, Read Directory, and Read Link Primitives. The last 4 words are different for each primitive.

Open Working Directory Primitive   Code = 30

OPENW TRAP,TREE_NAME,TREE_SIZE,BEHALF

The Open Working Directory Primitive resets the user's current working directory to the directory specified by 'TREE_NAME'. Execute access is required on the directory so specified.

Register	Parameter	Description
X0	(Implicit)	30 = Open Working Directory code number
X1	TRAP	Trap location
X4	TREE_NAME	Location of tree name of working directory
X5	TREE_SIZE	Number of words in tree name
X6	BEHALF	1 = Originator's behalf (normal setting) 0 = Owner's behalf

STATUS RETURN WORDS

Word 1	
0 - 17	Not used
18 - 35	Logical status code (See Appendix A-2)
Word 2	Originator (For use by special SPAWN)

REMARKS

There are certain instances when it becomes necessary to reference, by tree name, several items which are catalogued in the same directory (e.g., when a user is initially opening several files at sign on). The open working directory primitive enables the user to expedite the tree search for such items by allowing the search to start at the level of the working directory specified instead of at the root level.

When a process is initially spawned by the LISTENER, its current working directory is set to the user's main directory (i.e. the directory at the level immediately below the root directory). The initial current working directory, so specified, must be referenced using the originator's behalf. The current working directory can be changed by re-issuing the open working directory primitive with the tree name for the new directory. If a father spawns a son, the son's current working directory is initialized to the fathers.

The special symbol '*' is used as the first component of a tree name

REMARKS (Cont'd)

to indicate that referencing is to start from the directory specified by the current working directory. The '*' may be followed by a tree name of any level, up to the maximum allowable by the system.

Note that the '*' convention to be described can be used in specifying the tree name for any file or event primitive which utilizes the tree name parameter.

The following examples should clarify the use of the '*' convention and the current working directory.

Example 1

Assume the open working directory primitive is issued with the tree name as specified below:

ACI 6,GJF	Name of user's main directory
ACI 6,FORTRAN-PROGRAMS	Name of directory in main directory

The current working directory for this tree name is the directory GJF/FORTRAN-PROGRAMS.

File FORTA in the directory FORTRAN- PROGRAMS can be referenced by the following tree name using the '*' convention to denote use of the current working directory.

ACI 6,*	Indicator that initial components of tree name are specified by tree name of current working directory
ACI 6,FORTA	Name of file in current working directory.

In example 1 the tree name *,FORTA is equivalent to the tree name GJF FORTRAN-PROGRAMS/FORTA.

Example 2

Consider the working directory whose tree name is GJF/DIRECTORYA/DIRECTORYB.

Then the tree name * DIRECTORYC/DIRECTORYD/FILEA is equivalent to the tree name:

GJF/DIRECTORYA/DIRECTORYB/DIRECTORYC/DIRECTORYD/FILEA

REMARKS (Cont'd)

This second example illustrates that the current working directory can both reference and be followed by multiple level tree names.

Access when using the '*' convention

The use of the '*' convention does not alter the access which is allowed the user. That is, the access associated with the current working directory is the same as the access that would be granted if the directory was referenced from the root level.

Read Branch Primitive   Code = 31

```
DBRN TRAP,SYSID,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF
```

The Read Branch Primitive reads the contents of the branch specified by the tree name, into a 10 word block in the slave program which starts at 'BUFLOC'. If a system identifier is specified, system information (previously entered by the write system information primitive) will also be supplied. The format for the returned branch information is specified below.

Read access is required for the directory specified.

Register	Parameter	Description
X0	(Implicit)	31 = Read Branch Primitive code number
X1	TRAP	Trap location
X2	SYSID	Specifies either the System Identifier Number or a Block Number. The Block Number denotes the block of system information to be read. System Identifier Number Code 0 - System information not requested 1 - TSS 2 - MARK2 3 - GECOS Block Number Code -J Specifies that the Jth block of system information is to be read (J=1, 2, 3 at present). Note the minus preceding J.
X3	BUFLOC	Starting location of 10 word buffer in a slave program where branch information is returned.
X4	TREE_NAME	Location of the tree name of the directory
X5	TREE_SIZE	Number of words in treename. If X5=0, X4 is assumed to contain the FRN of the item being accessed.
X6	BEHALF	1 = Originator's behalf (normal setting) 0 = Owner's behalf

STATUS RETURN WORDS

Word 1

0 - 17	Not specified
18 - 35	Logical status code (See Appendix A-2)

Format for Returned Branch Information

Word 1 Upper	
Bits 0-1	Entry type 00 - File 01 - Directory 10 - Event 11 - Link
Bit 3	Owner's Dir. Priv. Switch (file)
Bits 12-17	Owner's Access (all except link)
Bits 14-17	Number of names in tree name (link)
Word 1 Lower	
Bit 18	Password Bit (=1 if item has password)
Bit 19	Active Bit (file or event) =1 if item is open
Bit 20	Write inhibit (file)
Bit 21	Access control list bit (not link) =1 if item has ACL
Bit 22	System information bit (file) =1 if file has system information in branch (See Remark 3)
Bits 31-35	Requestor's access (all except link)
Word 2 Upper	
Bits 0-11	Day item last used (file or event)
Bits 12-16	Universal access (all except link)
Word 2 Lower	Not used
Word 3 Upper	File length (file) or Current count (event) or Number of blocks used (root directory entry)
Word 3 Lower	Maximum file length in units (file) or Maximum queue length (event) or Maximum number of blocks allowed (root dir. entry)

Format for Returned Branch Information (cont'd)

Word 4 Upper	Units allocated (file) or
Bit 6	Pass event indicator (event)
Bit 7	Unlock event indicator (event)
Bit 8	Mode (event) 0=steady state 1=transient
Bit 9	Infinity bit (event)
Bit 10	Time event indicator (event)
Word 4 Lower	System ID (file) =0 if no system information returned = ID of system for returned information or Time limit (event)
Words 5 and 6	Password (returned only for priv. ID).
Words 7-10	System information

NOTE: The first 6 words of information have the same format as the information returned by the Read Directory and Read Link Primitives. The last 4 words are different for each primitive

REMARKS

1) System information can only be obtained from the Read Branch Primitive. (It is not returned by the Read Directory Primitive).

2) The Read Branch Primitive can never return information about a link.

3) In general there may be several blocks of system information associated with a given branch, there being one block for each system which has written such information. The System Information Bit (Bit 22 in the lower half of Word 1) provides the following information about these blocks.

   The System Information Bit is set ON if:

   a) The block of system information requested is not the last block of system information in the branch

REMARKS (Cont'd)

b) System information is present in the branch but none was returned (i.e., either 'SYSID' =0 was specified or the ID (or block) was invalid).

The system information Bit is set OFF if

a) The block of system information requested is the last block i the branch or

b) There is no system information at all in the branch

Note the above conditions hold whether 'SYSID' is an ID or a block number.

4) If system information is returned, the SYSID is given in Word 4 Lower of the information block.

Read Link Primitive Code = 32

RDLNK TRAP,BUFOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE

The Link name is returned in the user specified buffer area. If the buffer size specified is too small, then an error status is returned. A buffer size of 96_{10}(6 X 16) is the maximum size possible and therefore, will guarantee a correct buffer size for any link name.

Register	Parameter	Description
X0	(Implicit)	32 = Read Link Primitive code number
X1	TRAP	Trap location
X3	BUFLOC	Starting location of buffer area
X4	TREE_NAME	Location of tree name
X5	TREE_SIZE	Number of words in tree name
X6	BEHALF	1=originator's behalf (normal setting) 0=owner's behalf
X7	BUFSIZE	Size of buffer area must be a multiple of six words.

STATUS RETURN WORDS

Word 1

0 -17	Not specified
18 -35	Logical status code (See Appendix A-2)

FORMAT FOR RETURNED BUFFER INFORMATION

Words 1 to 6	Same as read directory
Words 7 to 12	First level link name
Words 13 to 18	Second level link name
Words 6N+1 to 6N+6	N'th level link name

Write System Information Primitive Code = 33

WSINF TRAP,SYSID,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,DELETE

The Write System Information Primitive allows various subsystems (TSS, BASIC, GECOS, etc.) to append additional system information to the file branch specified by 'TREE_NAME'. The subsystem specifies the information to be added in a four word information buffer located at 'BUFLOC'. If this is the first time information is to be written for the particular subsystem, a new information block is added. Otherwise, the existing information block is modified. An existing information block is deleted if 'DELETE' is set ON.

Register	Parameter	Description
X0	(Implicit)	33 = Write System Information Primitive code number
X1	TRAP	Trap location
X2	SYSID	Sub-system identifier number 1 - TSS 2 - MARK2 3 - GECOS
X3	BUFLOC	Location of the four words of sub-system information to be written
X4	TREE_NAME	Location of the tree name of the file to whose branch the additional sub-system information is to be added.
X5	TREE_SIZE	Number of words in tree name (a multiple of six). If X5=0, X4 is assumed to contain the FRN of the item being accessed.
X6	BEHALF	1 = Originator's behalf (normal setting) 0 = Owner's behalf
X7	DELETE	Delete switch =0 Add or modify block =1 Delete block

STATUS RETURN WORDS

Word 1

0 - 17	Not specified
18 - 35	Logical status code (See Appendix A-2)

Catalog Link Primitive Code = 34

CATLK TRAP,LINK_NAME,LINK_SIZE,TREE_NAME,TREE_SIZE,BEHALF

The CATLK primitive catalogs a link with a name of TREE_NAME. The link points to the item specified in the tree name LINK_NAME. All passwords associated with the item must be included in the LINK_NAME. The TREE_NAME is the name of the link, and the LINK_NAME is the name of the actual file pointed at by the link.

Register	Parameter	Description
X0	(Implicit)	34 = Catalog Link Primitive code number
X1	TRAP	Trap location
X2	L-NAME	Location of link name
X3	L-SIZE	Number of words in link name, must be a multiple of six
X4	TREE_NAME	Location of tree name
X5	TREE_SIZE	Number of words in tree name, must be a multiple of six
X6	BEHALF	1=originator's behalf (normal setting) 0=owner's behalf

STATUS RETURN WORDS

Word 1

0 -17	Not specified
18-35	Logical status code (See Appendix A-2)

REMARK

Example - if user X attempts to link to user Y's file, then on a 2 level basis,

TREENAME = X/LINK

LINKNAME = Y/FILE

Write Branch Primitive Code = 35

WTBRN TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF

The WTBRN primitive is used to alter the information or name of a file, event or directory. It allows slave processes to change certain information in the branch (that is, directory, file, event and link blocks).

F=File Block; E=Event Block
D=Directory Block; L=Link Block

The information that may be changed is:

1) password and name for F,E,D and L (the password is changed only if the password bit is on)
2) owners and universal access in F,E, and D
3) maximum length in F and E
4) owners access directory privilege switch in F
5) time limit in E

In the case of events, the following checks are made:

1) Time limit must be less than 1800 sec. and greater than zero. Otherwise, an error is returned.
2) Maximum Q length must be less than 15 and greater than zero. Otherwise, an error is returned.

In the case of files, the OADPS is changed in the AFT entry if the file is open. Also, the maximum length of the file is not changed if the write inhibit bit is on.

In the case of root directories, the maximum number of blocks allowed is changed if the user is a privileged user.

The buffer should be set up as a 10 word buffer with the format of the first six words of the RBRN buffer. The last 4 words contain the new name of the entry. If no change in the name is desired, the original name must be inserted.

Register	Parameter	Description
X0	(Implicit)	35 = Write Branch Primitive code number
X1	TRAP	Trap location
X3	BUFLOC	Location of 10-word buffer
X4	TREE_NAME	Location of tree name
X5	TREE_SIZE	Number of words in tree name
X6	BEHALF	1=originator's ID (normal setting) 0=owner's ID

STATUS RETURN WORDS

Word 1

0 -17	Undefined
18-35	Logical status code (See Appendix A-2)

Lock Primitive

Code = 36

LOCK TRAP,RN

The Lock Primitive enables a user to lock a file or event which he has opened. This prevents other users from operating on it until he has unlocked it. If the user has lock access, the executive inhibits interrupts while determining whether the item referenced by 'RN' (Reference Number) is currently locked. If the item is locked, interrupts are restored and a busy signal returned in the status word.

Otherwise, the item is locked on behalf of the user, interrupts are restored, and a successful return is made.

Register	Parameter	Description
X0	(Implicit)	36 = Lock Code Number
X1	TRAP	Trap address
X2	RN	Reference number of file or event to be locke

STATUS RETURN WORDS

Word 1

0 - 17	Undefined
18 - 35	Logical Status Code (See Appendix A-2)

Unlock Primitive Code = 37

UNLCK TRAP,RN

The Unlock Primitive enables the user to unlock a file or event, specified by 'RN' (Reference Number", that he had previously locked. A successful return is also given if the item referenced was not locked to begin with.

Register	Parameter	Description
X0	(Implicit)	37 = Unlock code number
X1	TRAP	Trap address
X2	RN	Reference number of file or event to be locked

STATUS RETURN WORDS

Word 1

0 - 17	Undefined
18 - 35	Logical Status Code (See Appendix A-2)

Notify Primitive Code = 38

NOTIF TRAP,ERN,CTRAP,STATE

The process which issues the notify primitive is trapped at location 'TRAPC' when the event specified by reference number 'RN' is caused with the given 'STATE'. If the process which causes event 'RN' has specified a message, this message is returned in Status Word 2 of TRAPC.

The location 'TRAP' is associated with the issuance of the notify. Control is returned to 'TRAP' after the notify has been completed with the acceptance (or rejection) indicated in the Status Word 1.

Register	Parameter	Description
X0	(Implicit)	38 = Notify primitive code number
X1	TRAP	Trap location associated with the issuance of the notify. This trap is similar to the trap associated with the issuance of any other primitive. When the operation of issuing the notify is completed, control returns to 'TRAP' with the success indicated in the corresponding status words.
X2	ERN	Event Reference Number of the event, whose causing will result in the process being notified.
X3	TRAPC	Trap location associated with the causing of event 'RN'. Control is returned to 'TRAPC' after some process issues a cause to event 'RN' with a state matching that specified by 'STATE'.
A	STATE	Value which must be specified by the CAUSE for the notify to occur. If the state is all ones (i.e. 777777777777_8), the process will be notified regardless of the state specified in the cause. Note, state can only be defined for transient mode events. For steady state events, state must be set to zero. If not, the primitive will be rejected.

STATUS RETURN WORDS

TRAP

Word 1

0 - 17	Not specified
18 - 35	Logical Status Code (See Appendix A-2)

Notify Primitive (Continued)

STATUS RETURN WORDS (Cont'd)

TRAPC

Word 1

0 - 17	FRN of passed item (pass event)
18 - 35	Logical Status Code (See Appendix A-2) Code is zero if trap has returned due to issuing of cause. Error code if trap returns because time limit is exceeded.

Word 2

Non-pass event	
0 - 35	Message (optional) returned by process issuing cau
Pass Event	
0 - 17	Message (optional) returned by process issuing cau
18 - 35	Accounting ID for pass event

WARNING

If a slave process continues processing after issuing a NOTIFY (i.e. it does not issue the PAUSE primitive) then it must be prepared to handle the notify trap whenever it occurs. In particular this means that the trap routine for the NOTIFY and all other trap routines which may occur at the same time as the NOTIFY must be coded to exit with a RET to the trap routine exit location. This is necessary since the order in which the traps return cannot be pre-determined. (The programmer should review the discussion on trap handling in Section II if this is not clear).

Care must also be taken in using the PAUSE primitive when a NOTIFY trap is outstanding. The trouble occurs when the trap for the primitive returns "fast" so that the PAUSE is issued when it is not really required. For example, suppose a primitive is issued followed by a PAUSE, and the executive returns control to the trap routine (i.e. the fast return) rather than to the instruction following the primitive MME. The trap routine, using the standard RET to the trap exit location, causes the PAUSE to be issued, even though the primitive has already been completed. Normally the PAUSE would be ignored by the executive since there would be no additional traps outstanding. However, since there is a NOTIFY trap outstanding, the PAUSE will be accepted and the process will go into a blocked state. The process will remain blocked until the CAUSE for the NOTIFY is issued which is not generally what the programmer wants. To avoid this possibility when a NOTIFY trap is outstanding, DO NOT ISSUE THE PAUSE IF THE TRAP FOR THE PRIMITIVE OPERATION HAS ALREADY RETURNED.

Notify Primitive (Continued)

The following code may be employed to determine whether a PAUSE should be issued when a NOTIFY trap is outstanding.

```
STZ     TRAP+2          Zero trap return location before issuing MM
MME     0               MME initiating the primitive operation
INHIB   SAVE,ON         Inhibit interrupts
SZN     TRAP+2          Test if trap has returned
TNZ     *+4             If so, bypass the PAUSE
LDXO    15,DU           Otherwise issue the PAUSE
MME     0
INHIB   RESTORE         Restore interrupts
TRA     *-4             Insure that exit is when trap returns
```

Cause Primitive Code = 39

CAUSE TRAP,ERN,STATE,MESSAGE,NUMBER,FRN,ACCESS

This primitive causes a subset of the processes awaiting event 'RN' whose state matches 'STATE' to be trapped. The number of processes to be trapped is specified by 'NUMBER'. (The address at which a process is trapped, and the state to be matched against, is specified by the Notify Primitive.)

'MESSAGE' specifies an optional one word message which the causing process can return to the process being notified.

Upon completion of the operation, the process issuing the cause is trapped at location 'TRAP' with the number of processes notified returned in the upper half of Status Word 1.

Register	Parameter	Description
X0	(Implicit)	39 = Cause Primitive code number
X1	TRAP	Trap location associated with issuance of the Cause
X2	ERN	Reference number of event being caused
X3	NUMBER	Number of processes (with state matching 'STATE') which are to be notified (i.e., trapped). If 'NUMBER'=0, all processes (with matching state) awaiting notification will be trapped. The current count is incremented by 'NUMBER' when the event is caused and decremented by one for each process which is trapped (for steady state events).
X6	RN	RN of item to pass (for pass mode)
X7	ACCESS (Bits 13-17)	Access on item passed (for pass mode)
A	STATE	The state parameter is defined only for transient mode events. It must be set to zero for steady state events. Only those processes on the event queue whose state matches that specified by 'STATE' will be trapped. If the state word is set to all ones (i.e. 777777777777_8) the number of processes specified by 'NUMBER' will be trapped regardless of the state in their queue entry.
Q	MESSAGE	An optional one word message that the causing process can send to the process being notified.

Cause Primitive (Continued)

STATUS RETURN WORDS

Word 1

0 - 17	The number of processes trapped as a result of issuing CAUSE. If this field is zero for a CAUSE to a transient mode event, then there is no NOTIFY outstanding for the CAUSE. The CAUSE should be reissued in this case.
18 - 35	Logical Status Code (See Appendix A-2)

Delete Event Primitive   Code = 40

DELET TRAP, ERN, STATE

An entry for this process which has the specified state is deleted fr(
the event queue specified by 'RN' (Reference Number).

Register	Parameter	Description
X0	(Implicit)	40 = Delete event code number
X1	TRAP	Trap address
X2	ERN	Reference number of event to be deleted
A	STATE	State (must be zero for steady state event)

STATUS RETURN WORDS

Word 1

0 - 17	=1 if an entry was successfully deleted
18 - 35	Logical status code (See Appendix A-2)

Uncause Event Primitive Code = 41

UNCAU TRAP,RN,NUMBER

The Uncause Primitive decreases the current count of the event specified by 'RN' (Reference Number) by the amount specified by 'NUMBER'.

The Uncause Primitive is meaningful only for events in the steady state mode since the current count will not be allowed to go negative.

Register	Parameter	Description
X0	(Implicit)	41 = Uncause event primitive code number
X1	TRAP	Trap address
X2	RN	Event reference number
X3	NUMBER	Number by which current count is to be decreased

STATUS RETURN WORDS

Word 1

0 - 17	Undefined
18 - 35	Logical Status Code (See Appendix A-2)

REMARKS

1) See "Summary of Macro Calls for Event Primitives" for a description of CURRENT COUNT.

Open Scratch Event Primitive   Code = 42

OPSCE TRAP,TIMLIM,MODE,MAXLEN

The Open Scratch Event Primitive creates a scratch event, returning the reference number for this event in the upper half of Status Word 1. Most further references to this event will be in terms of the reference number.

Register	Parameter	Description
X0	(Implicit)	42 = Open Scratch Event Primitive code number
X1	TRAP	Trap address
X3	TIMLIM	Time limit for process awaiting event (secs.) If zero, the maximum limit of 1/2 hour is granted.
X4	MODE	Mode bit and pass event indicator Bit 17 - Mode Bit = 0 Steady State mode = 1 Transient mode Bit 16 - Pass Event Indicator = 0 not an event of type "pass" = 1 scratch event is type "pass" (Only for transient mode events)
X5	MAXLEN	Maximum queue length for event (1 to 16). If set to zero, maximum of 16 is granted.

STATUS RETURN WORDS

Word 1

0 - 17	Event reference number of opened scratch event
18 - 35	Logical status code (See Appendix A-2)

System Status Measurements Primitive   Code = 45

MSTA TRAP,PLOC,LENGTH,BUFLOC

The 'status' primitive transfers the contents of absolute core locations into a slave program buffer area. It also returns a pointer to the MMDDT symbol table in the upper half of the first status word. This primitive can be issued only by privileged processes.

Register	Parameter	Description
X0	(Implicit)	45 = System Status Measurement Primitive code number
X1	TRAP	Trap address
X2	PLOC	Address of argument list in slave program
X3	LENGTH	# of arguments in list
X4	BUFLOC	Address of buffer where the status is to be put

REMARK

A check is made to see that the address of all the arguments is within the bounds of the BAR.

STATUS RETURN WORDS

Word 1

0 -17	Pointer to the MMDDT symbol table
18-35	Logical status code (See Appendix A-1)

FORMAT OF PARAMETER LIST

Each argument is 1 word in length

upper half of each word is the # of words to be read

lower half of each word is the absolute core location to start reading from

Measure Read Me Primitive   Code = 46

RD ME TRAP,BUFLOC,BSIZE

This primitive is issuable only by certain privileged processes (the Listener, in particular). It is a request by a slave program to have a block of information transferred into the accounting stream. It is assumed the block specified contains one or more complete accounting blocks, each properly identified.

Register	Parameter	Description
X0	(Implicit)	46 = Measure Read Me Primitive code numbe
X1	TRAP	Trap Location
X2	BUFLOC	Beginning address of buffer
X3	BUFSZE	Length of buffer (in words)

REMARK

A maximum of 32 words will be transferrable by one primitive call.

STATUS RETURN WORDS

Word 1

0 -17	Undefined
18-35	Logical status code (See Appendix A-1)

Create Segment Primitive Code = 47

CRSG TRAP,FRN,SEGNUM

The create segment primitive creates a segment from a specified file. It allows creation of a write inhibited segment, from a file, after the process has been spawned. The segment will be write inhibited only if the file was write inhibited.

Register	Parameter	Description
X0	(Implicit)	47 = Create Segment Primitive code number
X1	TRAP	Trap Location
X2	FRN	File Reference Number
X3	SEGNUM	Segment Number

STATUS RETURN WORDS

Word 1

0 -17	Undefined
18-35	Logical Status Code (See Spawn Primitive)

REMARKS

Access

Seg 0	Execute
Seg 1-3	Execute or Read

Write Me Primitive Code = 48

WRITEM TRAP,TRCPR,PLOC,LENGTH

The Write Me Primitive allows writing into absolute core location and may start 'sample' tracking of the process.

Register	Parameter	Description
X0	(Implicit)	48 = Write Me Primitive code number
X1	TRAP	Trap Location
X2	TRCPR	Tracking Parameter 0=to be ignored 1=start tracking process
X4	PLOC	Location of array of arguments
X5	LENGTH	Length of array of arguments (must be less than or equal to 64)

STATUS RETURN WORDS

Word 1

0 -17	Not specified
18-35	Logical status code (See Appendix A-1)

FORMAT OF AN ARGUMENT PAIR:

First Word: DU = Absolute Address

Second Word: New Value of Location

Who Am I Primitive — Code = 49

WAMI TRAP,CORELOC

The Who Am I primitive returns the first level of the current working directory and the Acode in a five word buffer which the user specifies.

Register	Parameter	Description
X0	(Implicit)	49 = Who Am I primitive code number
X1	TRAP	Trap Location
X4	CORELOC	Location of 5-word buffer

STATUS RETURN WORDS

Word 1

0 -17	Not specified
18-35	Logical status code 5 - I/O out of bounds

FORMAT OF RETURNED BUFFER INFORMATION

Words 1-4	First level of current working directory name
Word 5	Acode

Request Working Directory Primitive   Code = 51

RQWD TRAP,CORELOC

The Request Working Directory returns up to 6 levels of the current working directory in a 36 word buffer which the user specifies. The Acode is returned in the lower half of status return word 2.

Register	Parameter	Description
X0	(Implicit)	51 = Request Working Directory Primitive code number
X1	TRAP	Trap Location
X4	CORELOC	Location of 36-word buffer

STATUS RETURN WORDS

Word 1

0 -17	Number of levels passed back
18-35	Logical status code 5 - I/O out of bounds

Word 2

0 -17	Not specified
18-35	Acode

ASCII treenames to be returned in buffer.

APPENDIX A

SUMMARY OF LOGICAL STATUS CODES

LOGICAL STATUS CODE FOR I/O PRIMITIVES

Value Dec	Value Oct	Meaning
0	0	Successful operation
1	1	Invalid FRN
2	2	Invalid access specified
3	3	Operation cannot be done at this time (software)
4	4	Invalid operation for device type
5	5	I/O would be out of bounds
6	6	Amount requested greater than file length
7	7	Element size not a multiple of unit size
8	10	Too many units specified for device
9	11	Invalid mode specified
10	12	Hardware error - operation not complete
11	13	Device unavailable (hardware)
12	14	Parity error in transfer - operation completed
13	15	No freelist available for operation entry
14	16	End-of file encountered
15	17	Subchannel not currently connected
17	21	Element count invalid
19	23	Referencing a locked item
20	24	Drum transfer must start at even core location
21	25	Reading into a protected segment
22	26	No file storage available
23	27	*Terminate-writing* this subchannel
24	30	End of tape
25	31	Subchannel connection not established
27	33	*Quit-Reading* this subchannel
29	35	Timer Run Out on operator's console or ring already present on CLM channel
30	36	A to D conversion stopped, No read
31	37	Line-break has occurred
32	40	TTY parity error (used only for listener)

Note: The primitives, classified as 'I/O', for which the above status code is returned are:

04--Sequential Read
05--Append to File
06--Read Random File
07--Write Random File
08--Scratch File
09--Set Pointer
10--Request Status
11--Request Date and Time
49--Who Am I

LOGICAL STATUS CODE FOR FILE AND EVENT PRIMITIVES

Value Dec	Value Oct	Meaning
0	0	Operation Was Successful
1	1	Illegal File Reference Number
2	2	Access Denied
3	3	Operation Cannot Be Done At This Time
4	4	Branch Improperly Formed
5	5	Name Invalid
6	6	Directory Full
7	7	Unrecoverable Error
8	10	Directory Not Empty
9	11	Time Limit Exceeded
10	12	Invalid Command
11	13	Item Locked
12	14	Illegal Element Size
13	15	Illegal Device ID
14	16	Illegal Password
15	17	Buffer too Small
16	20	Access denied to Open on Behalf of Owner
17	21	Too many File Operations Outstanding
18	22	#AIT free list entries under threshold
19	23	KIT Free list is empty
20	24	No execute access on Directory Above Item Being Accessed
21	25	Device not allocated (for deallocate event only)

Note: The Primitives, classified as 'File and Event', for which the above status code is returned are:

20--Open
21--Close
22--Catalog
23--Destroy
24--Open Scratch
25--Update
26--Catalog Directory
27--Write Access Control List
28--Read Access Control List
29--Read Directory
30--Open Working Directory
31--Read Branch
32--Read Link
33--Write System Information
34--Catalog Link
35--Write Branch
36--Lock
37--Unlock
38--Notify
39--Cause
40--Delete Entry
41--Uncause
42--Open Scratch Event

SUMMARY OF STATUS RETURNS FOR CONTROL PRIMITIVES

Set Up Fault Vector, Set Up Squeze Mode, Enter Squeze Mode

Dec	Oct	Meaning
0	0	Successful Return
1	1	Out of Bounds
2	2	Upper Portion of Slave Fault Vector out of Bounds
4	4	Squeze Mode Not Set Up

Spawn and Create Segment

Dec	Oct	
4	4	Segment zero must be specified
5	5	Illegal segment specification
6	6	Invalid access on segment specification
7	7	Illegal File reference number
8	10	Only a file or event may be passed
9	11	Resources not available
12	14	Argument list must be at least 14 words long

Segment Primitives

Open, Close, Change Length, Exchange

Dec	Oct	Meaning
1	1	Segment number out of range
2	2	Segment not known
4	4	I/O activity in progress
5	5	Segment is write inhibit
6	6	Requested segment length equals zero
7	7	Segment length out of range
8	10	Segment already opened

APPENDIX B

SUMMARY OF THE MODE PARAMETERS FOR DIFFERENT DEVICE TYPES

SUMMARY OF THE MODE PARAMETERS FOR DIFFERENT DEVICE TYPES

Disc, Drum

Modes issued thru READ
- 5 - Rewind

Operator's Console

Modes issued through APEND
- 0 - Status
- 1 - Attention
- 2 - Apend
- 3 - Standby

Modes issued through READ
- 0 - Status
- 1 - Attention
- 2 - Read
- 3 - Standby

Card Reader

- 0 - Status
- 1 - Attention
- 2 - Read Card Binary
- 3 - Read Card Hollerith
- 4 - Read Card Mixed

Card Punch

- 0 - Status
- 1 - Attention
- 2 - Write Card Binary
- 3 - Write Card Hollerith
- 4 - Write Card Hollerith Edited

Magnetic Tape

Modes issued through READ
- 0 - Request Status
- 1 - Read Tape Decimal
- 2 - Read Tape Binary
- 3 - Set High Density
- 4 - Set Low Density
- 5 - Rewind
- 6 - Rewind/Unload
- 7 - Forward Space File
- 8 - Backspace File
- 9 - Forward Space Record
- 10 - Backspace Record

Magnetic Tape (Con't)

- 11 - Read Decimal Recovery
- 12 - Read Binary Recovery
- 13 - Read Tape 9
- 14 - Read Tape 9, Recovery
- 15 - Reread Decimal
- 16 - Reread Binary
- 17 - Set File Protect

Memory Interface

- 0 - Status
- 1 - Attention
- 2 - Read or Append

Magnetic Tape

Modes issued through APEND
- 0 - Status
- 1 - Write Decimal
- 2 - Write Binary
- 3 - Erase
- 4 - Write End-of-File (normal use)
- 5 - Write EOF Decimal
- 6 - Write EOF Binary
- 7 - Append Decimal Recovery
- 8 - Append Binary Recovery
- 9 - Write Tape 9
- 10 - Write Tape 9, Recovery

Line Printer
- 0 - Status
- 1 - Attention
- 2 - Write edited continuous, no slew
- 3 - Edited, no slew
- 4 - Edited, slew 1 line
- 5 - Edited, slew 2 lines
- 6 - Edited, slew top of form
- 7 - Non-edited, no slew
- 8 - Non-edited, slew 1 line
- 9 - Non-edited, slew 2 lines
- 10 - Non-edited, slew top of form
- 11 - Slew 1 line
- 12 - Slew 2 lines
- 13 - Slew top of form

Summary of the Mode Parameters for Different Device Types (con't)

Communications Line Multiplexers

Generally the CLM10 Modes are identical
to the teletype modes (See Below).

CLM10 Modes

Modes issued thru APPEND or READ

Oct	Dec	
00	00	Status Request
01	01	Initialize CLM10 system
02	02	Not used
03	03	Unmask one SBCH
04	04	Mask one SBCH
05	05	Accept ring
06	06	Initiate ACU

Modes issued only thru *APPEND*

07	07	Write to one SBCH
10	08	Terminate writing to one SBCH

Modes issued only thru *READ*

11	09	Quit reading from one SBCH
12	10	Accept attention read
13-17	11-15	Not presently in use

Read Modes transmitting all LF's

20	16	Issue LF and READ to CR or TRO
21	17	Read to CR, TRO, EOM, or EOT
22	18	Issue LF and READ to TRO with LF
23	19	READ to TRO or EOT
24-27	20-23	Not presently in use

Begin READ modes suppressing CRS not following NON-CRS

30	24	Issue LF and READ to CR
31	25	READ to CR
32	26	Issue LF and READ to TRO with LF
33	27	READ to TRO
34-37	28-31	Not presently in use

SUMMARY OF MODE PARAMETERS (Continued)

TTY

Modes which may be issued through APEND or READ

- 0 - Status request
- 1 - Disable all subchannels (SBCHS) (Init. TTY system)
- 2 - Enable all SBCHS
- 3 - Unmask one SBCH
- 4 - Mask one SBCH
- 5 - Accept ring
- 6 - Initiate ACU

Modes which may be issued only thru APEND

- 7 - Write to one SBCH
- 8 - Terminate writing to one SBCH

All of the modes which follow may be issued only through READ

- 9 - Quit reading from one SBCH
- 10-15- Not presently in use

Read modes transmitting all LF's

- 16 - Issue LF and READ to CR
- 17 - Just READ to CR
- 18 - Issue LF and READ to TRO with LF
- 19 - Just READ to TRO
- 20-23- Not presently in use

Read modes suppressing CR's not following Non-CR's

- 24 - Issue LF and READ to CR
- 25 - Just READ to CR
- 26 - Issue LF and READ to TRO with LF
- 27 - Just READ to TRO
- 28-31- Not presently in use

Note on READ-to-CR: Occurrence of TRO before CR terminates the READ.

APPENDIX C

Primitive Command Summary

SUMMARY OF PRIMITIVE COMMANDS AND THEIR CODE NUMBERS

Primitives Executed Directly by the Master Mode Executive

Primitive Code and Name	Primitive Classification
00--Privileged Command From Slave Exec	CONTROL
01--Set Up Fault Vector	CONTROL
02--Set Up Squeze Mode	CONTROL
03--Enter Squeze Mode	CONTROL
04--Read	I/O
05--Append	I/O
06--Random Read	I/O
07--Random Write	I/O
08--Scratch File	I/O
09--Set Pointer	I/O
10--Request Status	I/O
11--Request Date and Time	CONTROL
12--Request Elapsed Run Time	CONTROL
45--System Status Measurement	CONTROL
46--Measure Read Me	CONTROL
48--Write Me	CONTROL
49--Who Am I	CONTROL
51--Request Working Directory	CONTROL

Primitives Executed by the Slave Mode Executive

13--Spawn	CONTROL
14--Terminate	CONTROL
15--Pause	CONTROL
16--Open Segment	CONTROL
17--Close Segment	CONTROL
18--Change Segment Length	CONTROL
19--Exchange Segments	CONTROL
20--Open	FILE AND EVENT
21--Close	FILE AND EVENT
22--Catalog	FILE AND EVENT
23--Destroy	FILE AND EVENT
24--Open Scratch	FILE
25--Update	FILE
26--Catalog Directory	FILE
27--Write Access Control List	FILE
28--Read Access Control List	FILE
29--Read Directory	FILE
30--Open Working Directory	FILE
31--Read Branch	FILE
32--Read Link	FILE
33--Write System Information	FILE
34--Catalog Link	FILE
35--Write Branch	FILE
36--Lock	FILE AND EVENT
37--Unlock	FILE AND EVENT
38--Notify	EVENT
39--Cause	EVENT
40--Delete Entry	EVENT
41--Uncause	EVENT

Primitives Executed by the Slave Mode Executive (con't.)

42--Open Scratch Event	EVENT
47--Create Segment	CONTROL

Summary of Macro Calls for System Primitives

Primitive Code	Macro Call	
01	SETFV	TRAP,CORELOC
02	SETSQ	TRAP,LOCTBL
03	SQUEZ	TRAP,REGS,IC
04	READ	TRAP,FRN,CORELOC,N,MODE
05	APEND	TRAP,FRN,CORELOC,N,MODE
06	RRF	TRAP,FRN,FILELOC,CORELOC,N
07	WRF	TRAP,FRN,FILELOC,CORELOC,N
08	SCR	TRAP,FRN,FILELOC
09	SPTR	TRAP,FRN,N
10	RQST	TRAP,RN
11	RQDT	No Arguments
12	RQERT	No Arguments
13	SPAWN	TRAP,PLOC,LENGTH
14	TERM	No Arguments
15	PAUSE	No Arguments
16	OSEG	TRAP,SEGNUM,LENGTH
17	CLSEG	TRAP,SEGNUM
18	CHSEG	TRAP,SEGNUM,LENGTH
19	EXSEG	TRAP,SEG1NUM,SEG2NUM
20	OPEN	TRAP,TREE_NAME,TREE_SIZE,BEHALF,ELSIZE,ACCESS
21	CLOSE	TRAP,RN
22	CATLOG	TRAP,RN,TREE_NAME,TREE_SIZE,SWITCH,UACCESS,OACCESS
23	DESTRO	TRAP,TREE_NAME,TREE_SIZE,BEHALF
24	OPENS	TRAP,DEVID,MAXLEN,ELSIZE
25	UPDATE	TRAP,FRN
26	CATDIR	TRAP,TREE_NAME,TREE_SIZE,BEHALF,UACCESS,OACCESS
27	WRACL	TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE
28	RDACL	TRAP,INDEX,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,NUMBER
29	RDDIR	TRAP,INDEX,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE
30	OPENW	TRAP,TREE_NAME,TREE_SIZE,BEHALF
31	RDBRN	TRAP,SYSID,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF
32	RDLNK	TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE
33	WSINF	TRAP,SYSID,BUFLOC,TREE_NAME TREE_SIZE,BEHALF,DELETE
34	CATLK	TRAP,LINK_NAME,LINK_SIZE,TREE_NAME,TREE_SIZE,BEHALF
35	WTBRN	TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF
36	LOCK	TRAP,RN
37	UNLOCK	TRAP,RN
38	NOTIF	TRAP, ERN,CTRAP,STATE
39	CAUSE	TRAP,ERN,STATE,MESSAGE,NUMBER
40	DELET	TRAP,ERN,STATE
41	UNCAU	TRAP,ERN,NUMBER
42	OPSCE	TRAP,TIMLIM,MODE,MAXLEN
45	MSTA	TRAP,PLOC,LENGTH,BUFLOC
46	RDME	TRAP,BUFLOC,BUFSIZE
47	CRSG	TRAP,FRN,SEGNUM
48	WRITEM	TRAP,TRCPR,PLOC,LENGTH
49	WAMI	TRAP,CORELOC
51	RQWD	TRAP,CORELOC

Summary of Macro Calls for Control Primitives

Code	Macro Call	Description
01	SETFV TRAP,CORELOC	Setup Fault Vector
02	SETSQ TRAP,LOCTBL	Setup Squeze Mode
03	SQUEZ TRAP,REGS,IC	Enter Squeze Mode
11	RQDT No Arguments	Request Time and Date
12	RQERT No Arguments	Request Elapsed Run Time
13	SPAWN TRAP,PLOC,LENGTH	Spawn a process
14	TERM No Arguments	Terminate a process
15	PAUSE No Arguments	Block a process
16	OSEG TRAP,SEGNUM,LENGTH	Open a segment
17	CLSEG TRAP,SEGNUM	Close a segment
18	CHSEG TRAP,SEGNUM,LENGTH	Change segment length
19	EXSEG TRAP,SEG1NUM,SEG2NUM	Exchange segment numbers
45	MSTA TRAP,PLOC,LENGTH,BUFLOC	System status measurement
46	RDME TRAP,BUFLOC,BUFSIZE	Measure read me
48	WRITEM TRAP,TRCPR,PLOC,LENGTH	Write me
49	WAMI TRAP,CORELOC	Who am I
51	RQWD TRAP,CORELOC	Request working directory
47	CRSG TRAP,FRN,SEGNUM	Create Segment

Parameter Summary for Control Primitive Macro Calls

CORELOC	Location of slave fault vector	
LENGTH	SPAWN - No. of words in spawn parameter list at 'PLOC' OSEG and CHSEG - No. of words in segment	
LOCTBL	Location of squeze mapping table	
BLOC	Beginning address of block	
PLOC	Location of SPAWN parameter list	
REGS	Location of register settings to be used on entry to squezed prog.	
IC	IC and indicators on entry to squeezed program	
SEGNUM SEG1NUM SEG2NUM	Segment numbers (0, 1, 2, 3)	
TRCPR	Tracking parameters	0 (to be ignored) 1 (to start tracking this process)
TRAP	Location of trap routine	

Summary of Macro Calls for I/O Primitives

Code	Macro Call		Description
04	READ	TRAP FRN CORELOC,N,MODE	Sequential Read
05	APEND	TRAP,FRN,CORELOC,N,MODE	Sequential Write
06	RRF	TRAP,FRN,FILELOC,CORELOC,N	Random File Read
07	WRF	TRAP,FRN,FILELOC,CORELOC,N	Random File Write
08	SCR	TRAP,FRN,FILELOC	Scratch File
09	SPTR	TRAP,FRN,N	Set File Pointer
10	RQST	TRAP,RN	Request Status

Parameter Summary for I/O Primitive Macro Calls

TRAP	Location of trap routine
FRN	Reference number of a file
RN	Reference number of a file or an event
CORELOC	Starting core location in slave program for data transmission
FILELOC	Starting file location for data transmission (expressed as an element number). The first element is equal to zero.
N	Number of elements to be transmitted.
MODE	Device dependent indicator specifying mode of data transmission.

Summary of Macro Calls for File Primitives

Code	Macro Call	
20	OPEN	TRAP,TREE_NAME,TREE_SIZE,BEHALF,ELSIZE,ACCESS
21	CLOSE	TRAP,RN
22	CATLOG	TRAP,RN,TREE_NAME,TREE_SIZE,SWITCH,UACCESS,OACCESS
23	DESTRO	TRAP,TREE_NAME,TREE_SIZE,BEHALF
24	OPENS	TRAP,DEVID,MAXLEN,ELSIZE
25	UPDATE	TRAP,RN
26	CATDIR	TRAP,TREE_NAME,TREE_SIZE,BEHALF,UACCESS,OACCESS
27	WRACL	TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE
28	RDACL	TRAP,INDEX,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,NUMBER
29	RDDIR	TRAP,INDEX,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE
30	OPENW	TRAP,TREE_NAME,TREE_SIZE,BEHALF

Code	Macro Call
31	RDBRN TRAP,SYSID,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF
32	RDLNK TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,BUFSIZE
33	WSINF TRAP,SYSID,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF,DELETE
34	CATLK TRAP,LINK_NAME,LINK_SIZE,TREE_NAME,TREE_SIZE,BEHALF
35	WTBRN TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF
36	LOCK TRAP,RN
37	UNLOCK TRAP,RN

Parameter Summary for File Primitive Macro Calls

Parameter	Meaning
ACCESS	Requested access
UACCESS	Access granted to universe
OACCESS	Access granted to owner

0	31	32	33	34	3
Not Presently Used	READ	WRITE	APEND	EXEC	

BEHALF — Specifies whose ID is used in checking whether access is to be granted
  =1 Originator's ID (normal setting)
  =0 Owner's ID

BUFLOC — Location of buffer in slave program for reading and writing branch information

BUFSIZ — Number of words in buffer located at 'BUFLOC'

DELETE — Specifies action on system information block
  =0 Add or modify block
  =1 Delete block

DEVID — Specifies device on which scratch file is opened
  =1 Disc
  =2 Drum (only granted to privileged users)

ELSIZE — Element size, expressed in bits, in which subsequent data transactions for file are to be expressed. Must be a multiple of unit size.
  =0 Element size set equal to unit size

Device	Unit Size
Disc,Drum	1152 bits (32 words
Card Reader, Card Punch, Line Printer	6 bits
Teletypewriter	9 bits
Magnetic Tape	36 bits
CLM-10	36 bits
A to D converter	36 bits

Parameter Summary for File Primitive Macro Calls (Continued)

Parameter	Meaning
INDEX	The entry number at which reading begins (=1 for first entry).
LINK_NAME	The tree name which is stored in a branch of type "linl
LINK_SIZE	Number of words in LINK_NAME. Must be multiple of 6.
MAXLEN	Maximum length of scratch file, in elements
NUMBER	Number of ACL entries to be read. Maximum value is 10
RN	Reference number of file or event
SWITCH	Indicator associated with the cataloging of a file Bit 17 - ON if originator's behalf Bit 16 - OFF if access not granted to owner's files Bit 15 - OFF if file is not write inhibit when spawned

0	15	16	17
	WRINH	OAPS	BHALF

Parameter	Meaning
SYSID	Identification code for system information stored in branch =0 System information not requested =1 TSS =2 MARK 2 =3 GECOS =-J (J=1,2,3) specifies Jth block of system informatio:
TRAP	Location of trap routine
TREE_NAME	Location of tree name of entity
TREE_SIZE	Number of words in 'TREE_NAME'. Must be a multiple of

Summary of Macro Calls for Event Primitives

The following calls are applicable for files and events

Code	Macro Call	
20	OPEN	TRAP,TREE_NAME,TREE_SIZE,BEHALF,ELSIZE,ACCESS
21	CLOSE	TRAP,RN

Summary of Macro Calls for Event Primitives Continued

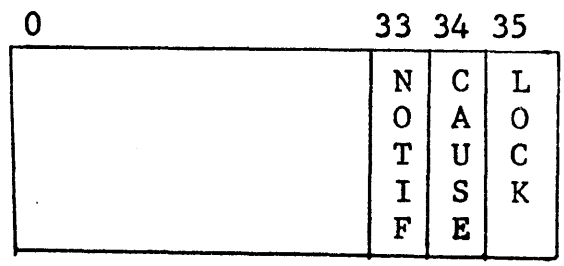
Code	Macro Call	
22	CATLOG	TRAP,RN,TREE_NAME,TREE_SIZE,SWITCH,UACCESS,OACCESS
23	DESTRO	TRAP,TREE_NAME,TREE_SIZE,BEHALF
36	LOCK	TRAP,RN
37	UNLCK	TRAP,RN

The following calls apply only to events

Code	Macro Call	
38	NOTIF	TRAP,ERN,CTRAP,STATE
39	CAUSE	TRAP,ERN,STATE,MESSAGE,NUMBER
40	DELET	TRAP,ERN,STATE
41	UNCAU	TRAP,ERN,NUMBER
42	OPSCE	TRAP,TIMLIM,MODE,MAXLEN

Parameter Summary for Event Primitive Macro Calls

Parameter	Meaning
ACCESS	Requested access
UACCESS	Access granted to universe
OACCESS	Access granted to owner



Parameter	Meaning
BEHALF	Specifies whose ID is used in checking whether access is to be granted =1 Originator's ID (normal setting) =0 Owner's ID
ELSIZE	Not specified for events
ERN	Reference number of an event
MAXLEN	Maximum number of entries which can be placed on the queue for a particular event. May be from 1 to 16.
MESSAGE	One word message to be transmitted to the notified process
MODE	Specifies event mode (Bit 17) and event type (Bit 16) Bits 16-17 00 -- Steady state mode, regular event

Parameter Summary for Event Primitive Macro Calls (Continued)

Parameter	Meaning
	10 -- Transient mode, regular event 11 -- Transient mode, event of type "pass" 10 -- Error. Pass event must be transient
NUMBER	Number of processes which are to be notified
RN	Reference number of a file or an event
STATE	Only those processes on the event queue whose state matches that specified by 'STATE' will be trapped as a result of a CAUSE. Must be set to zero for steady state events.
SWITCH	Bit 17 Behalf indicator. ON if originator's behalf (normal setting) Bits 15-16 Not specified for events
TIMLIM	Maximum time a process allowed to wait for an event to be caused. Expressed in seconds up to a maximum of 1800.
TRAP	Trap location associated with the issuance of the primitive.
TRAPC	Trap location associated with the causing of event 'ERN'.

REGISTER ASSIGNMENT FOR SYSTEM PRIMITIVES

REGISTER ASSIGNMENTS

PRIMITIVE	A	Q	X0 D	X0 ∅	X1	X2	X3	X4	X5	X6	X7
SET UP FAULT VECTOR			= 1	1	TRAP	CORELOC					
SET UP SQUEEZE MODE			= 2	2	TRAP	LOCTOL					
ENTER SQUEEZE MODE	ΣC		= 3	3	TRAP	PASS					
READ			= 4	4	TRAP	FRN		CORELOC	N	MODE	
APPEND			= 5	5	TRAP	FRN		CORELOC	N	MODE	
RANDOM READ			= 6	6	TRAP	FRN	FILELOC	CORELOC	N		
RANDOM WRITE			= 7	7	TRAP	FRN	FILELOC	CORELOC	N		
SCRATCH FILE			= 8	10	TRAP	FRN	FILELOC				
SET POINTER			= 9	11	TRAP	FRN	N				
REQUEST STATUS			= 10	12	TRAP	RN					
REQUEST DATE AND TIME			= 11	13							
REQUEST ELAPSED RUN TIME			= 12	14							
SPAWN			= 13	15	TRAP			PLOC	LENGTH		
TERMINATE			= 14	16							
PAUSE			= 15	17							
OPEN SEGMENT			= 16	20	TRAP	SEGNUM	LENGTH				
CLOSE SEGMENT			= 17	21	TRAP	SEGNUM					
CHANGE SEGMENT LENGTH			= 18	22	TRAP	SEGNUM	LENGTH				
EXCHANGE SEGMENT			= 19	23	TRAP	SEG1NUM	SEG2NUM				
OPEN		ACCESS	= 20	24	TRAP			T-NAME	T-SIZE	BHALF	ELSZE
CLOSE			= 21	25	TRAP	RN					
CATALOG	UACCESS	OACCESS	= 22	26	TRAP	RN		T-NAME	T-SIZE	SWITCH	
DESTROY			= 23	27	TRAP			T-NAME	T-SIZE	BHALF	
OPEN SCRATCH			= 24	30	TRAP			DEVID	MAXLEN		ELSIZE
UPDATE			= 25	31	TRAP	FRN					
CATALOG DIRECTORY	UACCESS	OACCESS	= 26	32	TRAP			T-NAME	T-SIZE	BHALF	
WRITE ACCESS CONTROL LIST			= 27	33	TRAP		BUFLOC	T-NAME	T-SIZE	BHALF	BUFSZE
READ ACCESS CONTROL LIST			= 28	34	TRAP	INDEX	BUFLOC	T-NAME	T-SIZE	BHALF	NUMBER
READ DIRECTORY			= 29	35	TRAP	INDEX	BUFLOC	T-NAME	T-SIZE	BHALF	BUFSZE
OPEN WORKING DIRECTORY			= 30	36	TRAP			T-NAME	T-SIZE	BHALF	
READ BRANCH			= 31	37	TRAP	SYSID	BUFLOC	T-NAME / RN	T-SIZE / 0	BHALF	
READ LINK			= 32	40	TRAP		BUFLOC	T-NAME	T-SIZE	BHALF	BUFSZE
WRITE SYSTEM INFORMATION			= 33	41	TRAP	SYSID	BUFLOC	T-NAME/FRN	T-SIZE/ 0	BHALF	DELETE
CATALOG LINK			= 34	42	TRAP	L-NAME	L-SIZE	T-NAME	T-SIZE	BHALF	
WRITE BRANCH			= 35	43	TRAP		BUFLOC	T-NAME	T-SIZE	BHALF	
LOCK			= 36	44	TRAP	RN					
UNLOCK			= 37	45	TRAP	RN					
NOTIFY	STATE		= 38	46	TRAP	RN	TRAPC				
CAUSE	STATE	MESSAGE	= 39	47	TRAP	RN	NUMBER			FRN (PASS)	ACCESS (PASS)
DELETE ENTRY	STATE		= 40	50	TRAP	RN					
UNCAUSE			= 41	51	TRAP	RN	NUMBER				
OPEN SCRATCH EVENT			= 42	52	TRAP		TIMLIM	MODE	MAXLEN		
			= 43	53							
			= 44	54							
SYSTEM STATUS MEASUREMENTS			= 45	55	TRAP	PLOC	LENGTH	BUFLOC			
MEASURE READ ME			= 46	56	TRAP	BUFLOC	BUFSZE				
CREATE SEGMENT			= 47	57	TRAP	FRN	SEGNUM				
WRITE ME			= 48	60	TRAP	TRCPR		PLOC	LENGTH		
WHO AM I			= 49	61	TRAP			CORELOC			
OUTPUT MEASUREMENTS			= 50	62	TRAP	BUFLOC	LENGTH				
REQUEST WORKING DIR			= 51	63	TRAP			CORELOC			

APPENDIX D

SUMMARY OF MACRO PROTOTYPES FOR SYSTEM PRIMITIVES

MACROS FOR CONTROL PRIMITIVES (CODE NUMBERS 1 THRU 3)

```
                         SETFV     TRAP,CORELOC

SETFV      MACRO
           REM
           LDX0     1,DU           SETUP FAULT VECTOR CODE NUMBER
           LDX1     #1             TRAP ADDRESS
           LDX2     #2             LOCATION OF SLAVE FAULT VECTOR
           MME      0
           REM
           ENDM


                         SETSQ     TRAP,LOCTBL

SETSQ      MACRO
           REM
           LDX0     2,DU           SETUP SQEEZE  MODE PRIMITIVE CODE NUMBER
           LDX1     #1             TRAP ADDRESS
           LDX2     #2             LOCATION OF SQUEZE TABLE
           MME      0
           REM
           ENDM


                         SQUEZ     TRAP,REGS,IC

SQUEZ      MACRO
           REM
           LDX0     3,DU           SQUEEZE PRIMITIVE CODE NUMBER
           LDX1     #1             TRAP ADDRESS
           LDX2     #2             LOCATION OF REGISTER SETTINGS
           LDA      #3             INSTRUCTION COUNTER INDICATORS
           MME
           REM
```

MACROS FOR I/O PRIMITIVES (CODE NUMBERS 4 THRU 10)

```
                         READ      TRAP,FRN,CORELOC,N,MODE

READ       MACRO
           REM
```

```
                      READ      TRAP,FRN,CORELOC,N,MODE

READ        LDX0      4,DU      READ PRIMITIVE CODE NUMBER
            LDX1      #1        TRAP ADDRESS
            LDX2      #2        FILE REFERENCE NUMBER
            LDX4      #3        STARTING LOCATION IN SLAVE AREA FOR READ
            LDX5      #4        NUMBER OF ELEMENTS TO BE READ
            LDX6      #5        MODE INDICATOR - DEVICE DEPENDENT
                                SPECIFYING
            MME       0         ISSUE THE READ
            REM
            ENDM


                      APEND     TRAP,FRN,CORELOC,N,MODE

END         MACRO
            LDX0      5,DU      APEND PRIMITIVE CODE # (I.E. SEQUENTIAL
                                WRIT)
            LDX1      #1        TRAP ADDRESS
            LDX2      #2        FILE REFERENCE NUMBER
            LDX4      #3        STARTING LOCATION IN SLAVE AREA FOR WRITE
            LDX5      #4        NUMBER OF ELEMENTS TO BE WRITTEN
            LDX6      #5        DEVICE DEPENDENT MODE INDICATOR FOR WRITE
            MME       0         ISSUE THE WRITE
            REM
            ENDM


                      RRF       TRAP,FRN,FILELOC,CORELOC,N

RRF         MACRO
            REM
            LDX0      6,DU      RANDOM READ CODE NUMBER
            LDX1      #1        TRAP ADDRESS
            LDX2      #2        FILE REFERENCE NUMBER
            LDX3      #3        STARTING LOC. IN FILE FOR READ
                                (ELEMENT # )
            LDX4      #4        STARTING LOC. IN SLAVE AREA FOR READ
                                (WORD)
            LDX5      #5        NUMBER OF ELEMENTS TO BE READ
            MME       0         ISSUE THE RANDOM READ
            REM
            ENDM
```

MACROS FOR I/O PRIMITIVES (CODE NUMBERS 4 THRU 10) cont'd

```
                         WRF        TRAP,FRN,FILELOC,CORELOC,N

WRF         MACRO
            REM
            LDX0         7,DU       RANDOM WRITE CODE NUMBER
            LDX1         #1         TRAP ADDRESS
            LDX2         #2         FILE REFERENCE NUMBER
            LDX3         #3         STARTING LOCATION IN FILE FOR WRITE
                                    (ELEMENT)
            LDX4         #4         STARTING LOC. IN SLAVE AREA FOR WRITE
                                    (WORD)
            LDX5         #5         NUMBER OF ELEMENTS TO BE WRITTEN
            MME          0          ISSUE THE RANDOM WRITE
            REM
            ENDM


                         SCR        TRAP,FRN,FILELOC

SCR         MACRO
            REM
            LDX0         8,DU       SCRATCH PRIMITIVE CODE NUMBER
            LDX1         #1         TRAP ADDRESS
            LDX2         #2         FILE REFERENCE NUMBER
            LDX3         #3         ELEMENT IN FILE TO SCRATCH FROM
            MME          0          ISSUE THE SCRATCH REQUEST
            REM
            ENDM


                         SPTR       TRAP,FRN,N

SPTR        MACRO
            REM
            LDX0         9,DU       SET POINTER CODE NUMBER
            LDX1         #1         TRAP ADDRESS
            LDX2         #2         FILE REFERENCE NUMBER
            LDX3         #3         # OF ELEMENTS READ PTR IS SHIFTED
            MME          0          ISSUE THE SCRATCH COMMAND
            REM
            ENDM
```

MACROS FOR I/O PRIMITIVES (CODE NUMBERS 4 THRU 10) cont'd

```
                        RQST       TRAP,RN

RQST        MACRO
            REM
            LDX0        10,DU      REQUEST STATUS CODE NUMBER.
            LDX1        #1         TRAP ADDRESS
            LDX2        #2         FILE OR EVENT REFERENCE NUMBER
            MME         0
            REM
            ENDM
```

MACROS FOR CONTROL PRIMITIVES (CODE NUMBERS 11 THRU 19)

```
                        RQDT       NO ARGUMENTS

RQDT        MACRO
            REM
            LDX0        11,DU      REQUEST DATE AND TIME PRIMITIVE CODE
                                   NUMBER
            MME         0
            REM
            ENDM

                        RQERT      NO ARGUMENTS

RQERT       MACRO
            REM
            LDX0        12,DU      REQUEST ELAPSED RUN TIME CODE NUMBER
            MME         0
            REM
            ENDM

                        SPAWN      TRAP,PLOC,LENGTH

SPAWN       MACRO
            REM
            LDX0        13,DU      SPAWN PRIMITIVE CODE NUMBER
            LDX1        #1         TRAP ADDRESS
            LDX4        #2         STARTING ADDRESS OF PARAMETER LIST FOR
                                   SPAWN
```

MACROS FOR CONTROL PRIMITIVES (CODE NUMBERS 11 THRU 19) cont'd

```
                    SPAWN     TRAP,PLOC,LENGTH

SPAWN     LDX5      #3        NUMBER OF WORDS IN ABOVE PARAMETER LIST
          LDQ       #4        ORIGINATOR (FOR SPECIAL SPAWN)
          MME       0
          REM
          ENDM
```

```
                    TERM      NO ARGUMENTS

TERM      MACRO
          REM
          LDX0      14,DU     TERMINATE PRIMITIVE CODE NUMBER
          MME       0
          REM
          ENDM
```

```
                    PAUSE     NO ARGUMENTS

PAUSE     MACRO
          REM
          LDX0      15,DU     PAUSE PRIMITIVE CODE NUMBER
          MME       0
          REM
          ENDM
```

MACROS FOR CONTROL PRIMITIVES (CODE NUMBERS 11 THRU 19)

```
                    OPSEG     TRAP,SEGNUM,LENGTH

OPSEG     MACRO
          REM
          LDX0      16,DU     OPEN SEGMENT PRIMITIVE CODE NUMBER
          LDX1      #1        TRAP ADDRESS
          LDX2      #2        SEGMENT NUMBER (SPECIFIED BY USER PROCESS)
          LDX3      #3        LENGTH OF SEGMENT TO BE OPENED
          MME       0
          REM
          ENDM
```

MACROS FOR CONTROL PRIMITIVES (CODE NUMBERS 11 THRU 19) cont'd

```
                    CLSEG     TRAP,SEGNUM

CLSEG     MACRO
          REM
          LDX0      17,DU     CLOSE SEGMENT PRIMITIVE CODE NUMBER
          LDX1      #1        TRAP ADDRESS
          LDX2      #2        SEGMENT NUMBER
          MME       0
          REM
          ENDM


                    CHSEG     TRAP,SEGNUM,LENGTH

CHSEG     MACRO
          REM
          LDX0      18,DU     CHANGE SEGMENT LENGTH PRIMITIVE CODE
                              NUMBER
          LDX1      #1        TRAP ADDRESS
          LDX2      #2        SEGMENT NUMBER
          LDX3      #3        DESIRED SEGMENT LENGTH
          MME       0
          REM
          ENDM


                    EXSEG     TRAP,SEG1NUM,SEG2NUM

EXSEG     MACRO
          REM
          LDX0      19,DU     EXCHANGE SEGMENT PRIMITIVE CODE NUMBER
          LDX1      #1        TRAP ADDRESS
          LDX2      #2        SEGMENT 1 NUMBER
          LDX3      #3        SEGMENT 2 NUMBER
          MME       0
          REM
          ENDM
```

MACROS FOR FILE PRIMITIVES (CODE NUMBERS 20 THRU 37)

```
                    OPEN      TRAP, TNAME,TSIZ,BHALF,ELSIZ,ACCESS

OPEN      MACRO
          REM
```

```
                        OPEN      TRAP,STATUS,TNAME,TSIZ,BHALF,ELSIZ,ACCESS

OPEN        LDX0        20,DU     OPEN PRIMITIVE CODE NUMBER
            LDX1        #1        TRAP ADDRESS
            LDX4        #2        LOCATION OF TREE NAME
            LDX5        #3        NUMBER OF WORDS IN TREE NAME
            LDX6        #4        BEHALF INDICATOR
            LDX7        #5        ELEMENT SIZE
            LDQ         #6        REQUESTED ACCESS
            MME         0
            REM
            ENDM


                        CLOSE     TRAP,RN

CLOSE       MACRO
            REM
            LDX0        21,DU     CLOSE PRIMITIVE CODE NUMBER
            LDX1        #1        TRAP ADDRESS
            LDX2        #2        FILE OR EVENT REFERENCE NUMBER
            MME         0
            REM
            ENDM


                        CATLOG    TRAP,RN,TNAME,TSIZE,SWITCH,UACCESS,OACCESS

CATLOG      MACRO
            REM
            LDX0        22,DU     CATLOG PRIMITIVE CODE NUMBER
            LDX1        #1        TRAP ADDRESS
            LDX2        #2        FILE OR EVENT REFERENCE NUMBER
            LDX4        #3        LOCATION OF TREE NAME
            LDX5        #4        NUMBER OF WORDS IN TREE NAME
            LDX6        #5        SWITCH INDICATOR
            LDA         #6        UNIVERSAL ACCESS
            LDQ         #7        OWNER'S ACCESS
            MME         0
            REM
            ENDM
```

MACROS FOR FILE PRIMITIVES (CODE NUMBERS 20 THRU 37)

```
                       DESTRO     TRAP,TREE-NAME,TREE-SIZE,BEHALF

DESTRO      MACRO
            REM
            LDX0       23,DU      DESTROY PRIMITIVE CODE NUMBER
            LDX1       #1         TRAP ADDRESS
            LDX4       #2         TREE NAME LOCATION
            LDX5       #3         NUMBER OF WORDS IN TREE NAME
            LDX6       #4         BEHALF
            MME        0
            REM
            ENDM


                       OPENS      TRAP,DEVID,MAXLEN,ELSIZE

OPENS       MACRO
            REM
            LDX0       24,DU      # OF THE OPEN SCRATCH COMMAND
            LDX1       #1         TRAP ADDRESS
            LDX4       #2         DEVICE ID
            LDX5       #3         MAXIMUM FILE LENGTH IN ELEMENTS
            LDX7       #4         ELEMENT SIZE IN BITS
            MME        0
            REM
            ENDM


                       UPDATE     TRAP,FRN

UPDATE      MACRO
            REM
            LDX0       25,DU      # OF UPDATE COMMAND
            LDX1       #1         TRAP ADDRESS
            LDX2       #2         FILE REFERENCE #
            MME         0
            REM
            ENDM
                       CATDIR     TRAP,TNAME,TSIZE,BHALF,UACCESS,OACCESS
CATDIR      MACRO
            REM
            LDX0       26,DU      CATLOG DIRECTORY PRIMITIVE CODE NUMBER
```

```
                    CATDIR   TRAP,TNAME,TSIZE,BHALF,UACCESS,OACCESS

CATDIR    LDX1      #1       TRAP ADDRESS
          LDX4      #2       LOCATION OF TREE NAME
          LDX5      #3       NUMBER OF WORDS IN TREE NAME
          LDX6      #4       BEHALF INDICATOR
          LDA       #5       UNIVERSAL ACCESS
          LDQ       #6       OWNER'S ACCESS
          MME       0
          REM
          ENDM

                    WRACL    TRAP,BUFLOC,TNAME,TSIZ,BHALF,BUFSIZ

WRACL     MACRO
          REM
          LDX0      27,DU    WRITE ACCESS CONTROL LIST CODE NUMBER
          LDX1      #1       TRAP LOCATION
          LDX3      #2       BUFFER LOCATION
          LDX4      #3       LOCATION OF TREE NAME
          LDX5      #4       NUMBER OF WORDS IN TREE NAME
          LDX6      #5       BEHALF INDICATOR
          LDX7      #6       BUFFER SIZE
          MME        0
          REM
          ENDM

                    RDACL    TRAP,INDEX,BUFLOC,TNAME,TSIZ,BHALF,NUMBER

RDACL     MACRO
          REM
          LDX0      28,DU    READ ACCESS CONTROL LIST CODE NUMBER
          LDX1      #1       TRAP LOCATION
          LDX2      #2       INDEX INTO ACL FOR READ TO BEGIN
          LDX3      #3       BUFFER LOCATION
          LDX4      #4       LOCATION OF TREE NAME
          LDX5      #5       NUMBER OF WORDS IN TREE NAME
          LDX6      #6       BEHALF INDICATOR
          LDX7      #7       NUMBER OF ACL ENTRIES TO BE READ
          MME       0
          REM
          ENDM
```

MACROS FOR FILE PRIMITIVES (CODE NUMBERS 20 THRU 37) cont'd

```
                        RDDIR     TRAP,INDEX,BUFLOC,TNAME,TSIZ,BHALF,BUFSIZ

RDDIR       MACRO
            REM
            LDX0        29,DU     READ DIRECTORY CODE NUMBER
            LDX1        #1        TRAP ADDRESS
            LDX2        #2        INDEX IN DIRECTORY OF FIRST BRANCH READ
            LDX3        #3        SLAVE LOC. WHERE BRANCH INFORMATION IS
                                  STORED
            LDX4        #4        LOCATION OF TREE NAME
            LDX5        #5        NUMBER OF WORDS IN TREE NAME
            LDX6        #6        BEHALF INDICATOR
            LDX7        #7        SIZE OF BUFFER WHERE BRANCH IS RETURNED
            MME         0
            REM
            ENDM

                        OPENW     TRAP,TREE-NAME,TREE-SIZE,BEHALF

OPENW       MACRO
            REM
            LDX0        30,DU     OPEN WORKING DIRECTORY CODE NUMBER
            LDX1        #1        TRAP LOCATION
            LDX4        #2        LOCATION OF TREE NAME
            LDX5        #3        NUMBER OF WORDS IN TREE NAME
            LDX6        #4        BEHALF INDICATOR
            MME         0
            REM
            ENDM

                        RDBR      TRAP,SYSID,BUFLOC,TNAME,TSIZE,BEHALF

RDBRN       MACRO
            REM
            LDX0        31,DU     READ BRANCH PRIMITIVE CODE NUMBER
            LDX1        #1        TRAP ADDRESS
            LDX2        #2        SYSTEM ID
            LDX3        #3        STARTING LOCATION OF BUFFER
            LDX4        #4        LOCATION OF TREE NAME
            LDX5        #5        NUMBER OF WORDS IN TREE NAME
            LDX6        #6        BEHALF INDICATOR
            MME         0
            REM
            ENDM
```

```
                      RDLNK     TRAP,BUFLOC,TREENAME,TREESIZE,BHALF,BUFSZE

RDLNK       MACRO
            REM
            LDX0      32,DU     READ LINK PRIMITIVE  CODE NUMBER
            LDX1      #1        TRAP ADDRESS
            LDX3      #2        STARTING LOCATION OF BUFFER
            LDX4      #3        LOCATION OF TREE NAME
            LDX5      #4        NUMBER OF WORDS IN TREE NAME
            LDX6      #5        BEHALF INDICATOR
            LDX7      #6        SIZE OF BUFFER
            MME       0
            REM
            ENDM


                      WSINF     TRAP,SYSID,BUFLOC,TNAME,TSIZ,BHALF,DELETE

WSINF       MACRO
            REM
            LDX0      33,DU     WRITE SYSTEM INFORMATION CODE NUMBER
            LDX1      #1        TRAP LOCATION
            LDX2      #2        SYSTEM ID
            LDX3      #3        LOCATION OF FOUR WORD INF BUFFER
            LDX4      #4        LOCATION OF TREE NAME
            LDX5      #5        NUMBER OF WORDS IN TREE NAME
            LDX6      #6        BEHALF INDICATOR
            LDX7      #7        DELETE SWITCH (=1 DELETE)
            MME       0
            REM
            ENDM


                      CATLK     TRAP,LNAME,LSIZ,TNAME,TSIZ,BHALF

CATLK       MACRO
            REM
            LDX0      34,DU     CATLOG LINK CODE NUMBER
            LDX1      #1        TRAP LOCATION
            LDX2      #2        LOCATION OF NAME TO BE CATALOGUED IN LINK
            LDX3      #3        NUMBER OF WORDS IN ABOVE NAME
            LDX4      #4        LOCATION OF TREE NAME OF DIRECTORY
            LDX5      #5        NUMBER OF WORDS IN TREE NAME FOR -X4-
            LDX6      #6        BEHALF INDICATOR
            MME       0
            REM
            ENDM
```

MACROS FOR FILE PRIMITIVES (CODE NUMBERS 20 THRU 37) cont'd

```
                        WTBRN     TRAP,BUFLOC,TREE_NAME,TREE_SIZE,BEHALF

WTBRN        MACRO
             REM
             LDX0      35,DU     WRITE BRANCH CODE NUMBER
             LDX1      #1        TRAP LOCATION
             LDX3      #2        STARTING LOCATION OF BUFFER
             LDX4      #3        TREE NAME LOCATION
             LDX5      #4        # WORDS IN TREE NAME
             LDX6      #5        BEHALF INDICATOR
             MME        0
             REM
             ENDM
                       LOCK      TRAP,RN
LOCK         MACRO
             REM
             LDX0      36,DU     LOCK PRIMITIVE CODE NUMBER
             LDX1      #1        TRAP ADDRESS
             LDX2      #2        FILE OR EVENT REFERENCE NUMBER
             MME       0
             REM
             ENDM


                       UNLCK     TRAP,RN

UNLCK        MACRO
             REM
             LDX0      37,DU     UNLOCK PRIMITIVE CODE NUMBER
             LDX1      #1        TRAP ADDRESS
             LDX2      #2        FILE OR EVENT REFERENCE NUMBER
             MME       0
             REM
             ENDM
```

MACROS FOR EVENT PRIMITIVES (CODE NUMBERS 38 THRU 42)

```
                       NOTIF     TRAP,ERN,TRAPC,STATE

NOTIF        MACRO
             REM
             LDX0      38,DU     NOTIFY EVENT PRIMITIVE CODE NUMBER
             LDX1      #1        TRAP LOCATION FOR COMPLETION OF NOTIFY
             LDX2      #2        EVENT REFERENCE NUMBER
             LDX3      #3        TRAPC LOC (CNTRL RETURNED WHEN EVNT CAUSED)
             LDA       #4        STATE
             MME       0
             REM
             ENDM
```

```
                CAUSE     TRAP,ERN,NUMBER,STATE,MESSAGE,RN,ACCESS

CAUSE     MACRO
          REM
          LDX0    39,DU     CAUSE PRIMITIVE CODE NUMBER
          LDX1    #1        TRAP ADDRESS
          LDX2    #2        EVENT REFERENCE NUMBER
          LDX3    #3        NUMBER OF PROCESSES TO BE TRAPPED
          LDX6    #6        REFERENCE # FOR PASS EVENT
          LDX7    #7        ACCESS FOR PASS EVENT
          LDA     #4        STATE
          LDQ     #5        MESSAGE
          MME     0
          REM
          ENDM


                DELET     TRAP,ERN,STATE

DELET     MACRO
          REM
          LDX0    40,DU     DELETE EVENT PRIMITIVE CODE NUMBER
          LDX1    #1        TRAP ADDRESS
          LDX2    #2        EVENT REFERENCE NUMBER
          LDA     #3        STATE
          MME     0
          REM
          ENDM


                UNCAU     TRAP,ERN,NUMBER

UNCAU     MACRO
          REM
          LDX0    41,DU     UNCAUSE EVENT CODE NUMBER
          LDX1    #1        TRAP ADDRESS
          LDX2    #2        EVENT REFERENCE NUMBER
          LDX3    #3        NUMBER BY WHICH CURRENT COUNT DECREASED
          MME     0
          REM
          ENDM
```

MACROS FOR CONTROL PRIMITIVES (CODE NUMBERS 45 THRU 51)

```
                        OPSCE       TRAP,TIMLIM,MODE,MAXLEN

OPSCE       MACRO
            REM
            LDX0        42,DU       OPEN SCRATCH EVENT CODE NUMBER
            LDX1        #1          TRAP ADDRESS
            LDX3        #2          TIME LIMIT
            LDX4        #3          MODE (STEADY-S-00, TRANS-01, PASS EVNT-11)
            LDX5        #4          MAXIMUM QUEUE LENGTH
            MME         0
            REM
            ENDM


                        MSTA        TRAP,PLOC,LENGTH,BUFLOC

MSTA        MACRO
            REM
            LDX0        45,DU       SYSTEM STATUS MEASUREMENTS CODE NUMBER
            LDX1        #1          TRAP ADDRESS
            LDX2        #2          ADDRESS OF ARGUMENT LIST
            LDX3        #3          # ARGUMENTS
            LDX4        #4          BUFFER ADDRESS
            MME         0
            REM
            ENDM


                        RDME        TRAP,BUFLOC,BUFSIZE

RDME        MACRO
            REM
            LDX0        46,DU       MEASURE READ ME CODE NUMBER
            LDX1        #1          TRAP ADDRESS
            LDX2        #2          BUFFER LOCATION
            LDX3        #3          BUFFER SIZE
            MME         0
            REM
            ENDM
```

```
                    CRSG      TRAP,FRN,SEGNUM

CRSG       MACRO
           REM
           LDX0     47,DU     CREATE SEGMENT CODE NUMBER
           LDX1     #1        TRAP ADDRESS
           LDX2     #2        FILE REFERENCE #
           LDX3     #3        SEGMENT #
           MME      0
           REM
           ENDM


                    WRITEM    TRAP,TRCPR,PLOC,LENGTH

WRITEM     MACRO
           REM
           LDX0     #48,DU    WRITE ME CODE NUMBER
           LDX1     #1        TRAP ADDRESS
           LDX2     #2        TRACKING PARAMETER
           LDX4     #3        ADDRESS OF ARGUMENT LIST
           LDX5     #4        # ARGUMENTS
           MME      0
           REM
           ENDM
                    WAMI      TRAP,CORELOC
WAMI       MACRO
           REM
           LDX0     49,DU     WHO AM I CODE NUMBER
           LDX1     #1        TRAP ADDRESS
           LDX4     #2        CORE LOCATION
           MME      0
           REM
           ENDM
                    RQWD      TRAP,CORELOC
RQWD       MACRO
           REM
           LDX0     51,DU     REQUEST WORKING DIRECTORY CODE NUMBER
           LDX1     #1        TRAP ADDRESS
           LDX4     #2        CORE LOCATION
           MME      0
           REM
           ENDM
```

INDEX

APPENDIX II

Listing of the Monitor

PAGE 219 THERE WERE NO WARNING FLAGS IN THE ABOVE ASSEMBLY

000000 PARITY ERRORS.

000240 PAGES.   009610 RECORDS.   048012 MILISEC PROCESSOR

ASSEMBLY CONTROL CARDS

```
                          1 *$*     DISK     ASMSETUP                                            110
                          2         TTLS     ASSEMBLY CONTROL CARDS                              100
                          3         TTL      INPUT/OUTPUT SCHEDULER -- IOS MONITOR               110
                          4         HEAD                                                         120
                          5 *                                                                    130
                          6 *                                                                    140
                          7 *                                  HOUSEKEEPING CARDS                150
                          8 *                                                                    160
                          9 *                                                                    170
                         10         PCC      ON                PRINT CONTROL CARDS               180
                         11         PMC      ON                EXPAND MACROS                     190
                         12         DETAIL   ON                EXPAND OCT,DEC,BCI,ASCII,DUP,ETC. 200
                         13         DETAIL   OFF                                                 210
                         14         PCC      OFF                                                 220
        000000           20         ORG      0                 SET ORIGIN                        280
                         21         HEAD                       AND UNHEADED                      290
                         22         LBL      IOS,IOS           I-NPUT/O-UTPUT S-CHEDULER         300
                         23 *                                                                    310
                         24 *                                                                    320
                         25 *                                                                    330
                         26 *       SQUASH ASSEMBLER BUG                                         340
                         27 *                                                                    350
 001000 400013           28 MME     OPD      012/0010,6/,02/2,6/,6/,02/2,02/3                    360
                         29 *                                                                    370
                         30 *                                                                    380
                         31 *                                                                    390
                         32 *                DEBUGGING AIDS                                      400
                         33 *                                                                    410
                         34 *       IF THE DEBUG SWITCH IS SET ON, THEN THE DEBUG MACROS         420
                         35 *       WILL BE EXPANDED, OTHERWISE NOT.                             430
                         36 *                                                                    440
        000000           37 OFF     EQU      0                 OFF SWITCH                        450
        000001           38 ON      EQU      1                 ON SWITCH                         460
                         39 *                                                                    470
        000001           40 DBG     SET      ON                INITIALLY ON                      480
                         41 *                                                                    490
                         42 *                                                                    500
                         44 *                                                                    520
                         45 TTL     OPSYN    TTLS              FOR SYSPROG PPRNT                 530
```

ASSEMBLY DECK SETUP

```
        HEAD
 *
 *
 *                                          ASSEMBLY DECK SETUP
 *
 *
 *      GMAP ASSEMBLY DECK SETUP -- THE DECK SETUP FOR ASSEMBLING THE
 *      THIS PROGRAM (HEREAFTER IOS) IS AS FOLLOWS.  ALL CONTROL
 *      CARDS BEGIN WITH A '$'.  THE FOLLOWING DECK WILL PRODUCE
 *      A NEW K* TAPE, A P* TAPE FOR THE LISTING AND IS ALSO USED
 *      TO PRODUCE A SYMBOL TABLE FOR DDT.  THE SECOND ACTIVITY,
 *      SYSPROG PPRNT, CREATES THE SYMBOL TABLE AND STORES THE
 *      RESULTS IN THE FILE 'RUBENS/IOS-SYM-TAB.  TO DO AN ASSEMBLY
 *      SIMPLY SUBMIT THE FOLLOWING PROGRAM TO TECOS:
 *              SOURCE RUBENS/<NAME-OF-DECK>$RUN$EXIT
 *      ALTER CARDS SHOULD BE SAVED PREVIOUSLY IN FILE 'RUBENS/ALTERS'
 *
 *
 *$     SNUMB   MBR
 *$     IDENT   RUBENS,MICHAEL
 *$     GMAP    COMDK,DECK
 *$     TAPE    G*,X1D,,TAPE#A,,IOSA
 *$     TAPE    *1,X2R
 *$     TAPE    K*,X3D,,,,IOSB
 *$     TAPE    P*,X4S,,,,IOSPRINT
 *$     ASCII   A*,BCDT,RUBENS/ALTERS
 *$     SYSPROG PPRNT
 *$     TAPE    IN,X4D,,,,IOSPRINT
 *$     PRMFL   A1,S,R,RUBENS/IOS-SYM-TAB-A
 *$     COMMENT PLEASE PRINT P* 'IOSPRINT' FOR MBR.  THANX.
 *$     ENDJOB
 *$EOD
```

BINDER DECK SETUP

```
        HEAD
*
*
*                                   BINDER DECK SETUP
*
*
*       THE BINDER DECK SETUP -- THE DECK SETUP FOR LOADING THE
*       ASSEMBLED PROGRAM INTO THE R & DC SYSTEM IS AS FOLLOWS.  NOTE
*       THAT THE DECK IS LOADED VIA 'MULTIBINDER' AND NOT THE USUAL
*       'BINDER'.  THIS IS NECESSARY IF BOTH MULTI-SEGMENTS AND USE
*       COUNTERS ARE TO BE USED.
*
        DCARD   3,$
$BUILD SPAWNSYS,RUBENS,MULTIBINDER
$       FNAME   */XIOSA
$       OBJECT
```

LOCATION COUNTERS

```
                HEAD
         *
         *
         *                                  LOCATION COUNTERS
         *
         *
         *      THE FOLLOWING PREDEFINES THE ORDER IN WHICH THE LOCATION
         *      COUNTERS WILL OCCUR, INDEPENDENT OF THE ORDER IN WHICH
         *      THEY ARE USED WITHIN THE PROGRAM.
         *
000000          USE     CODE                MAIN PROGRAM SEGMENT
000000   ZCODE  BSS     0
         *
         *
004620          USE     CONST               STORAGE FOR CONSTANTS, TABLES
004620   ZCONS  BSS     0
         *
         *
005140          USE     QSTOR               FOR ALL QUEUES
005140   ZQSTR  BSS     0
         *
         *
005300          USE     STORE               FOR ALL DYNAMIC STORAGE
005300   ZSTOR  BSS     0
         *
         *
000000          USE     CODE                CODE LOCATION COUNTER INITIALLY
         *$*    DISK    MMEDEFS
```

```
                  HEAD
           *
           *
           *                                  EXECUTIVE CONSTANTS
           *
           *
           *      THE DEFINITIONS OF SYSTEM PARAMETERS SUCH AS MME NUMBERS
           *      ARE INDICATED BY SYMBOLS WHICH START WITH A DECIMAL POINT.
           *      THEY ARE NOT HEADED SO A TYPICAL REFERENCE IS LDX X0,$.OPEN,DU.
           *
           *              DEFINITION OF MME NUMBERS
           *
000000     .PRIV  EQU    0                    MICRO-CODED PRIVILEGED PRIMITIVE
000001     .SETFV EQU    1                    SET UP FAULT VECTOR
000002     .SETSQ EQU    2                    SET UP SQUEEZE MODE
000003     .SQUEZ EQU    3                    ENTER SQUEEZE MODE
000004     .READ  EQU    4                    READ
000005     .APEND EQU    5                    APPEND
000006     .RRF   EQU    6                    READ RANDOM FILE
000007     .WRF   EQU    7                    WRITE RANDOM FILE
000010     .SCR   EQU    8                    SCRATCH
000011     .SPTR  EQU    9                    SET POINTER
000012     .RQST  EQU    10                   REQUEST STATUS
000013     .RQDT  EQU    11                   REQUEST DATE AND TIME
000014     .RQERT EQU    12                   REQUEST ELAPSED RUN TIME
000015     .SPAWN EQU    13                   SPAWN
000016     .TERM  EQU    14                   TERMINATE
000017     .PAUSE EQU    15                   PAUSE
000020     .OPSEG EQU    16                   OPEN SEGMENT
000021     .CLSEG EQU    17                   CLOSE SEGMENT
000022     .CHSEG EQU    18                   CHANGE SEGMENT LENGTH
000023     .EXSEG EQU    19                   EXCHANGE TWO SEGMENTS
000024     .OPEN  EQU    20                   OPEN
000025     .CLOSE EQU    21                   CLOSE
000026     .CATLG EQU    22                   CATALOGUE
000027     .DESTR EQU    23                   DESTROY
000030     .OPENS EQU    24                   OPEN SCRATCH
000031     .UPDAT EQU    25                   UPDATE
000032     .CATDR EQU    26                   CATALOGUE DIRECTORY
000033     .WRACL EQU    27                   WRITE ACCESS CONTROL LIST
000034     .RDACL EQU    28                   READ ACCESS CONTROL LIST
000035     .RDDIR EQU    29                   READ DIRECTORY
000036     .OPENW EQU    30                   OPEN WORKING DIRECTORY
000037     .RDBRN EQU    31                   READ BRANCH
000040     .RDLK  EQU    32                   READ LINK
000041     .WSINF EQU    33                   WRITE SYSTEM INFORMATION
000042     .CATLK EQU    34                   CATALOGUE LINK
000043     .WBRAN EQU    35                   WRITE BRANCH
000044     .LOCK  EQU    36                   LOCK
000045     .UNLCK EQU    37                   UNLOCK
```

DEFINITIONS OF EXECUTIVE SYSTEM CONSTANTS

```
000046      176 .NOTIF EQU     38              NOTIFY                                                   610
000047      177 .CAUSE EQU     39              CAUSE                                                    620
000050      178 .DELET EQU     40              DELETE EVENT                                             630
000051      179 .UNCAU EQU     41              UNCAUSE EVENT                                            640
000052      180 .OPSCE EQU     42              OPEN SCRATCH EVENT                                       650
000055      181 .MSTA  EQU     45              SYSTEM STATUS MEASUREMENTS                               660
000056      182 .RDME  EQU     46              MEASURE READ ME                                          670
000057      183 .CRSG  EQU     47              CREATE SEGMENT                                           680
000060      184 .WTME  EQU     48              WRITE ME                                                 690
000061      185 .WAMI  EQU     49              WHO AM I                                                 700
000063      186 .RQWD  EQU     51              REQUEST WORKING DIRECTORY                                710
            187 *$*    DISK    XBITS                                                                    720
```

BIT AND STATUS DEFINITIONS

```
                 HEAD    B
          *
          *
          *                         BIT DEFINITIONS
          *
          *
          *
          *              DEFINITIONS FOR REPEAT INSTRUCTIONS
          *
002000    RPT    BOOL    002000          COUNT FIELD FOR RPT INSTRUCTIONS
001000    ABIT   BOOL    001000          INCREMENT FIRST INDEX REGISTER
000400    BBIT   BOOL    000400          INCREMENT SECOND INDEX REGISTER
000200    CBIT   BOOL    000200          LOAD C(XO) FROM REPEAT INSTRUCTION
000100    TZE    BOOL    000100          ZERO
000040    TNZ    BOOL    000040          NON ZERO
000020    TMI    BOOL    000020          NEGATIVE
000010    TPL    BOOL    000010          POSITIVE
000004    TRC    BOOL    000004          CARRY
000002    TNC    BOOL    000002          NO CARRY
000001    TOV    BOOL    000001          OVERFLOW
          *
          *
          *              DEFINITIONS FOR INDICATOR REGISTER
          *
400000    ZER    BOOL    400000          ZERO INDICATOR BIT
200000    NEG    BOOL    200000          NEGATIVE
100000    CAR    BOOL    100000          CARRY
040000    OVF    BOOL    040000          OVERFLOW
020000    EOV    BOOL    020000          EXPONENT OVERFLOW
010000    EUN    BOOL    010000          EXPONENT UNDERFLOW
004000    OVM    BOOL    004000          OVERFLOW MASK -- ON PREVENTS OVERFLOW FAULTS
002000    TAL    BOOL    002000          TALLY RUNOUT
001000    PAR    BOOL    001000          PARITY ERROR
000400    PAM    BOOL    000400          PARITY ERROR MASK
000200    MOD    BOOL    000200          MASTER MODE
          *
          *
          *              DEFINITION FOR ACCESS BITS
          *
000020    RD     BOOL    20              READ ACCESS
000010    WT     EQU     RD/2            WRITE
000004    AP     EQU     WT/2            APPEND
000002    EX     EQU     AP/2            EXECUTE
000001    LK     EQU     EX/2            LOCK
000037    ALL    EQU     RD+WT+AP+EX+LK  ALL
000004    NO     BOOL    4               NOTIFY
000002    CA     EQU     NO/2            CAUSE
          *
```

B BIT AND STATUS DEFINITIONS

```
              *
              *
              *              DEFINITION FOR SPAWN BITS
              *
000010        CEND   BOOL    10                  CORE END SEGMENT
000004        BSTRD  BOOL    4                   PROESS TO KNOW OF ITSELF--BASTARD BIT
400000        WP     BOOL    400000              WRITE PROTECTED
000000        VOID   BOOL    0                   VOID SEGMENT
000001        PS     BOOL    1                   SPAWN FROM PARENT'S SEGMENT
000002        PF     BOOL    2                   SPAWN FROM PARENT'S FILE
              *
              *
              *              DEFINITIONS FOR MODE BITS AND PASS INDICATORS
              *
000000        SS     BOOL    0                   STEADY STATE
000001        TRANS  BOOL    1                   TRANSIENT
000000        NPASS  BOOL    0                   NOT OF TYPE PASS
000002        PASS   BOOL    2                   PASS EVENT
```

```
              *
              *
              *                 DEFINITIONS FOR EXEC STATUS RETURNS
              *
              *                 LOGICAL STATUS CODE FOR I/O PRIMITIVES
              *
000000        OK       BOOL     00               OK
000001        IFRN     BOOL     01               INVALID FRN
000002        IACC     BOOL     02               INVALID ACCESS SPECIFIED
000003        BZ       BOOL     03               EXECUTIVE TOO BUSY
000004        IOP      BOOL     04               INVALID OPERATION FOR THIS DEVICE
000005        IPTR     BOOL     5                COPY POINTER IS OUT OF BOUNDS
000006        IREQ     BOOL     6                AMOUNT REQUESTED GREATER THAN FILE LENGTH
000007        IELT     BOOL     07               ELEMENT SIZE IS NOT A MULTIPLE OF UNIT SIZE
000011        IMOD     BOOL     11               INVALID MODE
000012        HDWE     BOOL     12               HARDWARE ERROR -- OPERATION NOT COMPLETE
000013        DBZ      BOOL     13               DEVICE UNAVAILABLE (HARDWARE)
000016        EOF      BOOL     16               END-OF-FILE ENCOUNTERED
000026        NSTR     BOOL     26               NO FILE STORAGE AVAILABLE
000035        TRO      BOOL     35               TIMER RUN OUT ON OPERATOR'S CONSOLE
              *
              *
              *                 LOGICAL STAUS CODE FOR FILE AND EVENT PRIMITIVES
              *
000005        ITN      BOOL     5                INVALID NAME
000007        UERR     BOOL     7                UNRECOVERABLE ERROR
000011        TLE      BOOL     11               TIME LIMIT EXCEEDED
000013        LOCK     BOOL     13               ITEMED LOCKED
000016        IPWD     BOOL     16               INVALID PASSWORD
              *
              *
              *                 LOGICAL STATUS CODES FOR CONTROL PRIMITIVES
              *
              *
000004        IO       BOOL     4                I/O ACTIVITY IN PROGRESS
              *
              *
              *                 LOGICAL STATUS CODED FOR SPAWN AND CREATE SEGMENT
              *
000011        RNA      BOOL     11               RESOURCES NOT AVAILABLE (BOO-HISS)
              *
              *
              *                 MISCELLANEOUS BITS
              *
000077        STMK     BOOL     77               STATUS BIT MASK
400000        SIGN     BOOL     400000           SIGN BIT
400000        TERM     BOOL     400000           TERMINATOR BIT
200000        DELIM    EQU      TERM/2           DELIMINATOR
100000        DIGIT    EQU      DELIM/2          DIGIT
040000        OPR      EQU      DIGIT/2          OPERATOR
```

```
B                          STATUS DEFINITIONS

              *
              *
              *              DEFINITIONS FOR JSFLAG
              *
              *                                      J$FLAGS (UPPER)
              *
001700        MODMK  BOOL    1700                    MEDIA MASK FOR RCW (BITS 26 - 29)
740000        JOBMK  BOOL    740000                  JOB NUMBER MASK (TOP 4 BITS)
000017        BJBMK  BOOL    17                      4 BIT MASK
020000        HDRMK  BOOL    020000                  HEADER MASK (ON = OUTPUT BANNER)
000001        BHDR   BOOL    1                       HEADER BIT MASK
010000        OUTMK  BOOL    010000                  OUTPUT TYPE MASK (OFF = 512; ON = 320)
000001        BOTMK  BOOL    1                       OUTPUT BIT MASK
007777        SRTMK  BOOL    007777                  START ADDRESS IN ELEMENTS OF J$BUFSZ
007777        BSTMK  BOOL    7777                    BIT MASK
              *
              *
              *              COMMANDS RETURNED TO MONITOR
              *
002000        GET    BOOL    002000                  PERIPHERAL GOTTEN (NOT SENT)
004000        KILL   BOOL    004000                  PERIPHERAL KILLED
006000        REL    BOOL    006000                  PERIPHERAL RELEASED
010000        XXXX   BOOL    010000                  NOT USED
012000        RSTRT  BOOL    012000                  PERIPHERAL RESTARTED
014000        DONE   BOOL    014000                  JOB SUCCESSFULLY FINISHED
016000        RDY    BOOL    016000                  READY PERIPHERAL
              *
              *
              *              COMMANDS SENT TO SUB-MODULES
              *
000000        .ABRT  BOOL    000000                  ABORT PERIPHERAL  XXXX
002000        .GET   BOOL    002000                  GET PERIPHERAL  XXXX
004000        .KILL  BOOL    004000                  KILL PERIPHERAL  XXXX
006000        .REL   BOOL    006000                  RELEASE PERIPHERAL  XXXX
010000        .XXXX  BOOL    010000                  UNDEFINED
012000        .STRT  BOOL    012000                  RESTART PERIPHERAL  XXXX
              *$*    DISK    ASCII
```

B ASCII CHARACTFRS

```
                HEAD    A
       *
       *
       *                                ASCII CHARACTERS
       *
000000  NUL     BOOL    0               NULL
600004  EOT     BOOL    4+B$DELIM+B$TERM •EOT
000007  BELL    BOOL    7               BELL
000012  LF      BOOL    12              LINE FEED
000015  CR      BOOL    15              CARRIAGE RETURN
200040  SP      BOOL    40+B$DELIM      SPACE
000040  BLANK   BOOL    40              BLANK
600043  TERM    BOOL    43+B$DELIM+B$TERM  ASCII #
200044  DOL     BOOL    44+B$DELIM      DOLLAR SIGN
600046  AMPER   BOOL    46+B$DELIM+B$TERM  ASCII AMPERSAND
000052  AST     BOOL    52              ASTERISK
200054  COMMA   BOOL    54+B$DELIM      COMMA
000056  DP      BOOL    56              DECIMAL POINT
240057  SLASH   BOOL    57+B$DELIM+B$OPR •SLASH
100060  Do      BOOL    60+B$DIGIT      DIGIT ZERO
000072  COL     BOOL    72              COLON
200073  SCOL    BOOL    73+B$DELIM      SEMI-COLON
000077  QM      BOOL    77              QUESTION MARK
000101  LA      BOOL    101             LETTER A
000177  DEL     BOOL    177             DELETE
       *
       *
000177  MASK    BOOL    177             ASCII CHARACTER MASK
```

A SPECIAL ASCII TABLE

```
                                  HEAD   A
                           *
                           *
                           *                            SPECIAL ASCII TABLE
                           *
                           *
                 004620           USE    CONST
                 004620     ASCTB BSS    0
004620 000000200040               VFD    036/A$SP
004621 000000000041               VFD    036/41           !
004622 000000000042               VFD    036/42           "
004623 000000600043               VFD    036/A$TERM       #
004624 000000200044               VFD    036/A$DOL        $
004625 000000000045               VFD    036/45           %
004626 000000600046               VFD    036/A$AMPER      &
004627 000000000047               VFD    036/47           '
004630 000000000050               VFD    036/50           (
004631 000000000051               VFD    036/51           )
004632 000000000052               VFD    036/52           *
004633 000000040053               VFD    036/53+B$OPR     +
004634 000000200054               VFD    036/A$COMMA      ,
004635 000000040055               VFD    036/55+B$OPR     -
004636 000000000056               VFD    036/56           .
004637 000000240057               VFD    036/A$SLASH      /
004640 000000100060               VFD    036/60+B$DIGIT   0
004641 000000100061               VFD    036/61+B$DIGIT   1
004642 000000100062               VFD    036/62+B$DIGIT   2
004643 000000100063               VFD    036/63+B$DIGIT   3
004644 000000100064               VFD    036/64+B$DIGIT   4
004645 000000100065               VFD    036/65+B$DIGIT   5
END OF BINARY CARD IOS00003
004646 000000100066               VFD    036/66+B$DIGIT   6
004647 000000100067               VFD    036/67+B$DIGIT   7
004650 000000100070               VFD    036/70+B$DIGIT   8
004651 000000100071               VFD    036/71+B$DIGIT   9
004652 000000000072               VFD    036/72           :
004653 000000200073               VFD    036/A$SCOL       ;
004654 000000040074               VFD    036/74+B$OPR     <
004655 000000000075               VFD    036/75           =
004656 000000040076               VFD    036/76+B$OPR     >
004657 000000000077               VFD    036/77           ?
                 000040     ASCLN EQU    *-ASCTB          TABLE LENGTH
                 000000           USE    PREVIOUS
                            *$*   DISK   ACCOUNT
```

```
                HEAD    A
       *
       *
       *        ACB
       *
       *        THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
       *        IN THE ACCOUNTING CONTROL BLOCK (ACB).
       *
000000 FAC      EQU     0               FACILITY USED FOR ACCESSED
000001 TIMES    EQU     FAC+1           TEME OF START
000002 DATES    EQU     TIMES+1         DATE OF START
000003 ID       EQU     DATES+1         USER ID
000007 SHOP     EQU     ID+4            SHOP ORDER
000012 UNUSED   SET     SHOP+3          NOT USED
000013 DISK     EQU     UNUSED+1        DISK UNITS
000014 DRUM     EQU     DISK+1          DRUM UNITS
000015 OPCON    EQU     DRUM+1          OPERATOR'S CONSOLE
000016 CP       EQU     OPCON+1         CARD PUNCH
000017 LP       EQU     CP+1            LINE PRINTER
000020 CDRD     EQU     LP+1            CARD READER
000021 UNUSED   SET     CDRD+1          NOT USED
000022 CLM      EQU     UNUSED+1        CLM 10
000023 MT       EQU     CLM+1           MAGNETIC TAPE
000024 AD       EQU     MT+1            A/D CONVERTER
000025 UNUSED   SET     AD+1            NOT USED
000026 UNUSED   SET     UNUSED+1        NOT USED
000027 MIP      EQU     UNUSED+1        MEMORY INTERFACE PROCESSOR
000030 UNUSED   SET     MIP+1           NOT USED
000031 UNUSED   SET     UNUSED+1        NOT USED
000032 CPU      EQU     UNUSED+1        CPU TIME
000033 CORE     EQU     CPU+1           CORE PRODUCT
000035 RL       EQU     CORE+2          RL NUMBER (TTY)
00037  TIMEO    EQU     RL+2            TIME OFF
       *$*      DISK    TCB
```

A TRAP BLOCK DESCRIPTION

```
                 HEAD    T
          *
          *
          *                            TCB
          *
          *
          *      THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
          *      IN THE TRAP BLOCK (TBLOCK).
          *
000000    SRW1   EQU     0             FIRST STATUS RETURN WORD FROM EXEC
000001    SRW2   EQU     1             SECOND STATUS RETURN WORD
000002    RET    EQU     2             SAVED IC/IR WHEN EXEC SPRINGS TRAP
000003    XED    EQU     3             CONTROL IS TRANSFERRED HERE WHEN EXEC
          *                            SPRINGS THE TRAP.  IT CONTAINS AN XED
          *                            OF A CHAIN WHICH LINKS THE TRAP TO THE
          *                            MASTER TASK QUEUE.
000004    TRA    EQU     4             (UPPER) RESTART ADDRESS FOR TASKS ON
          *                            ON A QUEUE (SUCH AS THE Q$TASK)
          *                            (LOWER) MAY BE USED TO SAVE RETURN
          *                            FROM A REENTRANT ROUTINE
000005    LINK   EQU     5             (UPPER) LINK TO PREVIOUS TCB
000006    NCB    EQU     6             (UPPER) POINTER TO NCB
000006    JCB    EQU     NCB           (LOWER) POINTER TO JCB
000007    SPARE  EQU     7             SPARE
000030    LEN    EQU     24            LENGTH OF TCB (NICE IF MULTIPLE OF 8)
          *
          *
000027    TEMP1  EQU     LEN-1         TEMPORARY STORAGE AT END OF BLOCK
000026    TEMP2  EQU     TEMP1-1       MORE TEMPORARY STORAGE
000025    TEMP3  EQU     TEMP2-1
000024    TEMP4  EQU     TEMP3-1
000023    TEMP5  EQU     TEMP4-1
000022    TEMP6  EQU     TEMP5-1
000021    TEMP7  EQU     TEMP6-1
000020    TEMP8  EQU     TEMP7-1
          *
          *      NO ONE EXCEPT R$GETC SHOULD USE TEMP9 - TEM16
          *
000017    TEM9   EQU     TEMP8-1
000016    TEM10  EQU     TEM9-1
000015    TEM11  EQU     TEM10-1
000014    TEM12  EQU     TEM11-1
000013    TEM13  EQU     TEM12-1
000012    TEM14  EQU     TEM13-1
000011    TEM15  EQU     TEM14-1
000010    TEM16  EQU     TEM15-1
          *$*    DISK    JCB
```

T JOB CONTROL BLOCK DESCRIPTION

```
               HEAD    J
        *
        *
        *                              JCB
        *
        *
        *
        *      THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
        *      IN THE JOB CONTROL BLOCK (HERAFTER CALL JCB).
        *
000000  QFRN   EQU     0               (UPPER) FRN OF ASSOCIATED INPUT FILE
000001  QFLOC  EQU     QFRN+1          R/W PTR POSITION OF INPUT FILE IN *QFRN*
000002  FRN    EQU     QFLOC+1         FRN OF FILE TO BE PROCESSED
000003  TYPE   EQU     FRN+1           (UPPER) TYPE
000003  DISP   EQU     TYPE            (LOWER) DISPOSITION
000004  ACODE  EQU     DISP+1          ACODE FOR ACCOUNTING
000005  NCB    EQU     ACODE+1         (UPPER) PTR TO NCB
000005  TCB    EQU     NCB             (LOWER) PTR TO TCB
000006  STATI  EQU     NCB+1           INITIATE STATE FOR COMMUNICATIONS
000007  JOB    EQU     STATI+1         JOB NUMBER
000007  STATT  EQU     JOB             TERMINATE STATE FOR COMMUNICATIONS
000010  MESS   EQU     STATT+1         MESSAGE FOR COMMUNICATIONS
000011  BUF    EQU     MESS+1          WORKING BUFFER
000012  SIZE   EQU     BUF+1           AMOUNT OF DATA TO BE PROCESSED
000013  RES    EQU     SIZE+1          START OF RESOURCE REQUIREMENT LIST
000026  TT     SET     RES+3+1-QFRN+7  ROUND TO MULTIPLE OF 8
000020  TT     SET     TT/8*8          ROUND
000020  LEN    EQU     TT              LENGTH OF JCB (MINIMUM LENGTH = 16. )
        *$*    DISK    NCB
```

```
                HEAD    C
       *
       *
       *                                    NCB
       *
       *        THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
       *        IN THE NOTIFY BLOCK (ALIAS NCB).
       *
       *                A NCB IS A TCB WITH EXTRAS.
       *
000000   SRW1   EQU     T$SRW1
000001   SRW2   EQU     T$SRW2
000002   RET    EQU     T$RET
000003   XED    EQU     T$XED
000004   TRA    EQU     T$TRA
000005   LINK   EQU     T$LINK
000005   RLINK  EQU     LINK            (LOWER) RESTART AFTER NOTIFY
000006   NCB    EQU     T$NCB
000006   JCB    EQU     T$JCB
000007   ABBR   EQU     T$SPARE         ASCII ABREVIATION OF NCB
000027   TEMP1  EQU     T$TEMP1
000026   TEMP2  EQU     T$TEMP2
000025   TEMP3  EQU     T$TEMP3
000024   TEMP4  EQU     T$TEMP4
000023   TEMP5  EQU     T$TEMP5
000022   TEMP6  EQU     T$TEMP6
000021   TEMP7  EQU     T$TEMP7
000020   TEMP8  EQU     T$TEMP8
000017   TEM9   EQU     T$TEM9
000016   TEM10  EQU     T$TEM10
000015   TEM11  EQU     T$TEM11
000014   TEM12  EQU     T$TEM12
000013   TEM13  EQU     T$TEM13
000012   TEM14  EQU     T$TEM14
000011   TEM15  EQU     T$TEM15
000010   TEM16  EQU     T$TEM16
000024   ERN    EQU     TEMP4           ERN FOR NOTIFY ***BENE NOTA***
000025   STATE  EQU     TEMP3           STATE FOR NOTIFY ***BENE NOTA***
000030   QFRN   EQU     T$LEN           (UPPER) INPUT Q FILE
000031   QFLOC  EQU     QFRN+1          R/W PTR FOR 'QFRN'
000032   BUSY   EQU     QFLOC+1         NO. OF FILES CURRENTLY ACTIVE FROM THIS
                                        INPUT QUEUE FILE
000033   RUN    EQU     BUSY+1          PTR TO R$OMAX FOR THE TYPE REOURCES
                                        NEEDED BY THIS JOB.  IF R$OMAX = 0, THEN WE
                                        SHOULD IGNORE HIM FOR NOW (SAVVY?)
000034   QUEUE  EQU     RUN+1           PTR TO WAIT Q-LIST TO PLACE JOB ON
000035   RES    EQU     QUEUE+1         RESOURCE LIST  (MUST BE LAST)
                                        THIS WAY WE TEST FIRST TO SEE IF WE
                                        HAVE ANY PERIHERAL AT ALL
```

C NOTIFY BLOCK DESCRIPTION

```
*
*
*       THIS MACRO IS USED TO GENERATE THE NECESSARY NOTIFY CONTROL
*       BLOCKS FOR THE COMMUNICATIONS NETWORK.
*
*
*
NCB     MACRO   LABEL,ABBR,RESTART-ADD,ERN,STATE,R$OMAX,QLIST,RES,NO,RES,NO,...
        USE     STORE
        EIGHT
#1      BSS     0                   LABEL
        ZERO    0,B$TRO             SRW1: SIMULATE A TIMER RUNOUT
        ZERO    0,0                 SRW2
        ZERO    0,0                 RET
        ZERO    0,0                 XED
        ZERO    **,0                TRA
        ZERO    0,#3                LINK/ RESTART
        ZERO    *-C$NCB,*-C$JCB     'NCB/ JCB POINT TO THEMSELVES
        UASCI   1,#2                ASCII ABREVIATION
        DUP     1,16-4              TEM16 THRU TEMP5
        DEC     0
FR#1    ARG     #4                  ERN
        ZERO    #5,0                STATE
        DUP     1,2                 TEMP2 & TEMP1
        DEC     0
        ARG     -1                  QFRN
        DEC     0                   QFLOC
        DEC     0                   BUSY
        ARG     #6                  RUN PTR TO R$OMAX OF TYPE NEEDED
        ARG     #7                  Q-LIST PTR
        INE     '#8',''
        ZERO    #8,#9               RESOURCE TYPE/NUMBER NEEDED
        INE     '#10',''
        ZERO    #10,#11             DITTO
        INE     '#12',''
        ZERO    #12,#13
        DEC     -1                  MARK END OF RESOURCE REQUIREMENT LIST
        USE     PREVIOUS
        ENDM    NCB
*$*     DISK    INPUT
```

C INPUT DESCRIPTION FORMATS

```
               HEAD
        *
        *
        *      THE INPUT FILES TO BE PROCESSED ARE NAMED IN A FILE
        *      'PRINT-FILE-QUEUE', OR 'PUNCH-FILE-QUEUE'. BOTH
        *      ARE CATALOGED IN THE DIRECTORY 'SYSOUT'.  EACH FILE
        *      WHICH REPRESENTS A JOB IS NAMED IN A 64. WORD RECORD.
        *
        *      THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
        *      IN THE RECORD.
        *
000000  CKSM   EQU    0                CHECKSUM OF OTHER 63. WORDS
000001  TNSZ   EQU    1                (UPPER) NUMBER OF WORDS IN TREE-NAME
000001  BANR   EQU    1                (LOWER) NON-ZERO MEANS NO BANNER
000002  TYPE   EQU    2                (UPPER) TYPE OF FILE
                                       0 = 512 WORD EDITED BLOCK
                                       1 = 320 GECOS FORMAT
000002  DISP   EQU    TYPE             (LOWER) DISPOSITION OF FILE
                                       0 = DESTROY, SCRATCH, & CLOSE
                                       1 =          SCRATCH, & CLOSE
                                       2 =                     CLOSE
000003  ACODE  EQU    3                ACODE FOR BILLING
000004  TN     EQU    4                START OF TREE-NAME
        *
        *
000001  TYPMK  EQU    1                TYPE MASK
000003  DISMK  EQU    3                DISPOSITION MASK
        *
        *
000100  QBFSZ  EQU    64               INPUT BUFFER SIZE
004400  QELSZ  EQU    36*QBFSZ         ELEMENT SIZE = ONE RECORD
        *$*    DISK   QCB
```

QUEUE MANAGEMENT DEFINITIONS

```
              HEAD    Q
        *
        *
        *                               QCB
        *
        *
        *     THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
        *     IN A QBLOCK GENERATED BY THE QUEUE MACRO.
        *
000000  FIRST EQU     0                 POINTER TO FIRST BLOCK OF QUEUE
000001  LAST  EQU     FIRST+1           POINTER TO LAST BLOCK OF QUEUE
000002  XADD  EQU     LAST+1            INSTRUCTION PAIR FOR ADDINGA BLOCK
000004  XENQ  EQU     XADD+2            INSTRUCTION PAIR FOR ENQUEUEING
000006  XDEQ  EQU     XENQ+2            INSTRUCTION PAIR FOR DEQUEUEING
000010  XINV  EQU     XDEQ+2            INSTRUCTION PAIR FOR INVERTING
000012  BUSY  EQU     XINV+2            RESPONSIBLE BLOCK IF QUEUE IS BUSY
                                        ZERO OTHERWISE
000013  MAX   EQU     BUSY+1            MAXIMUM NUMBER OF ITEMS ASSOCIATED WITH Q
000014  AVAIL EQU     MAX+1             NUMBER OF ITEMS CURRENTLY AVAILABLE
000015  SPAR1 EQU     AVAIL+1           SPAR1
000016  SPAR2 EQU     SPAR1+1           SPAR2
000017  ABBR  EQU     SPAR2+1           ASCII ABBREVIATION OF QUEUE
000020  LEN   EQU     ABBR+1            LENGTH OF QUEUE (WISE TO KEEP EVEN)
        *
        *
000004  OFFST EQU     4                 OFFSET FOR QUEUE POINTER
000003  LINK  EQU     OFFST-1           FORWARD LINK POINTER
        *$*   DISK    CCB
```

Q CORE MANAGEMENT DEFINITIONS

```
               HEAD    R
        *
        *
        *                          CCB
        *
        *
        *      THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS IN
        *      A BLOCK ON THE FREE MEMORY LIST.
        *
000000   LINKF  EQU     0            POINTER TO SUCCESSOR (UPPER)
000000   LEN    EQU     LINKF        TOTAL LENGTH (IN WORDS) OF BLOCK (LOWER)
000001   LINKB  EQU     LINKF+1      POINTER TO PREDECCESSOR
        *$*     DISK    HEAD
```

R GLOBAL DEFINITIONS

```
                 HEAD
               *
               *
               *                         GLOBALS
               *
               *         HEAD SYMBOL USAGE
               *
                 HEAD                    GLOBAL AND EXEC CONSTANTS
                 HEAD    B               GENERAL PURPOSE BITS
                 HEAD    C               COMMUNICATIONS ROUTINES
                 HEAD    J               JCB SYMBOLS AND ROUTINES
                 HEAD    O               OPERATOR INTERFACE AND LOGGING
                 HEAD    Q               QUEUE SYMBOLS AND ROUTINES
                 HEAD    R               RESOURCE ALLOCATION
                 HEAD    S               STATISTICS COUNTERS
                 HEAD    T               TRAP SYMBOLS AND ROUTINES
                 HEAD    X               DIAGNOSTIC ROUTINES
                 HEAD
               *
               *
               *         INDEX REGISTER DEFINITIONS
               *
               * THE SYMBOLIC INDEX REGISTERS USED IN THIS PROGRAM ARE
               * ONE CHARACTER SYMBOLS, DEFINED UNDER EACH HEAD SYMBOL
               * IN USE IN THE PROGRAM.  INDEX REGISTER 0 IS SPECIAL,
               * SINCE IT IS USED FOR REPEAT INSTRUCTIONS, SO IT IS NOT
               * SYMBOLIC.
               *
                 HEAD    0,C,J,O,Q,R,T,X
               *
000001         T EQU     1               TRAP BLOCK POINTER
000002         X EQU     2               TEMP
000003         Y EQU     3               TEMP
000004         Z EQU     4               TEMP
000005         Q EQU     5               QUEUES AND GENERAL USE
000006         J EQU     6               JOB NUMBER
000007         L EQU     7               LINK REGISTER FOR SUBROUTINE CALLS
               *
               *
               *         OTHER GLOBAL SYMBOLS
               *
                 HEAD
777777     ERROR EQU     -1              USED TO GENERATE MEMORY FAULTS
525200       BUG BOOL    525200          BUGGING QUANTITY
525200    BUGBUG SET     BUG             MR. G. M. I. A. BUGGER
000777      CKMK BOOL    777             STATUS MASK (9 BITS)
000040     TALYB BOOL    40              TALLYB BIT
777700     TALMK BOOL    777700          MASK FOR TALLY COUNT FIELD
000100       TAL BOOL    100             TALLY DISPLACEMENT
000005     RTMAX EQU     5               RETRY ERROR ONLY 5 TIMES
```

GLOBAL CONSTANTS

```
                                750        HEAD
                                751 *
                                752 *
                                753 *                                 GLOBALS
                                754 *
                                755 *
                                756 *                  DUMPFILE PARAMETERS
                                757 *
                      004660    758        USE     CONST
   004660  122125102105         759 DTN    UASCI   6,RUBENS
   004666  111117123104         760        UASCI   6,IOSDUMP
 END OF BINARY CARD IOS00004
                      000014    761 DTSZ   EQU     *-DTN              TREE-SIZE
                      044000    762 DESZ   EQU     36*512             ELEMENT SIZE (ONE PAGE OF CORE)
                      000037    763 DACC   BOOL    B$ALL
                      000000    764        USE     PREVIOUS
                                765 *
                                766 *
                                767 *                  MEMORY MANAGEMENT PARAMETERS
                                768 *
                      000000    769 BUFSEG EQU     0                  SEGMENT WHERE DYNAMIC BUFFER IS LOCATED
                      001000    770 MQUAN  EQU     512                QUANTUM FOR MEMROY REQUEST
                      002000    771 1K     EQU     1024               ONE K DECIMAL
                      004001    772 RQMAX  EQU     2*1K+1             MAX MEMORY REQUEST/SHOT
                                773 *
                      005300    774        USE     STORE
   005300  000400 000000        775 AVAIL  ZERO    ZZ1,0              MEMORY AVAILABLE
   005301  001000 000000        776 MEMRQ  ZERO    MQUAN,0            MEMORY REQUIREMENT/ NEED ONE UNIT ALWAYS
   005302  006350 0000 00       777 MTOPO  ARG     ZTOPO              END OF PROGRAM
   005303  007000 0000 00       778 MTOP   ARG     ZTOP               TOP OF MEMORY
                      000000    779        USE     PREVIOUS
                                780 *$*    DISK    FAULT
```

LOW CORE ALLOCATION -- FAULT VECTOR

```
                     000000   782          USE    CODE                                      110
                              783          HEAD   X                                         120
                              784 *                                                         130
                              785 *                                                         140
                              786 *                              FAULT VECTOR               150
                              787 *                                                         160
                              788 *        CONSIDER ALL FAULTS FATAL                        170
                              789 *                                                         180
                     000000   790          ORG    0              IN CASE OF PRECEEDING ERRORS, FORCE ZERO 190
                     000000   791 FV       BSS    0              FAULT VECTOR               200
    000000  004106 7100 00    792          TRA    $UP            0 = SHUTDOWN               210
    000001  000370 7170 00    793          XED    FAULT                                     220
    000002  000000 000000     794 MEM      ZERO                  1 = MEMORY                 230
    000003  000370 7170 00    795          XED    FAULT                                     240
    000004  000000 000000     796 MME      ZERO                  2 = MASTER MODE ENTRY      250
    000005  000370 7170 00    797          XED    FAULT                                     260
    000006  000000 000000     798 FT       ZERO                  3 = FAULT TAG              270
    000007  000370 7170 00    799          XED    FAULT                                     280
    000010  000000 000000     800 TIMER    ZERO                  4 = TIMER RUNOUT           290
    000011  000370 7170 00    801          XED    FAULT                                     300
    000012  000000 000000     802 COMND    ZERO                  5 = COMMAND                310
    000013  000370 7170 00    803          XED    FAULT                                     320
    000014  000000 000000     804 DRL      ZERO                  6 = DERAIL                 330
    000015  000370 7170 00    805          XED    FAULT                                     340
    000016  000000 000000     806 LOCK     ZERO                  7 = LOCKUP                 350
 END OF BINARY CARD IOS00005
    000017  000370 7170 00    807          XED    FAULT                                     360
    000020  000000 000000     808 CONCT    ZERO                  8 = CONNECT                370
    000021  000370 7170 00    809          XED    FAULT                                     380
    000022  000000 000000     810 PARTY    ZERO                  9 = PARITY                 390
    000023  000370 7170 00    811          XED    FAULT                                     400
    000024  000000 000000     812 OP       ZERO                  10 = ILLEGAL OP CODE       410
    000025  000370 7170 00    813          XED    FAULT                                     420
    000026  000000 000000     814 ONC      ZERO                  11 = OPERATION NOT COMPLETE 430
    000027  000370 7170 00    815          XED    FAULT                                     440
    000030  000000 000000     816 STRT     ZERO                  12 = STARTUP               450
    000031  000370 7170 00    817          XED    FAULT                                     460
    000032  000000 000000     818 OFLOW    ZERO                  13 = OVERFLOW              470
    000033  000370 7170 00    819          XED    FAULT                                     480
    000034  000000 000000     820 DIV      ZERO                  14 = DIVIDE CHECK          490
    000035  000370 7170 00    821          XED    FAULT                                     500
    000036  000000 000000     822 XEC      ZERO                  15 = EXECUTE               510
    000037  000370 7170 00    823          XED    FAULT                                     520
```

```
                                 *
                                 *
                                 *                             DEBUG STORAGE
                                 *
                                 *      STORAGE FOR REGISTERS AND IC ON CRASH
                                 *
                       000040           EIGHT
                       000040    REGS   BSS     8              STORAGE FOR CRASH REGISTERS
  000050  000000 000000          IC1    ZERO                   IC BEFORE FAULT
  000051  000000 000000          IC     ZERO                   IC AT FAULT
                                 *
                                 *      DATE AND TIME OF CRASH
                                 *
                       000052    DATE   BSS     1              DATE OF CRASH
                       000053    TIME   BSS     1              TIME OF CRASH
  000054  001101070701           DATEA  DATE                   ASSEMBLY DATE
END OF BINARY CARD IOS00006
  000055  000000 0000 00         SBAR   ARG     0              BAR SETTING WHEN CRASHED
                                 *
                                 *      PATCH AREA
                                 *
                       000056           EVEN
                       000056    PATCH  BSS     64             LEAVE LOTS OF ROOM
                                 *
                                 *      STORAGE FOR DEBUGGING QUEUE
                                 *
                       000160           EIGHT                  QUASH STUPID ASSEMBLER BUG.
  000160  000170 0000 00         REG    ARG     DBGQ           POINTER TO NEXT ENTRY
                       000020    DBGQN  EQU     16             NUMBER OF ENTRIES
                       000170           EIGHT
                       000170    DBGQ   BSS     8*DBGQN        RESERVE SPACE
                                 *$*    DISK    DIAG
```

```
                  000370          USE     CODE
                                  HEAD    X
                            *
                            *
                            *                               DIAGNOSTICS
                            *
                            *     THIS SECTION IS ENTERED FROM THE FAULT VECTOR IN THE
                            *     EVENT OF A PROGRAMMING ERROR.  THEN REGISTERS ARE
                            *     PRESERVED.  THE ENTIRE PROGRAM IS WRITTEN OUT INTO
                            *     THE FILE RUBENS/LPCPDUMP.
                            *
                            *     CONSIDER ALL FAULTS FATAL.
                            *
                            *     ENTER FROM THE FAULT VECTOR BY  XED X$FAULT
                            *
                  000370          EVEN
                  000370    FAULT BSS     0               ENTRY POINT
 000370  000051 5540 00           STC1    IC              SAVE IC AND IR
 000371  000372 7100 00           TRA     *+1             BREAK XED
 000372  000040 7530 00           SREG    REGS            SAVE REGISTERS
 000373  000051 2200 00           LDX     0,IC            FIND IC+1 AT FAULT
 000374  777776 2350 10           LDA     -2,0            GET SAVED IC
 000375  000050 7550 00           STA     IC1             FOR SPECIAL LOCATION
 000376  000013 2200 03           LDX     0,$.RQDT,DU     REQUEST DATE AND TIME
 000377  000000 0010 00           MME
 000400  000052 7570 00           STAQ    DATE            SAVE DATE AND TIME
                            *
                            *     OPEN DUMP FILE
                            *
                  000401    FT1   BSS     0               OPEN THE DUMP FILE
 000401  000024 2200 03           LDX     0,$.OPEN,DU     MME NUMBER
 000402  000414 2210 03           LDX     1,TRAP1,DU      TRAP
 000403  004660 2240 03           LDX     4,$DTN,DU       TREE-NAME
 000404  000014 2250 03           LDX     5,$DTSZ,DU      TREE-SIZE
 000405  000001 2260 03           LDX     6,1,DU          BEHALF
 000406  044000 2270 03           LDX     7,$DESZ,DU      ELEMENT-SIZE
END OF BINARY CARD IOS00007
 000407  000037 2360 07           LDQ     $DACC,DL        ACCESSES
 000410  000000 0010 00           MME                     TRY OPENING
                            *
                            *     PAUSE TILL OPENED
                            *
 000411  000017 2200 03           LDX     0,$.PAUSE,DU    PAUSE INDEFINITELY
 000412  000000 0010 00           MME                     WAIT FOR TRAP
 000413  000411 7100 00           TRA     *-2             KEEP WAITING
                  000414    TRAP1 BSS     3               TRAP ON DUMPFILE OPEN
 000417  000414 2220 00           LDX     2,TRAP1         GET FRN
 000420  000425 6010 00           TNZ     FT2             DID WE REALLY GET IT OPEN?
 000421  000414 7200 00           LXL     0,TRAP1         NO, SEE WHY NOT
 000422  000003 1000 03           CMPX    0,B$BZ,DU       WAS THE EXEC TOO BUSY?
```

```
000423  000401 6000 00          907         TZE     FT1              YES. SO RETRY                          600
000424  000470 7100 00          908         TRA     TERM             NOPE. WELL THAT'S IT                   610
                                909 *                                                                   620
                                910 *       SCRATCH DUMP FILE                                           630
                                911 *                                                                   640
                      000425    912 FT2     BSS     0                                                   650
000425  000010 2200 03          913         LDX     0.$.SCR.DU       MME NUMBER                             660
000426  000435 2210 03          914         LDX     1.TRAP2.DU       TRAP ADDRESS                           670
000427  000414 2220 00          915         LDX     2.TRAP1          FRN OF FILE                            680
000430  000000 2230 03          916         LDX     3.0.DU           SCRATCH IT                             690
000431  000000 0010 00          917         MME                                                         700
                                918 *                                                                   710
                                919 *       PAUSE TILL SCRATCHED                                        720
                                920 *                                                                   730
000432  000017 2200 03          921         LDX     0.$.PAUSE.DU     ALWAYS PAUSE                           740
000433  000000 0010 00          922         MME                      WAIT                                   750
000434  000432 7100 00          923         TRA     *-2              FOREVER                                760
                      000435    924 TRAP2   BSS     3                                                   770
000440  000435 7200 00          925         LXL     0.TRAP2          CHECK EXEC STATUS RETURN               780
END OF BINARY CARD IOS00008
000441  000445 6000 00          926         TZE     FT3              OK. CONTINUE                           790
000442  000003 1000 03          927         CMPX    0.B$BZ.DU        NO. WELL WAS THE EXEC TOO BUSY?        800
000443  000425 6000 00          928         TZE     FT2              YES. JUST RETRY                        810
000444  000470 7100 00          929         TRA     TERM             NOPE. JUST BLEWIT                      820
                                930 *                                                                   830
                                931 *       START CORE TO FILE DUMP                                     840
                                932 *                                                                   850
                      000445    933 FT3     BSS     0                                                   860
000445  000007 2200 03          934         LDX     0.$.WRF.DU       WRITE RANDON FILE                      870
000446  000461 2210 03          935         LDX     1.TRAP3.DU       TRAP ADDRESS                           880
000447  000414 2220 00          936         LDX     2.TRAP1          FRN OF DUMPFILE                        890
000450  000000 2230 03          937         LDX     3.0.DU           TO: FILE LOCATION                      900
000451  000000 2240 03          938         LDX     4.0.DU           FROM: CORE LOCATION                    910
000452  000055 5500 00          939         SBAR    SBAR             GET CORE-SIZE                          920
000453  000055 2250 00          940         LDX     5.SBAR           LOAD NUMBER OF ELEMENTS                930
000454  000377 3650 03          941         ANX5    =0377.DU         MASK TO NUMBER OF ELEMENTS             940
000455  000000 0010 00          942         MME                      INITIATE THE COPY                      950
                                943 *                                                                   960
                                944 *       PAUSE TILL DUMP IS DONE                                     970
                                945 *                                                                   980
000456  000017 2200 03          946         LDX     0.$.PAUSE.DU     ALWAYS PAUSE                           990
000457  000000 0010 00          947         MME                      WAIT                                   1000
000460  000456 7100 00          948         TRA     *-2              FOREVER                                1010
                      000461    949 TRAP3   BSS     3                COPY TO DUMP FILE                      1020
000464  000461 7200 00          950         LXL     0.TRAP3          GET STATUS                             1030
000465  000470 6000 00          951         TZE     TERM             NOW IT IS TIME TO TERMINATE            1040
000466  000003 1000 03          952         CMPX    0.B$BZ.DU        WAS THE EXEC TOO BUSY                  1050
000467  000445 6000 00          953         TZE     FT3              YES. SO JUST RETRY                     1060
                      000470    954 TERM    BSS     0                TIME TO SAY SO LONG                    1070
000470  000016 2200 03          955         LDX     0.$.TERM.DU      TERMINATE                              1080
```

```
          X                      DIAGNOSTICS

 END OF BINARY CARD IOS00009
     000471  000000 0010 00       956         MME                    BYE-BYE                      1090
     000472  000470 7100 00       957         TRA      *-2           TAKE NO CHANCES              1100
                                  958 *$*     DISK     EXITM                                      1110
```

```
X                             EXIT MACRO

            000473                 USE      CODE
                                   HEAD
                              *
                              *
                              *                           EXIT
                              *
                              *    EXIT TERMINATES A THREAD OF CONTROL BY RETURNING TO THE
                              *    TASK DISTRIBUTOR.
                              *
                              EXIT MACRO                  NO ARGUMENTS
                                   TRA      $EXIT
                                   ENDM     EXIT
                              *$*  DISK     BUGM
```

BUGGING MACROS

```
000473          USE      CODE
                HEAD
          *
          *
          *     BUGGING MACROS PLANT ADDRESSES IN INVALID DATA AREAS SO THAT
          *     ANY UNAUTHORIZED USE OF SUCH DATA WILL RESULT IN A MEMORY FAULT
          *     OR EXECUTIVE CALL REJECT.
          *
          *
          *                                BUG
          *
          *     BUG FILLS BOTH UPPER AND LOWER HALVES OF A STORAGE WORD WITH
          *     THE BUG PATTERN $BUGBUG.
          *
        BUG     MACRO    STORAGE-ADDRESS
                IFE      $DBG,$ON,4
        BUGBUG  SET      BUGBUG+1
                LDX      0,BUGBUG,DU
                STX      0,#1
                SXL      0,#1
                ENDM     BUG
          *
          *
          *                                BUGU
          *
          *     BUGU FILLS THE UPPER HALF OF A STORAGE WORD WITH THE BUG
          *     PATTERN $BUGBUG
          *
        BUGU    MACRO    STORAGE-ADDRESS
                IFE      $DBG,$ON,3
        BUGBUG  SET      BUGBUG+1
                LDX      0,BUGBUG,DU
                STX      0,#1
                ENDM     BUGU
          *
          *
          *                                BUGL
          *
          *     BUGL FILL THE LOWER HALF OF A STORAGE WORD WITH THE BUG
          *     PATTERN $BUGBUG.
          *
        BUGL    MACRO    STORAGE-ADDRESS
                IFE      $DBG,$ON,3
        BUGBUG  SET      BUGBUG+1
                LDX      0,BUGBUG,DU
                SXL      0,#1
                ENDM     BUGL
```

BUGGING MACROS

```
*
*
*                                 BUGXR
*
*       BUGXR LOADS THE SPECIFIED INDEX REGISTER(S) WITH THE
*
BUGXR   MACRO   INDEX-REGISTER(S)
        IFE     $DBG,$ON,4
BUGBUG  SET     BUGBUG+1
        IDRP    #1
        LDX     #1,BUGBUG,DU
        IDRP
        ENDM    BUGXR
*
*
*                                 BUGA
*
*       BUGA LOADS THE CONTENTS OF THE THE A REGISTER WITH THE BUG
*       PATTERN $BUGBUG.
*
BUGA    MACRO   <NO ARGUMENTS>
        IFE     $DBG,$ON,3
BUGBUG  SET     BUGBUG+1
        LDA     BUGBUG,DU
        ORA     BUGBUG,DL
        ENDM    BUGA
*
*
*                                 BUGQ
*
*       BUGQ LOADS THE CONTENTS OF THE Q REGISTER WITH THE BUG
*       PATTERN $BUGBUG.
*
BUGQ    MACRO   <NO ARGUMENTS>
        IFE     $DBG,$ON,3
BUGBUG  SET     BUGBUG+1
        LDQ     BUGBUG,DU
        ORQ     BUGBUG,DL
        ENDM    BUGQ
*
*
*                                 DECRM MACRO
*
*       DECREMENT A COUNTER
*
DECRM   MACRO   COUNTER-ADDRESS
        LCQ     1,DL
        ASQ     #1
        ENDM    DECRM
*$*     DISK    CKPTM
```

CHECKPOINT MACRO

```
                      000473         USE     CODE
                                     HEAD    X
                              *
                              *
                              *                                CHECKPOINTS
                              *
                              *      THIS MARCRO CAUSES THE REGISTERS TO BE STORED IN 8-WORD
                              *      BLOCKS IN A CIRCULAR QUEUE FOR DEBUGGING USE.  INFORMATION
                              *      IS STORED IN THE FOLLOWING FORMAT:
                              *
                              *
                              *      C(0) =  C(X0) (UPPER)
                              *              C(X1) (LOWER)
                              *      C(1) =  C(X2) (UPPER)
                              *              C(X3) (LOWER)
                              *      C(2) =  C(X4) (UPPER)
                              *              C(X5) (LOWER)
                              *      C(3) =  C(X6) (UPPER)
                              *              C(X7) (LOWER)
                              *      C(4) =  C(A)
                              *      C(5) =  C(Q)
                              *      C(6) =  C(E) (0-7 BITS)
                              *      C(7) =  C(TR) (0-23 BITS)
                              *
                              *
                              *                                CKPT
                              *
                              CKPT   MACRO   <NO ARGUMENTS>
                                     IFE     $DBG,$ON,1
                                     XED     X$CKPT
                                     ENDM    CKPT
                              *
                              *
                              *              CKPT -- SUBROUTINE
                              *
                      000474         EVEN
                      000474  CKPT   BSS     0                 ENTRY POINT
000474  005304 5540 00               STC1    CKIC              SAVE IC
000475  000476 7100 00               TRA     *+1               BREAK XED
000476  005310 7530 00               SREG    CKREG             SAVE REGISTERS FOR A RELOAD
000477  000160 7530 51               SREG    REG,I             SAVE IN 8-WORD BLOCK
                              *
                              *      UPDATE POINTER FOR CIRCULAR QUEUE
                              *
000500  000160 2200 00               LDX     0,REG             GET CURRENT ADDRESS
000501  000010 0200 03               ADLX    0,8,DU            BUMP TO NEXT ENTRY
000502  000370 1000 03               CMPX    0,DBGQ+8*DBGQN,DU  OVER THE END?
000503  000505 6020 00               TNC     *+2               NO, THIS IS VALID
000504  000170 2200 03               LDX     0,DBGQ,DU         YES, RESET TO BEGINNING
000505  000160 7400 00               STX     0,REG             SAVE FOR NEXT TIME
```

```
        X                      CHECKPOINT MACRO

000506  005310 0730 00    1123          LREG   CKREG       RESTORE REGISTERS              610
000507  005304 6300 00    1124          RET    CKIC        RESTORE IC                     620
                          1125 *                                                          630
                          1126 *                                                          640
               005304     1127          USE    STORE                                      650
               005304     1128 CKIC     BSS    1           TEMP STORAGE FOR IC/IR         660
               005310     1129          EIGHT                                             670
               005310     1130 CKREG    BSS    8           TEMP STORAGE FOR REGISTERS     680
               000510     1131          USE    PREVIOUS                                   690
                          1132 *$*      DISK   SETUPM                                     700
```

X TRAP SETUP MACRO

```
                  000510   1134         USE     CODE
                           1135         HEAD
                           1136 *
                           1137 *
                           1138 *                               SETUP MACRO
                           1139 *
                           1140 SETUP   MACRO                   NO ARGUMENTS
                           1141         XED     $SETUP
                           1142         ENDM    SETUP
                           1143 *
                           1144 *
                           1145 *               SETUP -- SUBROUTINE TO SET UP A TRAP
                           1146 *
                           1147 *       CALL WITH
                           1148 *               C(XT) = TBLOCK-ADDRESS
                           1149 *               C(XJ) = JBLOCK ADDRESS
                           1150 *               C(X0) = TRANSFER ADDRESS FOR J$TRA
                           1151 *       ENTER BY
                           1152 *               XED T$SETUP
                           1153 *       DESTROYS C(A), C(Q), C(X0)
                           1154 *       USES NO TEMPORARIES
                           1155 *
                           1156 *
                  000510   1157         EVEN
                  000510   1158 SETUP   BSS     0
000510  000004 7400 11     1159         STX     0,T$TRA,T       SET T$TRA = RESTART ADDRESS
000511  000512 7000 00     1160         TSX     0,*+1           BREAK XED
000512  000000 4310 03     1161         FLD     0,DU            ZERO OUT A AND Q
END OF BINARY CARD IOS00010
000513  000000 7570 11     1162         STAQ    T$SRW1,T        ZERO STATUS WORDS
000514  000520 2370 00     1163         LDAQ    TRAP-1          GET ZERO, XED WORDS
000515  000002 7570 11     1164         STAQ    T$XED-1,T       SAVE ZERO, XED
000516  000000 7100 10     1165         TRA     0,0             RETURN
                           1166 *
                           1167 *
                           1168 *               TRAP -- XED SEQUENCE TO PUT BLOCK ON Q$TASK
                           1169 *
                  000520   1170         EVEN
000520  000000 000000      1171         ZERO                    CAN BE USED FOR CLEARING RET WORDS
000521  000522 7170 00     1172 TRAP    XED     *+1             THIS IS EXECUTED FROM THE TBLOCK
000522  005161 5540 54     1173         STC1    Q$LAST+Q$TASK,DI *UPDATE PREVIOUS LAST POINTER
000523  000524 7170 00     1174         XED     *+1             CONTINUE WITHOUT AFFECTING IC
000524  005161 5540 00     1175         STC1    Q$LAST+Q$TASK   UPDATE POINTER TO LAST
000525  777777 6300 04     1176         RET     -1,IC           RETURN TO POINT OF INTERRUPTION
                           1177 *$*     DISK    SYSCALLS
```

SYSTEM CALL MACRO DESCRIPTIONS

```
000526               USE     CODE
                     HEAD
           *
           *
           *
           *         ALL SYSTEM CALLS ARE DONE THRU PRE-DEFINED MACROS.  THOSE
           *         MACROS ARE LISTED IN THE FOLLOWING PAGES.  EACH OF THESE
           *         MACROS IS CODED TO SPECIFIC CONVENTIONS:
           *         ENTER BY
           *                 TSX  0,$<MACRO-NAME>
           *         ENTERED WITH
           *                 C(XT) = TBLOCK-ADDRESS
           *                 C(XJ) = JBLOCK-ADDRESS
           *         CALLS
           *                 SETUP
           *         ISSUES MME AND THEN 'EXITS'
           *         RETURNS TO FIRST LOCATION AFTER MACRO
           *         RETURNS WITH
           *                 C(XT) = TBLOCK-ADDRESS
           *                 C(XJ) = JBLOCK-ADDRESS
           *                 C(XL) = RESTART-ADDRESS
           *
           *         IN ACTUALITY, THE FOLLOWING HAPPENS:  THE MACRO DOES
           *         DO THE 'TSX0'.  HOWEVER, FOLLOWING THAT INSTRUCTION IS
           *         THE SET OF ARGUMENTS FOR THE MME.  THE SUBROUTINE ENTERED
           *         WHICH ALWAYS HAS THE SAME NAME AS THE MACRO KNOWS THE
           *         NUMBER OF ARGUMENTS IT IS PASSED.  THEREFORE IT SIMPLY CAL-
           *         CULATES THE RESTART ADDRESS.  IT THEN CALLS 'SETUP' WHICH
           *         RESETS THE TRAP BLOCK (C(XT) = X1) WITH THE FIRST THREE WORDS
           *         BEING ZEROED AND THE LINK WORD FOR THE EXEC SET TO THE
           *         XED INSTRUCTION TO PLACE THE BLOCK ON THE Q$TASK QUEUE.  HENCE
           *         WHEN TRAPPED, THE BLOCK WILL BE LINKED ON THE Q$TASK QUEUE
           *         AND THE INTERRUPT WILL BE TRANSPARENT TO THE CURRENTLY RUN-
           *         NING TASK UNLESS THE CURRENT TASK IS 'PAUSE'.  THE PARAMETERS
           *         ARE THEN FETCHED (SEE PROGRAMMER'S REFERENCE MANUAL IF YOU
           *         DON'T FULLY UNDERSTAND 'IDC' MODIFICATIONS: CAUSE IF YOU DON'T
           *         YOU WILL BE LOST) AND THE MME EXECUTED.  'EXIT' IS CALLED
           *         THUS PLACING THE CURRENTLY RUNNING TASK IN A BLOCKED STATE
           *         WHILE STARTING THE NEXT READY-TO-RUN TASK.  THIS IS HOW WE
           *         MULTI-PROGRAM.  WHEN THE EXEC TRAPS THIS OPERATION, AS
           *         WE HAVE SAID, THE XED INSTRUCTION PLACES THE TASK BACK
           *         ON THE Q$TASK QUEUE.
           *$*       DISK    SETFVM
```

SETUP FAULT VECTOR MACRO

```
              000526            USE    CODE
                                HEAD
                         *
                         *
                         *                              SETFV
                         *
                         SETFV  MACRO  CORELOC
                                TSX    0,$SETFV
                                ARG    #1               ADDRESS OF SLAVE FAULT VECTOR
                                ENDM   SETFV
                         *
                         *
                         *             SETFV -- SUBROUTINE
                         *
                         *      THIS SUBROUTINE IS CALLED BY THE SETFV MACRO.  IT ISSUES THE
                         *      COMMAND TO LOCATE THE SLAVE FAULT VECTOR.
                         *
                         *      CALL WITH
                         *             C(XT) = TBLOCK-ADDRESS
                         *             C(XJ) = JBLOCK ADDRESS
                         *      ENTER BY
                         *             TSX 0,$SETFV
                         *             ARG CORE LOCATION TO FAULT VECTOR
                         *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                         *      RETURNS WITH
                         *             C(XT) = TBLOCK-ADDRESS
                         *             C(XJ) = JBLOCK ADDRESS
                         *             C(XL) = RESTART ADDRESS
                         *      USES LOCAL TEMPORARY ONLY
                         *
                         *
000526  005320 7400 00   SETFV  STX    0,SETFT          POINTER TO ARGUMENT LIST
000527  000001 0200 03          ADLX   0,1,DU           RESTART ADDRESS
              000530            SETUP
000530  000510 7170 00          XED    $SETUP
000531  005320 2220 57          LDX    2,SETFT,IDC      LOAD ADDRESS OF FAULT VECTOR
000532  000001 2200 03          LDX    0,,SETFV,DU      LOAD MME NUMBER
              000533            CKPT                    CHECKPOINT
000533  000474 7170 00          XED    X$CKPT
000534  000000 0010 00          MME                     SETUP FAULT VECTOR
              000535            EXIT
000535  003074 7100 00          TRA    $EXIT
                         *
              005320            USE    STORE
005320  000000 0000 20   SETFT  ARG    0,*              POINTER TO ARGUMENT LIST
              000536            USE    PREVIOUS
                         *$*    DISK   READM
```

READ MACRO

```
                  000536   1268        USE     CODE
                           1269        HEAD
                           1270 *
                           1271 *
                           1272 *                          READ
                           1273 *
                           1274 READ   MACRO   FRN,CORELOC,N,MODE
                           1275        TSX     0,$READ
                           1276        ARG     #1              FRN ADDRESS
                           1277        ARG     #2              ADDRESS OF CORE LOC
                           1278        ARG     #3              NUMBER OF ELEMENTS
                           1279        ARG     #4              MODE
                           1280        ENDM    READ
                           1281 *
                           1282 *
                           1283 *              READ -- SUBROUTINE
                           1284 *
                           1285 *      THIS SUBROUTINE IS CALLED BY THE READ MACRO.  IT ISSUES THE
                           1286 *      COMMAND TO READ THE NEXT N ELEMENTS OF FRN IN A PARTICULAR MODE.
                           1287 *
                           1288 *      CALL WITH
                           1289 *              C(XT) = TBLOCK-ADDRESS
                           1290 *              C(XJ) = JBLOCK-ADDRESS
                           1291 *      ENTER BY
                           1292 *              TSX 0,$READ
                           1293 *              ARG ADDRESS OF FRN
                           1294 *              ARG ADDRESS OF CORELOC
                           1295 *              ARG N
                           1296 *              ARG MODE
                           1297 *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                           1298 *      RETURNS WITH
                           1299 *              C(XT) = TBLOCK-ADDRESS
                           1300 *              C(XJ) = JBLOCK-ADDRESS
                           1301 *              C(XL) = RESTART-ADDRESS
                           1302 *      USES LOCAL TEMPORARY ONLY
                           1303 *
                           1304 *
 END OF BINARY CARD IOS00011
    000536  005321 7400 00  1305 READ   STX     0,READT         POINTER TO ARGUMENT LIST
    000537  000004 0200 03  1306        ADLX    0,4,DU          RESTART ADDRESS
                  000540   1307        SETUP
    000540  000510 7170 00             XED     $SETUP
    000541  005321 2220 57  1308        LDX     2,READT,IDC     LOAD FRN
    000542  005321 2240 57  1309        LDX     4,READT,IDC     LOAD CORE LOC
    000543  005321 2250 57  1310        LDX     5,READT,IDC     LOAD N
    000544  005321 2260 57  1311        LDX     6,READT,IDC     LOAD MODE
    000545  000004 2200 03  1312        LDX     0,.READ,DU      LOAD MME NUMBER
                  000546   1313        CKPT                    CHECKPOINT
    000546  000474 7170 00             XED     X$CKPT
    000547  000000 0010 00  1314        MME                     READ
```

READ MACRO

```
                  000550    1315         EXIT                                                   580
000550  003074 7100 00                    TRA     $EXIT
                            1316 *                                                               590
                  005321    1317         USE     STORE                                           600
005321  000000 0000 20      1318 READT   ARG     0,*         POINTER TO ARGUMENT LIST            610
                  000551    1319         USE     PREVIOUS                                        620
                            1320 *$*     DISK    APPENDM                                         630
```

APEND MACRO

```
                     000551   1322          USE     CODE                                                110
                              1323          HEAD                                                        120
                              1324 *                                                                    130
                              1325 *                                                                    140
                              1326 *                                  APEND                             150
                              1327 *                                                                    160
                              1328 APEND    MACRO   FRN,CORELOC,N,MODE                                  170
                              1329          TSX     0,$APEND                                            180
                              1330          ARG     #1                FRN ADDRESS                       190
                              1331          ARG     #2                ADDRESS OF CORE LOC               200
                              1332          ARG     #3                NUMBER OF ELEMENTS                210
                              1333          ARG     #4                MODE                              220
                              1334          ENDM    APEND                                               230
                              1335 *                                                                    240
                              1336 *                                                                    250
                              1337 *                APEND -- SUBROUTINE                                 260
                              1338 *                                                                    270
                              1339 *        THIS SUBROUTINE IS CALLED BY THE APEND MACRO.  IT ISSUES    280
                              1340 *        THE COMMAND TO APEND N ELEMENTS TO THE FRN SPECIFIED        290
                              1341 *        VIA THE SPECIFIED MODE.                                     300
                              1342 *                                                                    310
                              1343 *        CALL WITH                                                   320
                              1344 *                C(XT) = TBLOCK-ADDRESS                              330
                              1345 *                C(XJ) = JBLOCK-ADDRESS                              340
                              1346 *        ENTER BY                                                    350
                              1347 *                TSX 0,$APEND                                        360
                              1348 *                ARG FRN ADDRSS                                      370
                              1349 *                ARG ADDRESS OF CORE LOC                             380
                              1350 *                ARG NUMBER OF ELEMENTS                              390
                              1351 *                ARG MODE                                            400
                              1352 *        RETURNS TO FIRST LOC AFTER MACRO EXPANSION                  410
                              1353 *        RETURNS WITH                                                420
                              1354 *                C(XT) = TBLOCK-ADDRESS                              430
                              1355 *                C(XJ) = JBLOCK ADDRESS                              440
                              1356 *                C(XL) = RESTART ADDRESS                             450
                              1357 *        USES LOCAL TEMPORARY ONLY                                   460
                              1358 *                                                                    470
                              1359 *                                                                    480
    000551  005322 7400 00    1360 APEND    STX     0,APNDT           POINTER TO ARGUMENT LIST          490
    000552  000004 0200 03    1361          ADLX    0,4,DU            RESTART ADDRESS                   500
                     000553   1362          SETUP                                                       510
    000553  000510 7170 00                  XED     $SETUP
    000554  005322 2220 57    1363          LDX     2,APNDT,IDC       LOAD FRN                          520
    000555  005322 2240 57    1364          LDX     4,APNDT,IDC       LOAD CORE LOC                     530
    000556  005322 2250 57    1365          LDX     5,APNDT,IDC       LOAD NUMBER OF ELEMENTS           540
    000557  005322 2260 57    1366          LDX     6,APNDT,IDC       LOAD MODE                         550
    000560  000005 2200 03    1367          LDX     0,.APEND,DU       LOAD MME NUMBER                   560
                     000561   1368          CKPT                      CHECKPOINT                        570
 END OF BINARY CARD IOS00012
    000561  000474 7170 00                  XED     X$CKPT
```

APEND MACRO

```
000562  000000 0010 00    1369        MME                    APEND                          580
                000563    1370        EXIT                                                  590
000563  003074 7100 00                TRA     $EXIT
                          1371 *                                                            600
                005322    1372        USE     STORE                                         610
005322  000000 0000 20    1373 APNDT  ARG     0,*            POINTER TO ARGUMENT LIST       620
                000564    1374        USE     PREVIOUS                                      630
                          1375 *$*    DISK    RRFM                                          640
```

READ RANDOM MACRO

```
                  000564     1377        USE     CODE
                             1378        HEAD
                             1379 *
                             1380 *
                             1381 *                              READR
                             1382 *
                             1383 RRF    MACRO   FRN,FILELOC,CORELOC,N
                             1384        TSX     0,$RRF
                             1385        ARG     #1              FILE REFERENCE ADDRESS
                             1386        ARG     #2              SOURCE ELEMENT ADDRESS
                             1387        ARG     #3              DESTINATION ADDRESS
                             1388        ARG     #4              NUMBER OF ELEMENTS TO TRANSMIT
                             1389        ENDM    RRF
                             1390 *
                             1391 *
                             1392 *              RRF -- SUBROUTINE
                             1393 *
                             1394 *      THIS SUBROUTINE IS CALLED BY THE RRF MACRO.  IT ISSUES THE
                             1395 *      COMMAND TO READ RANDOMLY FROM THE FILE SPECIFIED BY THE
                             1396 *      FRN STARTING AT ELEMENT NUMBER FILELOC TO CORE STARTING AT
                             1397 *      CORELOC FOR N ELEMENTS.
                             1398 *
                             1399 *      CALL WITH
                             1400 *              C(XT) = TBLOCK-ADDRESS
                             1401 *              C(XJ) = JBLOCK ADDRESS
                             1402 *      ENTER BY
                             1403 *              TSX 0,$RRF
                             1404 *              ARG FILE-REFERENCE-NUMBER
                             1405 *              ARG SOURCE-ELEMENT-NUMBER
                             1406 *              ARG DESTINATION-ADDRESS
                             1407 *              ARG NUMBER-OF-ELEMENTS
                             1408 *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                             1409 *      RETURNS WITH
                             1410 *              C(XT) = TBLOCK-ADDRESS
                             1411 *              C(XJ) = JBLOCK ADDRESS
                             1412 *              C(XL) = RESTART ADDRESS
                             1413 *      USES ONLY LOCAL TEMPORARY
                             1414 *
000564  005323 7400 00       1415 RRF    STX     0,RRFT          POINTER TO ARGUMENT LIST
000565  000004 0600 03       1416        ADX     0,4,DU          RESTART ADDRESS
                  000566     1417        SETUP
000566  000510 7170 00                   XED     $SETUP
000567  005323 2220 57       1418        LDX     2,RRFT,IDC      LOAD FRN
000570  005323 2230 57       1419        LDX     3,RRFT,IDC      LOAD SOURCE ELEMENT NUMBER
000571  005323 2240 57       1420        LDX     4,RRFT,IDC      DESTINATION ADDRESS
000572  005323 2250 57       1421        LDX     5,RRFT,IDC      NUMBER OF ELEMENTS TO TRANSFER
000573  000006 2200 03       1422        LDX     0,.RRF,DU       LOAD MME NUMBER
                  000574     1423        CKPT                    CHECKPOINT
000574  000474 7170 00                   XED     X$CKPT
000575  000000 0010 00       1424        MME                     READ RANDOM FILE
```

READ RANDOM MACRO

```
                        000576     1425           EXIT                                                      590
  000576  003074 7100 00                          TRA     $EXIT
                                   1426 *                                                                   600
                        005323     1427           USE     STORE                                             610
  005323  000000 0000 20           1428 RRFT      ARG     0,*              POINTER TO ARGUMENT LIST         620
                        000577     1429           USE     PREVIOUS                                          630
                                   1430 *$*       DISK    WRFM                                              640
```

WRITE RANDOM MACRO

```
                    000577     1432         USE     CODE
                               1433         HEAD
                               1434 *
                               1435 *
                               1436 *                               WRF
                               1437 *
                               1438 WRF     MACRO   FRN,FILELOC,CORELOC,N
                               1439         TSX     0,$WRF
                               1440         ARG     #1                   FILE REFERENCE ADDRESS
                               1441         ARG     #2                   DESTINATION ELEMENT ADDRESS
                               1442         ARG     #3                   SOURCE ADDRESS (CORE)
                               1443         ARG     #4                   NUMBER OF ELEMENTS TO TRANSMIT
                               1444         ENDM    WRF
                               1445 *
                               1446 *
                               1447 *               WRF -- SUBROUTINE
                               1448 *
                               1449 *       THIS COMMAND IS CALLED BY THE WRF MACRO.  IT ISSUES THE
                               1450 *       COMMAND TO WRITE RANDOMLY TO THE FILE SPECIFIED BY THE
                               1451 *       FRN STARTING AT ELEMENT NUMBER FILELOC FROM CORE STARTING AT
                               1452 *       CORELOC FOR N ELEMENTS.
                               1453 *
                               1454 *       CALL WITH
                               1455 *               C(XT) = TBLOCK ADDRESS
                               1456 *               C(XJ) = JBLOCK ADDRESS
                               1457 *       ENTER BY
                               1458 *               TSX 0,$WRF
                               1459 *               ARG FILE-REFERENCE-NUMBER
                               1460 *               ARG DESTINATION-ELEMENT-NUMBER
                               1461 *               ARG SOURCE-ADDRESS
                               1462 *               ARG NUMBER-OF-ELEMENTS
                               1463 *       RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                               1464 *       RETURNS WITH
                               1465 *               C(XT) = TBLOCK ADDRESS
                               1466 *               C(XJ) = JBLOCK ADDRESS
                               1467 *               C(XL) = RESTART ADDRESS
                               1468 *       USES ONLY LOCAL TEMPORARY
                               1469 *
   000577  005324 7400 00      1470 WRF     STX     0,WRFT               POINTER TO ARGUMENT LIST
   000600  000004 0600 03      1471         ADX     0,4,DU               RESTART ADDRESS
                    000601     1472         SETUP
 END OF BINARY CARD IOS00013
   000601  000510 7170 00               XED     $SETUP
   000602  005324 2220 57      1473         LDX     2,WRFT,IDC           LOAD FRN
   000603  005324 2230 57      1474         LDX     3,WRFT,IDC           LOAD DESTINATION ELEMENT NUMBER
   000604  005324 2240 57      1475         LDX     4,WRFT,IDC           LOAD SOURCE ADDRESS
   000605  005324 2250 57      1476         LDX     5,WRFT,IDC           NUMBER OF ELEMENTS TO TRANSFER
   000606  000007 2200 03      1477         LDX     0,.WRF,DU            LOAD MME NUMBER
   000607  000000 0010 00      1478         MME                          WRITE RANDOM FILE
                    000610     1479         CKPT                         CHECKPOINT
```

WRITE RANDOM MACRO

```
000610  000474 7170 00                      XED     X$CKPT
                 000611      1480           EXIT
000611  003074 7100 00                      TRA     $EXIT
                             1481 *
                 005324      1482           USE     STORE
005324  000000 0000 20       1483 WRFT      ARG     0,*               POINTER TO ARGUMENT LIST
                 000612      1484           USE     PREVIOUS
                             1485 *$*       DISK    SCRM
```

SCRATCH MACRO

```
                    000612    1487         USE      CODE                                                      110
                              1488         HEAD                                                               120
                              1489 *                                                                          130
                              1490 *                                                                          140
                              1491 *                                 SCRATCH                                  150
                              1492 *                                                                          160
                              1493 SCR     MACRO    FRN,FLOC                                                  170
                              1494         TSX      0,$SCR                                                    180
                              1495         ARG      #1               FILE REFERENCE ADDRESS                   190
                              1496         ARG      #2               FILE LOCATION (ELEMENTS)                 200
                              1497         ENDM     SCR                                                       210
                              1498 *                                                                          220
                              1499 *                                                                          230
                              1500 *                SCRATCH -- SUBROUTINE                                     240
                              1501 *                                                                          250
                              1502 *       THIS SUBROUTINE IS CALLED BY THE SCR MACRO.  IT ISSUES THE         260
                              1503 *       COMMAND TO SCRATCH A FILE.  NOTE THAT THE FILE IS SCRATCHED TO     270
                              1504 *       THE BEGINNING.                                                     280
                              1505 *                                                                          290
                              1506 *       CALL WITH                                                          300
                              1507 *                C(XT) = TBLOCK ADDRESS                                    310
                              1508 *                C(XJ) = JBLOCK ADDRESS                                    320
                              1509 *       ENTER BY                                                           330
                              1510 *                TSX 0,$SCR                                                340
                              1511 *                ARG FILE-REFERENCE-ADDRESS                                350
                              1512 *       RETURNS TO FIRST LOC AFTER MACRO EXPANSION                         360
                              1513 *       RETURNS WITH                                                       370
                              1514 *                C(XT) = TBLOCK ADDRESS                                    380
                              1515 *                C(XJ) = JBLOCK ADDRESS                                    390
                              1516 *                C(XL) = RESTART ADDRESS                                   400
                              1517 *       USES ONLY LOCAL TEMPORARY                                          410
                              1518 *                                                                          420
  000612  005325 7400 00      1519 SCR     STX      0,SCRT           POINTER TO ARGUMENT LIST                 430
  000613  000002 0600 03      1520         ADX      0,2,DU           RESTART ADDRESS                          440
                    000614    1521         SETUP                                                              450
  000614  000510 7170 00               XED      $SETUP
  000615  005325 2220 57      1522         LDX      2,SCRT,IDC       LOAD FILE REFERENCE NUMBER               460
  000616  005325 2230 57      1523         LDX      3,SCRT,IDC       LOAD STARTING SCRATCH ADDRESS            470
  000617  000010 2200 03      1524         LDX      0,.SCR,DU        LOAD MME NUMBER                          480
                    000620    1525         CKPT                      CHECKPOINT                               490
  000620  000474 7170 00               XED      X$CKPT
  000621  000000 0010 00      1526         MME                       SCRATCH FILE                             500
                    000622    1527         EXIT                                                               510
  000622  003074 7100 00               TRA      $EXIT
                              1528 *                                                                          520
                    005325    1529         USE      STORE                                                     530
END OF BINARY CARD IOS00014
  005325  000000 0000 20      1530 SCRT    ARG      0,*              POINTER TO ARGUMENT LIST                 540
                    000623    1531         USE      PREVIOUS                                                  550
                              1532 *$*     DISK     SPTRM                                                     560
```

SET POINTER MACRO

```
                000623   1534        USE     CODE
                         1535        HEAD
                         1536 *
                         1537 *
                         1538 *                            SET POINTER
                         1539 *
                         1540 SPTR   MACRO   FRN,N
                         1541        TSX     0,$SPTR
                         1542        ARG     #1                FRN ADDRESS
                         1543        ARG     #2                NUMBER OF ELEMENTS TO MOVE POINTER
                         1544        ENDM    SPTR
                         1545 *
                         1546 *
                         1547 *              SET POINTER -- SUBROUTINE
                         1548 *
                         1549 *      THIS SUBROUTINE IS CALLED BY THE  SPTR  MACRO.  IT ISSUES
                         1550 *      THE COMMAND TO ADD (OR SUBTRACT) N ELEMENTS TO THE CUR-
                         1551 *      RENT SETTING OF THE READ POINTER.
                         1552 *
                         1553 *      CALL WITH
                         1554 *              C(XT) = TBLOCK-ADDRESS
                         1555 *              C(XJ) = JBLOCK-ADDRESS
                         1556 *      ENTER BY
                         1557 *              TSX 0,$SPTR
                         1558 *              ARG FRN
                         1559 *              ARG NUMBER OF ELEMENTS
                         1560 *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                         1561 *      RETURNS WITH
                         1562 *              C(XJ) = JCB
                         1563 *              C(XT) = TCB
                         1564 *              C(XL) = RESTART ADDRESS
                         1565 *      USES LOCAL TEMPORARY ONLY
                         1566 *
000623  005326 7400 00   1567 SPTR   STX     0,SPTRT           POINTER TO ARGUMENT LIST
000624  000002 0200 03   1568        ADLX    0,2,DU            RESTART ADDRESS
                000625   1569        SETUP
000625  000510 7170 00               XED     $SETUP
000626  005326 2220 57   1570        LDX     2,SPTRT,IDC       LOAD FRN
000627  005326 2230 57   1571        LDX     3,SPTRT,IDC       LOAD N
000630  000011 2200 03   1572        LDX     0,.SPTR,DU        LOAD MME NUMBER
                000631   1573        CKPT                      CHECKPOINT
000631  000474 7170 00               XED     X$CKPT
000632  000000 0010 00   1574        MME                       SET POINTER
                000633   1575        EXIT
000633  003074 7100 00               TRA     $EXIT
                         1576 *
                005326   1577        USE     STORE
005326  000000 0000 20   1578 SPTRT  ARG     0,*               POINTER TO ARGUMENT LIST
                000634   1579        USE     PREVIOUS
                         1580 *$*    DISK    RQSTM
```

REQUEST STATUS MACRO

```
                  000634     1582          USE     CODE
                             1583          HEAD
                             1584 *
                             1585 *
                             1586 *                            REQUEST STATUS
                             1587 *
                             1588 RQST     MACRO   FRN
                             1589          TSX     0,$RQST
                             1590          ARG     #1              FRN ADDRESS
                             1591          ENDM    RQST
                             1592 *
                             1593 *
                             1594 *                REQUEST STATUS -- SUBROUTINE
                             1595 *
                             1596 *        THIS SUBROUTINE IS CALLED BY THE  RQST  MACRO.  IT ISSUES
                             1597 *        THE COMMAND TO REQUEST STATUS ON THE FRN SPECIFIED.
                             1598 *
                             1599 *        CALL WITH
                             1600 *                C(XT) = TCB
                             1601 *                C(XJ) = JCB
                             1602 *        ENTER BY
                             1603 *                TSX 0,$RQST
                             1604 *                ARG FRN
                             1605 *        RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                             1606 *        RETURNS WITH
                             1607 *                C(XT) = TCB
                             1608 *                C(XJ) = JCB
                             1609 *                C(XL) = RESTART ADDRESS
                             1610 *        USES LOCAL TEMPORARY ONLY
                             1611 *
                             1612 *
 000634  005327 7400 00      1613 RQST     STX     0,RQSTT         POINTER TO ARGUMENT LIST
 000635  000001 0200 03      1614          ADLX    0,1,DU          RESTART ADDRESS
                  000636     1615          SETUP
 000636  000510 7170 00                    XED     $SETUP
 000637  005327 2220 57      1616          LDX     2,RQSTT,IDC     LOAD FRN
 000640  000012 2200 03      1617          LDX     0,.RQST,DU      LOAD MME NUMBER
                  000641     1618          CKPT                    CHECKPOINT
 000641  000474 7170 00                    XED     X$CKPT
 000642  000000 0010 00      1619          MME                     REQUEST STATUS
                  000643     1620          EXIT
 000643  003074 7100 00                    TRA     $EXIT
                             1621 *
                  005327     1622          USE     STORE
END OF BINARY CARD IOS00015
 005327  000000 0000 20      1623 RQSTT    ARG     0,*             POINTER TO ARGUMENT LIST
                  000644     1624          USE     PREVIOUS
                             1625 *$*      DISK    SPAWNM
```

SPAWN MACRO

```
                     000644          USE     CODE
                                     HEAD
                                *
                                *
                                *                            SPAWN
                                *
                                SPAWN   MACRO   PLOC,LENGTH,ORIGINATOR
                                        TSX     0,$SPAWN
                                        ARG     #1              PARAMETER LIST ADDRESS
                                        ARG     #2              LENGTH ADDRESS
                                        ARG     #3              ORIGINATOR ADDRESS
                                        ENDM    SPAWN
                                *
                                *
                                *               SPAWN -- SUBROUTINE
                                *
                                *       THIS SUBROUTINE IS CALLED BY THE  SPAWN  MACRO.  IT ISSUES THE
                                *       COMMAND TO SPAWN A PROGRAM.
                                *
                                *       CALL WITH
                                *               C(XT) = TBLOCK ADDRESS
                                *               C(XJ) = JBLOCK ADDRESS
                                *       ENTER BY
                                *               TSX 0,SPAWN
                                *               ARG PARAMETER LIST ADDRESS
                                *               ARG LENGTH ADDRESS
                                *               ARG ORIGINATOR ADDRESS
                                *       RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                                *       RETURNS WITH
                                *               C(XT) = TBLOCK ADDRESS
                                *               C(XJ) = JBLOCK ADDRESS
                                *               C(XL) = RESTART ADDRESS
                                *       USES ONLY LOCAL TEMPORARY
                                *
000644  005330 7400 00          SPAWN   STX     0,SPWNT         POINTER TO ARGUMENT LIST
000645  000003 0600 03                  ADX     0,3,DU          RESTART ADDRESS
                     000646             SETUP
000646  000510 7170 00                  XED     $SETUP
000647  005330 2240 57                  LDX     4,SPWNT,IDC     LOAD PARAMETER LIST ADDRESS
000650  005330 2250 57                  LDX     5,SPWNT,IDC     LOAD P. L. ADDRESS
000651  005330 2360 57                  LDQ     SPWNT,IDC       LOAD ORIGINATOR
000652  000015 2200 03                  LDX     0,.SPAWN,DU     LOAD MME NUMBER
                     000653             CKPT                    CHECKPOINT
000653  000474 7170 00                  XED     X$CKPT
000654  000000 0010 00                  MME                     ISSUE SPAWN
                     000655             EXIT
000655  003074 7100 00                  TRA     $EXIT
                                *
                     005330             USE     STORE
005330  000000 0000 20          SPWNT   ARG     0,*             POINTER TO ARGUMENT LIST
```

SPAWN MACRO

```
      000656      1674          USE       PREVIOUS                                   580
                  1675 *$*      DISK      CHSEGM                                     590
```

CHANGE SEGMENT MACRO

```
                     000656      1677          USE     CODE                                       110
                                 1678          HEAD                                               120
                                 1679 *                                                           130
                                 1680 *                                                           140
                                 1681 *                            CHSEG                          150
                                 1682 *                                                           160
                                 1683 CHSEG    MACRO   SEGMENT-NUMBER,LENGTH                      170
                                 1684          TSX     0,$CHSEG                                   180
                                 1685          ARG     #1              NUMBER-OF-SEGMENT          190
                                 1686          ARG     #2              NEW-LENGTH                 200
                                 1687          ENDM    CHSEG                                      210
                                 1688 *                                                           220
                                 1689 *                                                           230
                                 1690 *                CHSEG -- SUBROUTINE                        240
                                 1691 *                                                           250
                                 1692 *        THIS SUBROUTINE IS CALLED BY THE CHSEG MACRO.  IT ISSUES   260
                                 1693 *        THE COMMAND TO CHANGE THE LENGTH OF THE NAMED SEGMENT TO   270
                                 1694 *        THE NEW LENGTH SPECIFIED.                          280
                                 1695 *                                                           290
                                 1696 *        CALL WITH                                          300
                                 1697 *                C(XT) = TBLOCK-ADDRESS                     310
                                 1698 *                C(XJ) = JBLOCK ADDRESS                     320
                                 1699 *        ENTER BY                                           330
                                 1700 *                TSX 0,$CHSEG                               340
                                 1701 *                ARG SEGMENT-NUMBER                         350
                                 1702 *                ARG LENGTH                                 360
                                 1703 *        RETURNS TO FIRST LOC AFTER MACRO EXPANSION         370
                                 1704 *        RETURNS WITH                                       380
                                 1705 *                C(XT) = TBLOCK-ADDRESS                     390
                                 1706 *                C(XJ) = JBLOCK ADDRESS                     400
                                 1707 *                C(XL) = RESTART ADDRESS                    410
                                 1708 *        USES LOCAL TEMPORARY ONLY                          420
                                 1709 *                                                           430
    000656  005331 7400 00       1710 CHSEG    STX     0,CHSGT         POINTER TO ARGUMENT LIST   440
    000657  000002 0200 03       1711          ADLX    0,2,DU          RESTART ADDRESS            450
                     000660      1712          SETUP                                              460
    000660  000510 7170 00                     XED     $SETUP
    000661  005331 2220 57       1713          LDX     2,CHSGT,IDC     LOAD SEGMENT NUMBER        470
    000662  005331 2230 57       1714          LDX     3,CHSGT,IDC     LOAD SEGMENT LENGTH        480
    000663  000022 2200 03       1715          LDX     0,.CHSEG,DU     LOAD MME NUMBER            490
                     000664      1716          CKPT                    CHECKPOINT                 500
    000664  000474 7170 00                     XED     X$CKPT
  END OF BINARY CARD IOS00016
    000665  000000 0010 00       1717          MME                     CHANGE SEGMENT             510
                     000666      1718          EXIT                                               520
    000666  003074 7100 00                     TRA     $EXIT
                                 1719 *                                                           530
                     005331      1720          USE     STORE                                      540
    005331  000000 0000 20       1721 CHSGT    ARG     0,*             POINTER TO ARGUMENT LIST   550
                     000667      1722          USE     PREVIOUS                                   560
```

CHANGE SEGMENT MACRO

```
1723 *$*     DISK     OPENM                                              570
```

OPEN MACRO

```
                     000667          USE     CODE
                                     HEAD
                               *
                               *
                               *                               OPEN
                               *
                               OPEN  MACRO   TREENAME,TREESIZE,BEHALF,ELSIZE,ACCESSES
                                     TSX     0,$OPEN
                                     ARG     #1                TREE-NAME-ADDRESS
                                     ARG     #2                TREE-SIZE-ADDRESS
                                     ARG     #3                BEHALF
                                     ARG     #4                ELEMENT SIZE ADDRESS
                                     ARG     #5                ACCESSES
                                     ENDM    OPEN
                               *
                               *
                               *             OPEN -- SUBROUTINE
                               *
                               *     THIS SUBROUTINE IS CALLED BY THE OPEN MACRO.  IT ISSUES THE
                               *     COMMAND TO OPEN A FILE.
                               *
                               *     CALL WITH
                               *             C(XT) = TBLOCK ADDRESS
                               *             C(XJ) = JBLOCK ADDRESS
                               *     ENTER BY
                               *             TSX 0,$OPEN
                               *             ARG TREE-NAME-ADDRESS
                               *             ARG TREE-SIZE
                               *             ARG BEHALF
                               *             ARG ELEMENT-SIZE
                               *             ARG ACCESSES
                               *     RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                               *     RETURNS WITH
                               *             C(XT) = TBLOCK ADDRSS
                               *             C(XJ) = JBLOCK ADDRESS
                               *             C(XJ) = RESTART ADDRESS
                               *     USES ONLY LOCAL TEMPORARY
                               *
000667  005332 7400 00         OPEN  STX     0,OPENT           POINTER TO ARGUMENT LIST
000670  000005 0600 03               ADX     0,5,DU            RESTART ADDRESS
                     000671          SETUP
000671  000510 7170 00               XED     $SETUP
000672  005332 2240 57               LDX     4,OPENT,IDC       TREE-NAME ADDRESS
000673  005332 2250 57               LDX     5,OPENT,IDC       TREE-SIZE
000674  005332 2200 57               LDX     0,OPENT,IDC       SETTING -- SPARE XJ
000675  005332 2270 57               LDX     7,OPENT,IDC       LOAD ELEMENT SIZE
000676  005332 2360 57               LDQ     OPENT,IDC         ACCESSES
000677  000000 6260 10               EAX     6,0,0             MOVE SETTING INTO X6
000700  000024 2200 03               LDX     0,.OPEN,DU        MME NUMBER
                     000701          CKPT                      CHECKPOINT
```

OPEN MACRO

```
000701  000474 7170 00                     XED     X$CKPT
000702  000000 0010 00        1774         MME                       OPEN                          600
                  000703      1775         EXIT                                                    610
000703  003074 7100 00                     TRA     $EXIT
                              1776 *                                                             620
                  005332      1777         USE     STORE                                         630
005332  000000 0000 20        1778 OPENT   ARG     0,*               POINTER TO ARGUMENT LIST      640
                  000704      1779         USE     PREVIOUS                                      650
                              1780 *$*     DISK    CLOSEM                                        660
```

CLOSE MACRO

```
                    000704    1782        USE     CODE                                               110
                              1783        HEAD                                                       120
                              1784 *                                                                 130
                              1785 *                                                                 140
                              1786 *                            CLOSE                                150
                              1787 *                                                                 160
                              1788 CLOSE  MACRO   FRN                                                170
                              1789        TSX     0,$CLOSE                                           180
                              1790        ARG     #1            FILE REFERENCE ADDRESS               190
                              1791        ENDM    CLOS                                               200
                              1792 *                                                                 210
                              1793 *                                                                 220
                              1794 *              CLOSE -- SUBROUTINE                                230
                              1795 *                                                                 240
                              1796 *      THIS SUBROUTINE IS CALLED BY THE CLOS MACRO.  IT ISSUES THE 250
                              1797 *      MME TO CLOSE A FILE.                                       260
                              1798 *                                                                 270
                              1799 *      CALL WITH                                                  280
                              1800 *              C(XT) = TBLOCK-ADDRESS                             290
                              1801 *              C(XJ) = JBLOCK ADDRESS                             300
                              1802 *      ENTER BY                                                   310
                              1803 *              TSX 0,$CLOS                                        320
                              1804 *              ARG FILE-REFERENCE-ADDRESS                         330
                              1805 *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION                 340
                              1806 *      RETURNS WITH                                               350
                              1807 *              C(XT) = TBLOCK-ADDRESS                             360
                              1808 *              C(XJ) = JBLOCK ADDRESS                             370
                              1809 *              C(XL) = RESTART ADDRESS                            380
                              1810 *      USES LOCAL TEMPORARY ONLY                                  390
                              1811 *                                                                 400
  000704  005333 7400 00      1812 CLOSE  STX     0,CLOST       POINTER TO ARGUMENT LIST             410
END OF BINARY CARD IOS00017
  000705  000001 0200 03      1813        ADLX    0,1,DU        RESTART ADDRESS                      420
                    000706    1814        SETUP                                                      430
  000706  000510 7170 00                  XED     $SETUP
  000707  005333 2220 57      1815        LDX     2,CLOST,IDC   LOAD FILE REFERENCE                  440
  000710  000025 2200 03      1816        LDX     0,.CLOSE,DU   LOAD MME NUMBER                      450
                    000711    1817        CKPT                  CHECKPOINT                           460
  000711  000474 7170 00                  XED     X$CKPT
  000712  000000 0010 00      1818        MME                   CLOS                                 470
                    000713    1819        EXIT                                                       480
  000713  003074 7100 00                  TRA     $EXIT
                              1820 *                                                                 490
                    005333    1821        USE     STORE                                              500
  005333  000000 0000 20      1822 CLOST  ARG     0,*           POINTER TO ARGUMENT LIST             510
                    000714    1823        USE     PREVIOUS                                           520
                              1824 *$*    DISK    DESTROM                                            530
```

DESTROY MACRO

```
                   000714          USE     CODE
                                   HEAD
                           *
                           *
                           *                               DESTROY (UNCATALOG BY TREE-NAME)
                           *
                           DESTRO  MACRO   TREE-NAME,TREE-SIZE,BEHALF
                                   TSX     0,$DESTRO
                                   ARG     #1              TREE-NAME ADDRESS
                                   ARG     #2              TREE-SIZE ADDRESS
                                   ARG     #3              BEHALF
                                   ENDM    DEST
                           *
                           *
                           *               DESTROY -- SUBROUTINE
                           *
                           *       THIS SUBROUTINE IS CALLED BY THE DEST MACRO.  IT ISSUES THE
                           *       COMMAND TO DESTROY (I.E. UNCATALOG) A FILE.
                           *
                           *       CALL WITH
                           *               C(XT) = TBLOCK ADDRESS
                           *               C(XJ) = JBLOCK ADDRESS
                           *       ENTER BY
                           *               TSX 0,$DEST
                           *               ARG TREE-NAME-ADDRESS
                           *               ARG TREE-SIZE
                           *               ARG BEHALF
                           *       RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                           *       RETURNS WITH
                           *               C(XT) = TBLOCK ADDRESS
                           *               C(XJ) = JBLOCK ADDRESS
                           *               C(XL) = RESTART ADDRESS
                           *       USES ONLY LOCAL TEMPORARY
                           *
000714  005334 7400 00     DESTRO  STX     0,DESTT         POINTER TO ARGUMENT LIST
000715  000003 0600 03             ADX     0,3,DU          RESTART ADDRESS
                   000716          SETUP
000716  000510 7170 00             XED     $SETUP
000717  005334 2240 57             LDX     4,DESTT,IDC     LOAD TREE-NAME
000720  005334 2250 57             LDX     5,DESTT,IDC     TREE-SIZE
000721  005334 2260 57             LDX     6,DESTT,IDC     BEHALF
000722  000027 2200 03             LDX     0,.DESTR,DU     MME NUMBER
                   000723          CKPT                    CHECKPOINT
000723  000474 7170 00             XED     X$CKPT
000724  000000 0010 00             MME                     DESTROY
                   000725          EXIT
000725  003074 7100 00             TRA     $EXIT
                           *
                   005334          USE     STORE
005334  000000 0000 20     DESTT   ARG     0,*             POINTER TO ARGUMENT LIST
```

DESTROY MACRO

```
                    000726      1873            USE     PREVIOUS                                         .      580
                                1874 *$*        DISK    UPDATEM                                                 590
```

UPDATE MACRO

```
                   000726    1876        USE     CODE
                             1877        HEAD
                             1878 *
                             1879 *
                             1880 *                                UPDATE
                             1881 *
                             1882 UPDATE MACRO   FRN
                             1883        TSX     0,$UPDATE
                             1884        ARG     #1                FILE REFERNECE ADDRESS
                             1885        ENDM    UPDAT
                             1886 *
                             1887 *
                             1888 *              UPDATE -- SUBROUTINE
                             1889 *
                             1890 *      THIS SUBROUTINE IS CALLED BY THE UPDAT MACRO.  IT ISSUES THE
                             1891 *      UPDATE MME ON THE FRN SPECIFIED
                             1892 *
                             1893 *      CALL WITH
                             1894 *              C(XT) = TBLOCK ADDRESS
                             1895 *              C(XJ) = JBLOCK ADDRESS
                             1896 *      ENTER BY
                             1897 *              TSX 0,$UPDATE
                             1898 *              ARG FILE-REFERENCE-ADDRESS
                             1899 *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                             1900 *      RETURNS WITH
                             1901 *              C(XT) = TBLOCK ADDRESS
                             1902 *              C(XJ) = JBLOCK ADDRESS
                             1903 *              C(XL) = RETART ADDRESS
                             1904 *      USES ONLY LOCAL TEMPORARY
                             1905 *
 END OF BINARY CARD IOS00018
    000726  005335 7400 00   1906 UPDATE STX     0,UPDTT           POINTER TO ARGUMENT LIST
    000727  000001 0600 03   1907        ADX     0,1,DU            RESTART ADDRESS
                   000730    1908        SETUP
    000730  000510 7170 00               XED     $SETUP
    000731  005335 2220 57   1909        LDX     2,UPDTT,IDC       LOAD FILE REFERENCE
    000732  000031 2200 03   1910        LDX     0,,UPDAT,DU       MME NUMBER
                   000733    1911        CKPT                      CHECKPOINT
    000733  000474 7170 00               XED     X$CKPT
    000734  000000 0010 00   1912        MME                       UPDATE
                   000735    1913        EXIT
    000735  003074 7100 00               TRA     $EXIT
                             1914 *
                   005335    1915        USE     STORE
    005335  000000 0000 20   1916 UPDTT  ARG     0,*               POINTER TO ARGUMENT LIST
                   000736    1917        USE     PREVIOUS
                             1918 *$*    DISK    LOCKM
```

LOCK MACRO

```
                  000736    USE    CODE
                            HEAD
                  *
                  *
                  *                       LOCK
                  *
                  LOCK      MACRO  FRN
                            TSX    0,$LOCK
                            ARG    #1            FILE-REFERENCE ADDRESS
                            ENDM   LOCK
                  *
                  *
                  *                LOCK -- SUBROUTINE
                  *
                  *         THIS SUBROUTINE IS CALLED BY THE LOCK MACRO.  IT ISSUES THE
                  *         MME TO LOCK A FILE.
                  *
                  *         CALL WITH
                  *                C(XT) = TBLOCK-ADDRESS
                  *                C(XJ) = JBLOCK ADDRESS
                  *         ENTER BY
                  *                TSX 0,$LOCK
                  *                ARG FILE-REFERENCE-ADDRESS
                  *         RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                  *         RETURNS WITH
                  *                C(XT) = TBLOCK-ADDRESS
                  *                C(XJ) = JBLOCK ADDRESS
                  *                C(XL) = RESTART ADDRESS
                  *         USES LOCAL TEMPORARY ONLY
                  *
000736  005336 7400 00  LOCK  STX    0,LOCKT       POINTER TO ARGUMENT LIST
000737  000001 0200 03        ADLX   0,1,DU        RESTART ADDRESS
                  000740      SETUP
000740  000510 7170 00        XED    $SETUP
000741  005336 2220 57        LDX    2,LOCKT,IDC   LOAD FILE REFERENCE
000742  000044 2200 03        LDX    0,,LOCK,DU    MME NUMBER
                  000743      CKPT                 CHECKPOINT
000743  000474 7170 00        XED    X$CKPT
000744  000000 0010 00        MME                  LOCK
                  000745      EXIT
000745  003074 7100 00        TRA    $EXIT
                  *
                  005336      USE    STORE
005336  000000 0000 20  LOCKT ARG    0,*           POINTER TO ARGUMENT LIST
                  000746      USE    PREVIOUS
                  *$*         DISK   UNLOCKM
```

UNLOCK MACRO

```
                    000746    1964        USE     CODE
                              1965        HEAD
                              1966 *
                              1967 *
                              1968 *                        UNLOCK
                              1969 *
                              1970 UNLCK  MACRO   FRN
                              1971        TSX     0,$UNLCK
                              1972        ARG     #1             FILE REFERENCE ADDRESS
                              1973        ENDM    UNLK
                              1974 *
                              1975 *
                              1976 *              UNLOCK -- SUBROUTINE
                              1977 *
                              1978 *      THIS SUBROUTINE IS CALLED BY THE UNLK MACRO.  IT ISSUES THE
                              1979 *      MME TO UNLOCK A FILE.
                              1980 *
                              1981 *      CALL WITH
                              1982 *              C(XT) = TBLOCK-ADDRESS
                              1983 *              C(XJ) = JBLOCK ADDRESS
                              1984 *      ENTER BY
                              1985 *              TSX 0,$UNLK
                              1986 *              ARG FILE-REFERENCE-ADDRESS
                              1987 *      RETURNS TO FIRST LOC AFTER THE MACRO EXPANSION
                              1988 *      RETURNS WITH
                              1989 *              C(XT) = TBLOCK-ADDRESS
                              1990 *              C(XJ) = JBLOCK ADDRESS
                              1991 *              C(XL) = RESTART ADDRESS
                              1992 *      USES LOCAL TEMPORARY ONLY
                              1993 *
END OF BINARY CARD IOS00019
    000746  005337 7400 00    1994 UNLCK  STX     0,UNLKT        POINTER TO ARGUMENT LIST
    000747  000001 0200 03    1995        ADLX    0,1,DU         RESTART ADDRESS
                    000750    1996        SETUP
    000750  000510 7170 00                XED     $SETUP
    000751  005337 2220 57    1997        LDX     2,UNLKT,IDC    LOAD FILE REFERENCE
    000752  000045 2200 03    1998        LDX     0,,UNLCK,DU    MME NUMBER
                    000753    1999        CKPT                   CHECKPOINT
    000753  000474 7170 00                XED     X$CKPT
    000754  000000 0010 00    2000        MME                    UNLOCK
                    000755    2001        EXIT
    000755  003074 7100 00                TRA     $EXIT
                              2002 *
                    005337    2003        USE     STORE
    005337  000000 0000 20    2004 UNLKT  ARG     0,*            POINTER TO ARGUMENT LIST
                    000756    2005        USE     PREVIOUS
                              2006 *S*    DISK    NOTIFM
```

NOTIFY MACRO

```
                 000756        USE     CODE
                               HEAD
                       *
                       *                        NOTIFY
                       *
                       NOTIF   MACRO   ERN,STATE
                               TSX     0,$NOTIF
                               ARG     #1               FRN ADDRESS
                               ARG     #2               STATE ADDRESS
                               ENDM    NOTIF
                       *
                       *
                       *               NOTIFY -- SUBROUTINE
                       *
                       *       THIS SUBROUTINE IS CALLED BY THE  NOTIF  MACRO.  IT ISSUES
                       *       THE NOTIFY ON THE SPECIFIED EVENT.  NOTE THAT C(X3) POINT TO
                       *       CTRAP.
                       *
                       *       CALL WITH
                       *               C(XT) = TBLOCK-ADDRESS
                       *               C(XJ) = JBLOCK ADDRESS
                       *               C(X3) = CTRAP ADDRESS
                       *       ENTER BY
                       *               TSX 0,NOTIF
                       *               ARG EVENT-FRN
                       *               ARG STATE ADDRESS
                       *       RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                       *       RETURNS WITH
                       *               C(XT) = TBLOCK-ADDRESS
                       *               C(XJ) = JBLOCK ADDRESS
                       *               C(XL) = RESTART ADDRESS
                       *       USES LOCAL TEMPORARY ONLY
                       *
000756  005340 7400 00 NOTIF   STX     0,NOTFT          POINTER OF ARGUMENT LIST
000757  000002 0200 03         ADLX    0,2,DU           RESTART ADDRESS
                 000760        SETUP
000760  000510 7170 00         XED     $SETUP
000761  005340 2220 57         LDX     2,NOTFT,IDC      LOAD EVENT FRN
000762  005340 2350 57         LDA     NOTFT,IDC        LOAD STATE
000763  000046 2200 03         LDX     0,,NOTIF,DU      LOAD MME NUMBER
                 000764        CKPT                     CHECKPOINT
000764  000474 7170 00         XED     X$CKPT
000765  000000 0010 00         MME                      NOTIFY
                 000766        EXIT
000766  003074 7100 00         TRA     $EXIT
                       *
                 005340        USE     STORE
005340  000000 0000 20 NOTFT   ARG     0,*              POINTER TO ARGUMENT LIST
                 000767        USE     PREVIOUS
                       *$*     DISK    CAUSEM
```

CAUSE MACRO

```
                        000767          USE     CODE
                                        HEAD
                                *
                                *
                                *                               CAUSE
                                *
                                CAUSE   MACRO   ERN,NUMBER,STATE,MESSAGE,ACCESSES,FRN (N.B. ORDERING)
                                        TSX     0,$CAUSE
                                        ARG     #1              FILE REFERENCE ADDRESS
                                        ARG     #2              NUMBER WHICH ARE TO NOTIFIED
                                        ARG     #3              STATE
                                        ARG     #4              MESSAGE
                                        ARG     #5              FRN OF ITME TO PASS
                                        ARG     #6              ACCESSES ON PASSED ITEM
                                        ENDM    CAUS
                                *
                                *
                                *               CAUSE -- SUBROUTINE
                                *
                                *       THIS SUBROUTINE IS CALLED BY THE CAUS MACRO.  IT ISSUES THE
                                *       MME TO CAUSE THE SPECIFIED FILE.
                                *
                                *       CALL WITH
                                *               C(XJ) = JBLOCK ADDRESS
                                *               C(XT) = TBLOCK-ADDRESS
                                *       ENTER BY
                                *               TSX 0,$CAUS
                                *               ARG FILE-REFERENCE-NUMBER OF EVENT
                                *               ARG NUMBER
                                *               ARG PASSED-FILE-REFERENCE
                                *               ARG ACCESSES-ON-PASSED-ITEMS
                                *               ARG STATE
                                *               ARG MESSAGE
                                *       RETURNS TO FIRST LOC AFTER THE MACRO EXPANSION
                                *       RETURNS WITH
                                *               C(XJ) = JBLOCK ADDRESS
                                *               C(XT) = TBLOCK-ADDRESS
                                *               C(XL) = RESTART ADDRESS
                                *       USES LOCAL TEMPORARY ONLY
                                *
END OF BINARY CARD IOS00020
   000767  005341 7400 00       CAUSE   STX     0,CAUST         POINTER TO ARGUMENT LIST
   000770  000006 0200 03               ADLX    0,6,DU          RESTART ADDRESS
                        000771          SETUP
   000771  000510 7170 00               XED     $SETUP
   000772  005341 2220 57               LDX     2,CAUST,IDC     LOAD FILE REFERENCE
   000773  005341 2230 57               LDX     3,CAUST,IDC     NUMBER
   000774  005341 2350 57               LDA     CAUST,IDC       STATE
   000775  005341 2360 57               LDQ     CAUST,IDC       MESSAGE
   000776  005341 2200 57               LDX     0,CAUST,IDC     LOAD FRN
```

CAUSE MACRO

```
000777  005341 2270 57      2104          LDX     7,CAUST,IDC      LOAD ACCESSES                  590
001000  000000 6260 10      2105          EAX     6,0,0            MOVE FRN TO X6                 600
001001  000047 2200 03      2106          LDX     0,,CAUSE,DU      MME NUMBER                     610
                   001002   2107          CKPT                     CHECKPOINT                     620
001002  000474 7170 00                    XED     X$CKPT
001003  000000 0010 00      2108          MME                      CAUSE                          630
                   001004   2109          EXIT                                                    640
001004  003074 7100 00                    TRA     $EXIT
                            2110 *                                                                650
                   005341   2111          USE     STORE                                           660
005341  000000 0000 20      2112 CAUST    ARG     0,*              POINTER TO ARGUMENT LIST       670
                   001005   2113          USE     PREVIOUS                                        680
                            2114 *$*      DISK    OPSCEM                                          690
```

OPEN SCRATCH EVENT MACRO

```
                  001005     2116        USE     CODE
                             2117        HEAD
                             2118 *
                             2119 *
                             2120 *                             OPSCE
                             2121 *
                             2122 OPSCE  MACRO   TIMLIM,MODE,MAXLEN
                             2123        TSX     0,$OPSCE
                             2124        ARG     #1              TIME LIMIT
                             2125        ARG     #2              MODE
                             2126        ARG     #3              MAXIMUM QUEUE LENGTH
                             2127        ENDM    OPSCE
                             2128 *
                             2129 *
                             2130 *              OPEN SCRATCH EVENT --SUBROUTINE
                             2131 *
                             2132 *      THIS SUBROUTINE IS CALLED BY THE  OPSCE  MACRO.  IT OPENS A
                             2133 *      SCRATCH EVENT.
                             2134 *
                             2135 *      CALL WITH
                             2136 *              C(XT) = TBLOCK ADDRESS
                             2137 *              C(XJ) = JBLOCK ADDRESS
                             2138 *      ENTER BY
                             2139 *              TSX 0,$OPSCE
                             2140 *              ARG TIME-LIMIT
                             2141 *              ARG MODE
                             2142 *              ARG MAX-QUEUE-LENGTH
                             2143 *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                             2144 *      RETURNS WITH
                             2145 *              C(XT) = TBLOCK ADDRESS
                             2146 *              C(XJ) = JBLOCK ADDRESS
                             2147 *              C(XL) = RESTART ADDRESS
                             2148 *      USES ONLY LOCAL TEMPORARY
                             2149 *
 001005  005342 7400 00      2150 OPSCE  STX     0,OPSET         POINTER TO ARGUEMNT LIST
 001006  000003 0600 03      2151        ADX     0,3,DU          RESTART ADDRESS
                  001007     2152        SETUP
 001007  000510 7170 00                  XED     $SETUP
 001010  005342 2230 57      2153        LDX     3,OPSET,IDC     LOAD TIME LIMIT
 001011  005342 2240 57      2154        LDX     4,OPSET,IDC     LOAD MODE
END OF BINARY CARD IOS00021
 001012  005342 2250 57      2155        LDX     5,OPSET,IDC     LOAD MAX QUEUE LENGTH
 001013  000052 2200 03      2156        LDX     0,.OPSCE,DU     LOAD MME NUMBER
                  001014     2157        CKPT                    CHECKPOINT
 001014  000474 7170 00                  XED     XSCKPT
 001015  000000 0010 00      2158        MME                     OPEN SCRATCH EVENT
                  001016     2159        EXIT
 001016  003074 7100 00                  TRA     $EXIT
                             2160 *
                  005342     2161        USE     STORE
```

OPEN SCRATCH EVENT MACRO

```
005342  000000 0000 20     2162 OPSET  ARG     0,*          POINTER TO ARGUMENT LIST            570
                  001017   2163        USE     PREVIOUS                                         580
                           2164 *$*    DISK    ACCTM                                            590
```

ACCOUNTING MACRO

```
                  001017                USE     CODE
                                        HEAD
                                 *
                                 *
                                 *                          ACCOUNTING
                                 *
                                 ACCT   MACRO   MODE,ACODE,DATA
                                        TSX     0,$ACCT
                                        ARG     #1          MODE
                                        ARG     #2          ACODE (ACODE/WORD NUMBER)
                                        ARG     #3          DATA
                                        ENDM    ACCT
                                 *
                                 *
                                 *              ACCOUNTING -- SUBROUTINE
                                 *
                                 *      THIS SUBROUTINE IS CALLED BY THE ACCT MACRO.  IT ISSUES
                                 *      THE COMMAND TO 1: REQUEST ACODE, OR 2: ALTER ACCOUNTING
                                 *      INFORMATION DEPENDING ON THE MODE (0 AND 1 RESPECTIVELY).
                                 *
                                 *      CALL WITH
                                 *              C(XT) = TBLOCK-ADDRESS
                                 *              C(XJ) = JBLOCK-ADDRESS
                                 *      ENTER BY
                                 *              TSX 0,$ACCT
                                 *              ARG MODE
                                 *              ARG ACODE
                                 *              ARG DATA
                                 *      RETURNS TO FIRST LOC AFTER THE MACRO EXPANSION
                                 *      RETURNS WITH
                                 *              C(XT) = TBLOCK-ADDRESS
                                 *              C(XJ) = JBLOCK-ADDRESS
                                 *              C(XL) = RESTART-ADDRESS
                                 *      USES LOCAL TEMPORARY ONLY
                                 *
001017  005343 7400 00           ACCT   STX     0,ACCTT     POINTER TO ARGUMENT LIST
001020  000003 0200 03                  ADLX    0,3,DU      RESTART ADDRESS
                  001021                SETUP
001021  000510 7170 00                  XED     $SETUP
001022  001017 2220 57                  LDX     2,ACCT,IDC  LOAD MODE
001023  005343 2350 57                  LDA     ACCTT,IDC   LOAD ACODE
001024  005343 2360 57                  LDQ     ACCTT,IDC   LOAD DATA
001025  000056 2200 03                  LDX     0,,RDME,DU  LOAD MME NUMBER
                  001026                CKPT                CHECKPOINT
001026  000474 7170 00                  XED     X$CKPT
001027  000000 0010 00                  MME                 ACCOUNTING CALL
                  001030                EXIT
001030  003074 7100 00                  TRA     $EXIT
                                 *
                  005343                USE     STORE
```

ACCOUNTING MACRO

```
    005343  00000C C000 20         2213 ACCTT  ARG      0,*              POINTER TO ARGUMENT LIST                580
                    001031         2214        USE      PREVIOUS                                                 590
                                   2215 *$*    DISK     CHECK                                                    600
```

CHECK MACRO

```
001031              USE     CODE
                    HEAD
           *
           *
           *                                 CHECK
           *
           *        THIS MACRO IS USED TO CHECK THE RESULT OF A TRAPPING MME.
           *        THE FIRST ARGUMENT IS THE ADDRESS TO WHICH TO TRANSFER IF THE
           *        STATUS RETURN IS ZERO.  THIS ARGUMENT MAY BE OMITTED, AND NO
           *        TEST WILL BE ASSEMBLED.  THE REMAINING ARGUMENTS COME IN PAIRS.
           *        THE FIRST OF THE PAIR IS A BOOLEAN PATTERN AGAINST WHICH A
           *        COMPARISON WILL BE MADE.  THE SECOND IS THE TRANSFER ADDRESS IN
           *        CASE OF A MATCH.  CURRENTLY THERE MAY BE ONLY 8 SUCH PAIRS.
           *
           *
           CHECK    MACRO   ZEROSTATADD,BOOLPAT,XFERADD,BOOLPAT,XFERADD,ETC.
                    LXL     0,TSSRW1,T        PICK UP LOGICAL STATUS
                    ANX     0,B$STMK,DU       ISOLATE STATUS
                    INE     '#1','
                    TZE     #1                ZERO STATUS TEST
                    INE     '#2','',23
                    CMPX    0,#2,DU
                    TZE     #3                FIRST PAIR OF TESTS
                    INE     '#4','',20
                    CMPX    0,#4,DU
                    TZE     #5                SECOND PAIR OF TESTS
                    INE     '#6','',17
                    CMPX    0,#6,DU
                    TZE     #7                THIRD PAIR OF TESTS
                    INE     '#8','',14
                    CMPX    0,#8,DU
                    TZE     #9                FOURTH PAIR OF TESTS
                    INE     '#10','',11
                    CMPX    0,#10,DU
                    TZE     #11               FIFTH PAIR OF TESTS
                    INE     '#12','',8
                    CMPX    0,#12,DU
                    TZE     #13               SIXTH PAIR OF TESTS
                    INE     '#14','',5
                    CMPX    0,#14,DU
                    TZE     #15               SEVENTH PAIR OF TESTS
                    INE     '#16','',2
                    CMPX    0,#16,DU
                    TZE     #17               EIGHTH PAIR OF TESTS
                    TRA     $ERROR            DIE ON UNEXPECTED RETURN
                    ENDM    CHECK
           *$*      DISK    QQ
```

```
001031           USE     CODE
                 HEAD    Q
           *
           *             QUEUE MANAGEMENT -- GENERAL INTRODUCTION
           *
           *     EACH QUEUE IN THE PROGRAM HAS A SIMILAR STRUCTURE.  A QUEUE
           *     CONSISTS OF A (POSSIBLY EMPTY) LINKED LIST OF BLOCK.  THE
           *     POINTERS POINT TO WORD 4 (Q$OFFST) OF A BLOCK.  THE LINK
           *     POINTERS ARE STORED IN WORD 3 (Q$LINK) OF A BLOCK.  THE WORD AT
           *     LOCATION Q$FIRST POINTS TO Q$OFFST OF THE FIRST BLOCK OF THE
           *     QUEUE.  THE LOCATION Q$LAST POINTS TO Q$OFFST OF THE LAST BLOCK
           *     OF THE QUEUE.  THE EMPTY QUEUE IS DENOTED BY THE WORD AT Q$LAST
           *     POINTING TO Q$FIRST+1.
           *
           *
           *                             QUEUE
           *
           *     THIS GENERATES A QBLOCK.  THIS STRUCTURE MUST AGREE
           *     WITH THE STRUCTURE DEFINED FOR QUEUE MANAGEMENT.
           *
           QUEUE MACRO   QBLOCK-LOCATION-SYMBOL,ASCII-NAME
                 USE     QSTOR             PUT ALL QUEUES CONTIGUOUS
                 EVEN                      FOR XED
           #1    BSS     0                 NAME OF QUEUE
                 ARG     $ERROR            FIRST
                 ARG     Q$#1+Q$FIRST+1    LAST
                 STX     0,Q$#1+Q$LAST,DI  *XADD
                 STX     0,Q$#1+Q$LAST
                 EAX     Q,Q$#1            ENQ
                 TSX     L,Q$ENQ
                 EAX     Q,Q$#1            DEQ
                 TSX     L,Q$DEQ
                 EAX     Q,Q$#1            INV
                 TSX     L,Q$INV
                 ARG     0                 BUSY
                 DEC     0                 MAX
                 DEC     0                 AVAIL
                 DEC     0                 SPARE1
                 DEC     0                 SPARE2
                 UASCI   1,#1              ABBREVIATION
                 ENDM    QUEUE
           *$*   DISK    ENQ
```

```
                     001031    2308           USE      CODE
                               2309           HEAD     Q
                               2310 *
                               2311 *
                               2312 *                                ENQ
                               2313 *
                               2314 *         ENQ SUSPENDS A TASK UNTIL THE SPECIFIED QUEUE CAN BE MADE
                               2315 *         AVAILABLE.
                               2316 *
                               2317 ENQ       MACRO    QADDRESS
                               2318           XED      Q$#1+Q$XENQ
                               2319           ENDM     ENQ
                               2320 *
                               2321 *                  ENQ -- SUBROUTINE TO SERIALIZE RESOURCE USE
                               2322 *
                               2323 *         THIS SUBPROGRAM RETURNS IMMEDIATELY IF THERE IS NO NEED
                               2324 *         TO QUEUE.  IF NECESSARY TO DO SO, IT SUSPENDS EXECUTION OF
                               2325 *         THE CURRENT TASK UNTIL A WAKE-UP IS GENERATED BY A DEQ FOR
                               2326 *         THIS QUEUE AND THE RESOURCE CAN BE ALLOCATED TO THIS TASK.
                               2327 *
                               2328 *         CALL WITH
                               2329 *                  C(XT) = TBLOCK-ADDRESS
                               2330 *                  C(XJ) = JBLOCK-ADDRESS
                               2331 *                  C(XQ) = QBLOCK-ADDRESS
                               2332 *         ENTER BY
                               2333 *                  TSX L,Q$ENQ
                               2334 *         CLOBBERS ALL BUT C(XT), C(XJ), AND C(XL)
                               2335 *         USES NO LOCAL TEMPORARIES
                               2336 *
                               2337           INHIB    SAVE,ON
                     001031    2338 ENQ       BSS      0
  001031  000014 0542 15       2339           AOS      AVAIL,Q        ONE MORE BLOCK CURRENTLY ON QUEUE
END OF BINARY CARD IOS00022
  001032  000014 2352 15       2340           LDA      AVAIL,Q        UPDATE MAX LENGTH OF QUEUE
  001033  000013 1152 15       2341           CMPA     MAX,Q          .
  001034  001036 6022 00       2342           TNC      *+2            .
  001035  000013 7552 15       2343           STA      MAX,Q          .
  001036  000012 2342 15       2344           SZN      BUSY,Q         IS THIS QUEUE BUSY?
  001037  001053 6012 00       2345           TNZ      ENQ1           YES, WILL HAVE TO QUEUE FOR IT
  001040  000012 7412 15       2346           STX      T,BUSY,Q       NO, MARK WHO IS RESPONSIBLE
                               2347           INHIB    RESTORE
                     001041    2348           BUGA
                     525201         BUGBUG SET      BUGBUG+1
  001041  525201 2350 03                      LDA      BUGBUG,DU
  001042  525201 2750 07                      ORA      BUGBUG,DL
                     001043    2349           BUGQ
                     525202         BUGBUG SET      BUGBUG+1
  001043  525202 2360 03                      LDQ      BUGBUG,DU
  001044  525202 2760 07                      ORQ      BUGBUG,DL
                     001045    2350           BUGXR    (0,X,Y,Z,Q)
```

```
                       525203          BUGBUG SET     BUGBUG+1
    001045  525203 2200 03                    LDX     0,BUGBUG,DU
    001046  525203 2220 03                    LDX     X,BUGBUG,DU
    001047  525203 2230 03                    LDX     Y,BUGBUG,DU
    001050  525203 2240 03                    LDX     Z,BUGBUG,DU
    001051  525203 2250 03                    LDX     Q,BUGBUG,DU
    001052  000000 7100 17        2351        TRA     0,L                RETURN TO CALLER                   540
    001053  000004 7470 11        2352 ENQ1   STX     L,T$TRA,T          SAVE RESTART ADDRESS               550
    001054  000004 6200 11        2353        EAX     0,OFFST,T          STORE POINTER WITH OFFSET          560
    001055  777777 6000 00        2354        TZE     $ERROR             ***DBG                             570
    001056  777777 6040 00        2355        TMI     $ERROR             ***DBG                             580
    001057  000002 7170 15        2356        XED     XADD,Q             IN QUEUE LINKED LIST OF BLOCKS     590
                       001060     2357        EXIT                                                          600
 END OF BINARY CARD IOS00023
    001060  003074 7100 00                    TRA     $EXIT
                                  2358 *$*    DISK    ENQF                                                  610
```

```
               001061     2360          USE     CODE
                          2361          HEAD    Q
                          2362 *
                          2363 *
                          2364 *                          ENQF
                          2365 *
                          2366 *        THIS SUBROUTINE PUTS A TRAP BLOCK ON THE FRONT OF A QUEUE.
                          2367 *
                          2368 *        CALL WITH
                          2369 *                C(XT) = TRAP-BLOCK-ADDRESS
                          2370 *                C(XQ) = QUEUE-ADDRESS
                          2371 *        ENTER BY
                          2372 *                TSX L,Q$ENQF
                          2373 *        CLOBBERS C(XO), C(XX)
                          2374 *
                          2375          INHIB   SAVE,ON
               001061     2376 ENQF     BSS     0
001061  000012 2342 15    2377          SZN     BUSY,Q          IS QUEUE BUSY?
001062  001067 6012 00    2378          TNZ     ENQF0           YES, GET ON FRONT OF IT
001063  000012 7412 15    2379          STX     T,BUSY,Q        NO, SEIZE THE RESOURCE
               001064     2380          BUGXR   (0,X)
               525204          BUGBUG SET     BUGBUG+1
001064  525204 2202 03                  LDX     0,BUGBUG,DU
001065  525204 2222 03                  LDX     X,BUGBUG,DU
001066  000000 7102 17    2381          TRA     0,L             AND RETURN IMMEDIATELY
                          2382 *
                          2383 *        CHECK IF QUEUE IS EMPTY
                          2384 *
               001067     2385 ENQF0    BSS     0
001067  000004 7472 11    2386          STX     L,T$TRA,T       SAVE RESTART ADDRESS
001070  000004 6202 11    2387          EAX     0,Q$OFFST,T     GET ADDRESS OF FOURTH WORD OF BLOCK
001071  000001 6222 15    2388          EAX     X,Q$FIRST+1,Q   IS THE QUEUE
001072  000001 1022 15    2389          CMPX    X,Q$LAST,Q      EMPTY?
001073  001100 6002 00    2390          TZE     ENQF2           YES, MUST HANDLE AS SPECIAL CASE
                          2391 *
                          2392 *        GET ON FRONT OF NON-EMPTY QUEUE
                          2393 *
               001074     2394 ENQF1    BSS     0
001074  000000 2222 15    2395          LDX     X,Q$FIRST,Q     POINT TO PRESENT FIRST BLOCK ON QUEUE
001075  000000 7402 15    2396          STX     0,Q$FIRST,Q     PLACE THIS BLOCK FIRST
001076  000003 7422 11    2397          STX     X,Q$LINK,T      AND MOVE OLD ONE DOWN
               001077     2398          EXIT
001077  003074 7102 00                  TRA     $EXIT
                          2399 *
                          2400 *        GET ON FRONT OF EMPTY QUEUE BY MAKING Q$FIRST AND Q$LAST
                          2401 *        POINT TO THIS BLOCK.
                          2402 *
               001100     2403 ENQF2    BSS     0
001100  000000 7402 15    2404          STX     0,Q$FIRST,Q     MAKE IT THE FIRST BLOCK
001101  000001 7402 15    2405          STX     0,Q$LAST,Q      AND ALSO THE LAST
```

```
                     001102     2406        EXIT                        EXIT         570
001102  003074 7102 00                      TRA      $EXIT
                                2407        INHIB    RESTORE                         580
                                2408 *$*    DISK     DEQ                             590
```

```
                      001103          USE     CODE
                                      HEAD    Q
                              *
                              *                               DEQ
                              *
                              *       DEQ RELEASES A SERIALLY REUSABLE RESOURCE SO THAT OTHER
                              *       PROCESSES MAY USE IT.  OTHER TASKS ARE AUTOMATICALLY INITIATED
                              *       BY PUTTING THEM ON THE Q$TASK QUEUE.
                              *
                              DEQ     MACRO   QADDRESS
                                      XED     Q$#1+Q$XDEQ
                                      ENDM    DEQ
                              *
                              *               DEQ -- SUBROUTINE TO WAKE-UP WAITING TASKS
                              *
                              *       THIS SUBROUTINE CHECKS TO SEE IF ANY OTHER TASKS ARE ASLEEP
                              *       IN THE QUEUE.  IF THE QUEUE IS NOT EMPTY, A TBLOCK IS TAKEN
                              *       FROM IT AND PUT ON THE Q$TASK QUEUE TO WAKE IT UP.  THIS SUB-
                              *       PROGRAM ALWAYS RETURNS TO THE CALLER IMMEDIATELY.
                              *
                              *       CALL WITH
                              *               C(XQ) = QBLOCK-ADDRESS
                              *               C(XT) = TBLOCK-ADDRESS
                              *               C(XJ) = JBLOCK-ADDRESS
                              *       ENTER BY
                              *               TSX L,Q$DEQ
                              *       DESTROYS C(XO), C(XX)
                              *       USES NO LOCAL TEMPORARIES
                              *
                                      INHIB   SAVE,ON
                      001103  DEQ     BSS     0
                      001103          DECRM   (AVAIL,Q)       ONE LESS BLOCK ON QUEUE
 001103  000001 3362 07               LCQ     1,DL
END OF BINARY CARD IOS00024
 001104  000014 0562 15               ASQ     AVAIL,Q
 001105  000012 2342 15               SZN     BUSY,Q          IF QUEUE IS NOT BUSY
 001106  777777 6002 00               TZE     $ERROR          WE ARE IN BAD TROUBLE
 001107  000012 4502 15               STZ     BUSY,Q          MARK QUEUE NOT BUSY
 001110  000001 6202 15               EAX     O,FIRST+1,Q     ADDRESS OF FIRST ELEMENT
 001111  000001 1002 15               CMPX    O,LAST,Q        DOES LAST POINT TO IT?
 001112  000000 6002 17               TZE     O,L             YES, QUEUE EMPTY -- RETURN
 001113  000000 2202 15               LDX     O,FIRST,Q       GET OFFSET POINTER TO BLOCK
 001114  777777 6002 00               TZE     $ERROR          ***BLEWIT
 001115  777777 6042 00               TMI     $ERROR          ***DBG
 001116  777774 6222 10               EAX     X,-OFFST,O      RELATE TO THE BEGINNING OF BLOCK
 001117  000012 7402 15               STX     O,BUSY,Q        REMEMBER WHO IS RESPONSIBLE
 001120  000003 2222 12               LDX     X,LINK,X        GET NEXT ELEMENT ON QUEUE
 001121  000000 7422 15               STX     X,FIRST,Q       NOW MAKE IT FIRST
 001122  005162 7172 00               XED     XADD+TASK       ADD TO TASK QUEUE
 001123  000001 1002 15               CMPX    O,LAST,Q        LAST BLOCK ON QUEUE?
```

```
         Q                    QUEUE MANAGEMENT -- DEQ

001124  000000 6012 17     2457          TNZ     0,L              NO, RETURN                              580
001125  000001 6202 15     2458          EAX     0,FIRST+1,Q      YES, SET UP QUEUE TO APPEAR EMPTY       590
001126  000001 7402 15     2459          STX     0,LAST,Q         BY MAKING LAST POINT TO FIRST+1         600
                           2460          INHIB   RESTORE                                                  610
001127  000000 7100 17     2461          TRA     0,L              RETURN                                  620
                           2462 *$*      DISK    INVERT                                                   630
```

```
                     001130          USE     CODE
                                     HEAD    Q
                                *
                                *
                                *                              INV
                                *
                                *    THIS ROUTINE INVERTS THE ORDER OF THE TOP TWO ITEMS ON
                                *    THE SPECIFIED QUEUE.
                                *
                                INV  MACRO   QADDRESS
                                     XED     Q$#1+Q$XINV
                                     ENDM    INV
                                *
                                *            INV -- SUBROUTINE TO SWAP THE TOP TWO ITEMS ON A LIST.
                                *
                                *    THIS SUBROUTINE TRIES TO SWITCH THE ORDER OF THE TOP TWO ITEMS
                                *    ON THE NAMED QUEUE.  IF THE NAMED QUEUE IS EMPTY OF HAS ONLY
                                *    A SINGLE ITEM, NO ACTION IS TAKEN; OTHERWISE THE TWO ARE
                                *    ARE INVERTED.
                                *
                                *    CALL WITH
                                *            C(XT) = TCB
                                *            C(XJ) = JCB
                                *            C(XQ) = QUEUE-ADDRESS
                                *    ENTER BY
                                *            TSX L,Q$INV
                                *    CLOBBERS C(XO), C(XX), C(XY)
                                *
                                     INHIB   SAVE,ON          PREPARE TO FOOL WITH QUEUES
                     001130     INV  BSS     0
                                *
                                *    CHECK IF QUEUE IS EMPTY
                                *
 001130  000001 6222 15              EAX     X,FIRST+1,Q      DOES FIRST POINT TO LAST?
END OF BINARY CARD IOS00025
 001131  000001 1022 15              CMPX    X,LAST,Q         EMPTY?
 001132  001146 6002 00              TZE     INV1             IF SO, EXIT
                                *
                                *    CHECK TO SEE IF AT LEAST TWO ITEMS
                                *
 001133  000000 2222 15              LDX     X,FIRST,Q        DOES FIRST
 001134  000001 1022 15              CMPX    X,LAST,Q         POINT TO LAST?
 001135  001146 6002 00              TZE     INV1             IF SO, EXIT
                                *
                                *    SWAP THE TOP TWO
                                *
 001136  777777 2232 12              LDX     Y,-OFFST+LINK,X *C(X) = TOP
                                                              C(Y) = SECOND
 001137  777777 2202 13              LDX     O,-OFFST+LINK,Y *SAVE PTR TO THIRD
 001140  000000 7432 15              STX     Y,FIRST,Q        MAKE SECOND FIRST
```

```
001141  777777 7422 13         2513        STX     X,-OFFST+LINK,Y *MAKE FIRST SECOND                     600
001142  777777 7402 12         2514        STX     0,-OFFST+LINK,X *MAKE SECOND POINT TO THIRD            610
001143  000001 1032 15         2515        CMPX    Y,LAST,Q        WAS SECOND REALLY LAST?                620
001144  001146 6012 00         2516        TNZ     *+2             NO                                     630
001145  000001 7422 15         2517        STX     X,LAST,Q        YES, SWITCH IT                         640
                               2518 *                                                                 650
                               2519 *      DONE, BUG REGISTERS AND RETURN                             660
                               2520 *                                                                 670
                     001146    2521 INV1   BSS     0                                                  680
                     001146    2522        BUGXR   (0,X,Y)                                            690
                     525205         BUGBUG SET     BUGBUG+1
001146  525205 2202 03                     LDX     0,BUGBUG,DU
001147  525205 2222 03                     LDX     X,BUGBUG,DU
001150  525205 2232 03                     LDX     Y,BUGBUG,DU
001151  000000 7102 17         2523        TRA     0,L             RETURN TO CALLER                       700
                               2524        INHIB   RESTORE                                            710
                               2525 *$*    DISK    BRANCH                                             720
```

```
001152          USE     CODE
                HEAD    Q
       *
       *
       *                                BRANCH MACRO
       *
       *        THIS MACRO CREATES AN ASYNCHRONOUS TASK TO BE PREFORMED AT A
       *        LATER TIME.  IT CAN GIVE THE CREATED TASK THE CURRENT TBLOCK
       *        OR GIVE IT A NEW TBLOCK.  EITHER WAY FOUR PARAMETERS MAY BE
       *        BETWEEN THE TWO TBLOCKS.
       *
       *        PAST INFORMATION IS PLACED IN TEMP1 THRU TEMP4 OF
       *        THE TASK PLACED ON THE Q$TASK QUEUE.
       *
       *        CALLS
       *                T$GETT
       *        CLOBBERS C(XX), C(X0)
       *
       *
       BRANCH MACRO     PASS,XFER ADD,C(TEMP1),C(TEMP2),C(TEMP3),C(TEMP4)
                TSX     0,T$GETT          GET A NEW TBLOCK
                EAX     X,0,T             C(XX) POINTS TO NEW TBLOCK
                LDX     T,T$LINK,X        C(XT) POINTS TO OLD TBLOCK
                INE     '#3','',2
                LDQ     #3                SAVE FIRST PARAMETER
                STQ     T$TEMP1,X
                INE     '#4','',2
                LDQ     #4                SAVE SECOND PARAMETER
                STQ     T$TEMP2,X
                INE     '#5','',2
                LDQ     #5                SAVE THIRD PARAMETER
                STQ     T$TEMP3,X
                INE     '#6','',2
                LDQ     #6                SAVE FOURTH PARAMETER
                STQ     T$TEMP4,X
                INE     '#1','PASS',1     T IS THE BLOCK THAT IS PASSED
                EAX     T,0,X             PASS NEW BLOCK
                EAX     0,#2              POINT TO TRANSFER ADDRESS
                STX     0,T$TRA,T         INTO QUEUE BLOCK
                EAX     0,Q$OFFST,T       PREPARE TO QUEUE
                XED     Q$XADD+Q$TASK     GET ON THE TASK QUEUE
                IFE     '#1','PASS',1     WHICH BLOCK TO WE GIVE BACK AS CURRENT
                EAX     T,0,X             GIVE BACK NEW BLOCK
                INE     '#1','PASS',1
                LDX     T,T$LINK,X        NO, GIVE BACK OLD BLOCK
                BUGXR   (0,X)
                ENDM    BRANCH
       *$*      DISK    OP
```

```
        Q                                      QUEUES -- Q$OP00 QUEUE

                                 2576       HEAD    Q                                                     110
                                 2577 *                                                                   120
                                 2578 *                                                                   130
                                 2579 *                               Q$OP00 QUEUE                        140
                                 2580 *                                                                   150
                                 2581 *     SINCE THE LOGGING DEVICE IS A SINGLE RESOURCE, TASKS MUST     160
                                 2582 *     QUEUE FOR IT.  HENCE THIS QUEUE HOLDS THOSE TASKS WAITING     170
                                 2583 *     TO LOG A MESSAGE TO THE OPERATOR'S CONSOLE TTY.               180
                                 2584 *                                                                   190
                       001152    2585       QUEUE   OP00                                                  200
                       005140               USE     QSTOR
                       005140               EVEN
                       005140        OP00   BSS     0
  005140  777777 0000 00                    ARG     $ERROR
  005141  005141 0000 00                    ARG     Q$OP00+Q$FIRST+1
  005142  005141 7400 54                    STX     0,Q$OP00+Q$LAST,DI
  005143  005141 7400 00                    STX     0,Q$OP00+Q$LAST
END OF BINARY CARD IOS00026
  005144  005140 6250 00                    EAX     Q,Q$OP00
  005145  001031 7070 00                    TSX     L,Q$ENQ
  005146  005140 6250 00                    EAX     Q,Q$OP00
  005147  001103 7070 00                    TSX     L,Q$DEQ
  005150  005140 6250 00                    EAX     Q,Q$OP00
  005151  001130 7070 00                    TSX     L,Q$INV
  005152  000000 0000 00                    ARG     0
  005153  000000000000                      DEC     0
  005154  000000000000                      DEC     0
  005155  000000000000                      DEC     0
  005156  000000000000                      DEC     0
  005157  117120060060                      UASCI   1,OP00            ABBREVIATION
                       005140    2586 OP    EQU     OP00         ***ASSUME ONLY ONE OP CONSOLE EVER; (REASONABLE)
                                 2587 *$*   DISK    01                                                    220
```

```
                                2589          HEAD    Q
                                2590 *
                                2591 *
                                2592 *                                  Q$TASK QUEUE
                                2593 *
                                2594 *        THIS QUEUE IS USED TO SCHEDULE THE ACTIVITY OF THE PROCESSOR
                                2595 *        ENTRIES ARE MADE BY THE USE OF THE NORMAL XED Q$ADD
                                2596 *        FEATURE AND ALSO BY AN XED CHAIN INSTIGATED BY THE ACTION OF
                                2597 *        A TRAP BEING SPRUNG BY THE EXECUTIVE.  ENTRIES ARE REMOVED BY
                                2598 *        A SPECIAL PROGRAM MODULE WHICH IS CALLED BY THE EXIT MACRO
                                2599 *        WHENEVER THE PROCESSOR IS FREE TO WORK ON A NEW TASK.
                                2600 *
                      005160    2601          QUEUE   TASK
                      005160                  USE     QSTOR
                      005160                  EVEN
                      005160            TASK  BSS     0
   005160  777777 0000 00                     ARG     $ERROR
   005161  005161 0000 00                     ARG     Q$TASK+Q$FIRST+1
   005162  005161 7400 54                     STX     0,Q$TASK+Q$LAST,DI
   005163  005161 7400 00                     STX     0,Q$TASK+Q$LAST
   005164  005160 6250 00                     EAX     Q,Q$TASK
   005165  001031 7070 00                     TSX     L,Q$ENQ
   005166  005160 6250 00                     EAX     Q,Q$TASK
   005167  001103 7070 00                     TSX     L,Q$DEQ
   005170  005160 6250 00                     EAX     Q,Q$TASK
 END OF BINARY CARD IOS00027
   005171  001130 7070 00                     TSX     L,Q$INV
   005172  000000 0000 00                     ARG     0
   005173  000000000000                       DEC     0
   005174  000000000000                       DEC     0
   005175  000000000000                       DEC     0
   005176  000000000000                       DEC     0
   005177  124101123113                       UASCI   1,TASK              ABBREVIATION
```

```
            0                          QUEUES -- Q$CORE QUEUE
                                 2603          HEAD    Q
                                 2604 *
                                 2605 *
                                 2606 *                            Q$CORE QUEUE
                                 2607 *
                                 2608 *        THIS QUEUE HOLDS THOSE TASKS WHICH REQUIRE MEMORY ALLOCATIONS
                                 2609 *        WHEN A PRIOR MEMORY REQUEST IS IN OPERATION.  THIS IS NOT
                                 2610 *        TO BE CONFUSED WITH THE THE FREE MEMORY LIST.
                                 2611 *
                     005200      2612          QUEUE   CORE
                     005200                    USE     QSTOR
                     005200                    EVEN
                     005200             CORE   BSS     0
 005200  777777 0000 00                        ARG     $ERROR
 005201  005201 0000 00                        ARG     Q$CORE+Q$FIRST+1
 005202  005201 7400 54                        STX     0,Q$CORE+Q$LAST,DI
 005203  005201 7400 00                        STX     0,Q$CORE+Q$LAST
 005204  005200 6250 00                        EAX     0,Q$CORE
 005205  001031 7070 00                        TSX     L,Q$ENQ
 005206  005200 6250 00                        EAX     0,Q$CORE
 005207  001103 7070 00                        TSX     L,Q$DEQ
 005210  005200 6250 00                        EAX     0,Q$CORE
 005211  001130 7070 00                        TSX     L,Q$INV
 005212  000000 0000 00                        ARG     0
 005213  000000000000                          DEC     0
 005214  000000000000                          DEC     0
 005215  000000000000                          DEC     0
END OF BINARY CARD IOS00028
 005216  000000000000                          DEC     0
 005217  103117122105                          UASCI   1,CORE           ABBREVIATION
```

```
                                        HEAD    Q
                                  *
                                  *
                                  *                               Q$INP1 QUEUE
                                  *
                                  *     THIS IS THE GENERAL JOB INPUT QUEUE.  JOBS REQUESTING ONE
                                  *     OR MORE RESOURCES ARE PLACED ON THIS LIST.  NOTE THAT THIS IS A
                                  *     LIST AS OPPOSED TO A TRUE QUEUE.
                                  *
                     005220             QUEUE   INP1
                     005220             USE     QSTOR
                     005220             EVEN
                     005220       INP1  BSS     0
005220  777777 0000 00                  ARG     $ERROR
005221  005221 0000 00                  ARG     Q$INP1+Q$FIRST+1
005222  005221 7400 54                  STX     0,Q$INP1+Q$LAST,DI
005223  005221 7400 00                  STX     0,Q$INP1+Q$LAST
005224  005220 6250 00                  EAX     Q,Q$INP1
005225  001031 7070 00                  TSX     L,Q$ENQ
005226  005220 6250 00                  EAX     Q,Q$INP1
005227  001103 7070 00                  TSX     L,Q$DEQ
005230  005220 6250 00                  EAX     Q,Q$INP1
005231  001130 7070 00                  TSX     L,Q$INV
005232  000000 0000 00                  ARG     0
005233  000000000000                    DEC     0
005234  000000000000                    DEC     0
005235  000000000000                    DEC     0
005236  000000000000                    DEC     0
005237  111116120061                    UASCI   1,INP1           ABBREVIATION
```

```
                                   2625        HEAD    Q                                                  470
                                   2626 *                                                                 480
                                   2627 *                                                                 490
                                   2628 *                                   Q$INP2 QUEUE                  500
                                   2629 *                                                                 510
                                   2630 *      THIS IS A SPECIAL LIST (REALLY A LIST, NOT A QUEUE).  HERE GOES  520
                                   2631 *      ALL SHORT PRINTER JOBS.  A SHORT PRINTER JOB IS ONE THAT REQUIRES530
                                   2632 *      AN ARBITRARY NUMBER OF PAGES TO BE PRINTED OR LESS.          540
                                   2633 *                                                                 550
                        005240     2634        QUEUE   INP2                                               560
                        005240                 USE     QSTOR
                        005240                 EVEN
                        005240         INP2    BSS     0
   005240  777777 0000 00                      ARG     $ERROR
   005241  005241 0000 00                      ARG     Q$INP2+Q$FIRST+1
 END OF BINARY CARD IOS00029
   005242  005241 7400 54                      STX     0,Q$INP2+Q$LAST,DI
   005243  005241 7400 00                      STX     0,Q$INP2+Q$LAST
   005244  005240 6250 00                      EAX     Q,Q$INP2
   005245  001031 7070 00                      TSX     L,Q$ENQ
   005246  005240 6250 00                      EAX     Q,Q$INP2
   005247  001103 7070 00                      TSX     L,Q$DEQ
   005250  005240 6250 00                      EAX     Q,Q$INP2
   005251  001130 7070 00                      TSX     L,Q$INV
   005252  000000 0000 00                      ARG     0
   005253  000000000000                        DEC     0
   005254  000000000000                        DEC     0
   005255  000000000000                        DEC     0
   005256  000000000000                        DEC     0
   005257  111116120062                        UASCI   1,INP2            ABBREVIATION
```

```
                                       2636          HEAD     Q                                                            580
                                       2637 *                                                                              590
                                       2638 *                                                                              600
                                       2639 *                                        Q$INP3 QUEUE                          610
                                       2640 *                                                                              620
                                       2641 *        THIS IS A SPECIAL LIST (REALLY, A LIST; NOT A QUEUE).  HERE           630
                                       2642 *        GOES ALL MEDIUM SIZE PRINTER JOBS.  A MEDIUM PRINTER JOB IS ONE       640
                                       2643 *        THAT REQUIRES AN ARBITRARY NUMBER OF PAGES TO BE PRINTED WHICH        650
                                       2644 *        IS BETWEEN TWO LIMITS (SMALL AND LARGE OF COURSE).                    660
                                       2645 *                                                                              670
                             005260    2646          QUEUE    INP3                                                         680
                             005260                  USE      QSTOR
                             005260                  EVEN
                             005260           INP3   BSS      0
     005260  777777 0000 00                          ARG      $ERROR
     005261  005261 0000 00                          ARG      Q$INP3+Q$FIRST+1
     005262  005261 7400 54                          STX      0,Q$INP3+Q$LAST,DI
     005263  005261 7400 00                          STX      0,Q$INP3+Q$LAST
     005264  005260 6250 00                          EAX      Q,Q$INP3
     005265  001031 7070 00                          TSX      L,Q$ENQ
     005266  005260 6250 00                          EAX      Q,Q$INP3
   END OF BINARY CARD IOS00030
     005267  001103 7070 00                          TSX      L,Q$DEQ
     005270  005260 6250 00                          EAX      Q,Q$INP3
     005271  001130 7070 00                          TSX      L,Q$INV
     005272  000000 0000 00                          ARG      0
     005273  000000000000                            DEC      0
     005274  000000000000                            DEC      0
     005275  000000000000                            DEC      0
     005276  000000000000                            DEC      0
     005277  111116120063                            UASCI    1,INP3                ABBREVIATION
                                       2647 *$*      DISK     GETC                                                         690
```

```
                     001152            USE    CODE
                                       HEAD   R
                                 *
                                 *                               GENERAL INTRODUCTION
                                 *
                                 *     BELOW IS THE FREE MEMORY LIST.  THE LIST CONSISTS OF A
                                 *     POSSIBLY EMPTY LINKED LIST OF BLOCKS.  THE FORWARD/BACKWARD
                                 *     POINTERS OF A BLOCK POINT TO THE FIRST WORD OF THE SUCCEEDING/
                                 *     PRECEEDING BLOCKS, RESPECTIVELY. THE LINK POINTERS ARE
                                 *     STORED IN WORDS 0 AND 1, RESPECTIVELY, OF THE BLOCK.
                                 *     HENCE THE MINIMAL SIZE OF A BLOCK IS TWO (2) WORDS.  THE
                                 *     TOTAL LENGTH OF THE BLOCK IS ALSO KEPT IN WORD 0.  BY
                                 *     DESIGN CONVENTIONS, THE POINTERS ARE UPPER HALF QUANTITIES
                                 *     AND THE LENGTH IS A LOWER HALF QUANTITY.  THE EMPTY LIST
                                 *     IS DENOTED BY THE FORWARD LINK OF THE 'FIRST' POINTING TO
                                 *     'LAST' AND BY THE BACKWARD LINK OF 'LAST' POINTING TO 'FIRST'.
                                 *
                                 *
                                 *     NOTE THAT ALL ALLOCATIONS ARE DONE IN MULTIPLES OF EIGHT.
                                 *     A BLOCK MUST BE DE-ALLOCATED AS ONE ENTITY.  NO PARTIAL RELEASES
                                 *     ARE ALLOWED.
                                 *
                                 *             FREE MEMORY LIST
                                 *
                     005344            USE    STORE
005344  006400 000000            FIRST ZERO   $NEXTF,0          FORWARD LINK/ LENGTH OF BLOCK
005345  000000 000000                  ZERO   0,                BACKWARD LINK/ <NOT USED>
                                 *
                                 *
005346  000000 000000            LAST  ZERO   0,0               FORWARD LINK/ LENGTH OF BLOCK
005347  006400 000000                  ZERO   $NEXTB,           BACKWARD LINK/ <NOT USED>
                     001152            USE    PREVIOUS
```

```
001152          USE     CODE
                HEAD    R
       *
       *
       *                                   MACROS
       *
       *
       *        THE FOLLOWING MACROS ARE USED TO ALLOCATE/DEALLOCATE BLOCKS
       *        OF CORE.  ONLY INDEX REGISTER J IS GUARANTEED ACCROSS THESE
       *        CALLS.
       *
       *                                   GETC MACRO
       *
       *        CALL WITH
       *                C(AL) = NUMBER-OF-WORDS-REQUESTED
       *        ENTER BY
       *                TSX L,R$GETC
       *        RETURNS TO 0,L
       *        RETURNS WITH
       *                C(AU) = BUFFER-ADDRESS
       *                C(AL) = BUFFER-LENGTH
       *
       GETC     MACRO   WORD-COUNT-ADDRESS/ 'A'
                INE     '#1','A'
                LDA     #1
                TSX     L,R$GETC
                ENDM    GETC
       *
       *
       *                                   RELC MACRO
       *
       *        CALL WITH
       *                C(AU) = BUFFER-ADDRESS
       *                C(AL) = BUFFER-ADDRESS
       *        ENTER BY
       *                TSX L,R$RELC
       *        RETURNS TO 0,L
       *        RETURNS WITH
       *                DESTROYS C(A)
       *
       RELC     MACRO   RELEASE-ADDRESS&COUNT/ 'A'
                INE     '#1','A'
                LDA     #1
                TSX     L,R$RELC
                ENDM    RELC
       *$*      DISK    GETC1
```

```
                      001152   2729        USE     CODE                                        110
                               2730        HEAD    R                                           120
                               2731 *                                                          130
                               2732 *                                                          140
                               2733 *                              ALLOCATION                  150
                               2734 *                                                          160
                               2735 *      THIS SUBROUTINE REMOVES A BLOCK OF N WORDS FROM THE FREE       170
                               2736 *      MEMORY LIST.  IF THERE IS NO BLOCK OF N WORDS OR GREATER       180
                               2737 *      ON THE FREE LIST, THERE A REQUEST FOR MORE MEMORY IS MADE,     190
                               2738 *      AND THE PROCESS IS REPEATED.  SINCE MEMORY REQUESTS ARE        200
                               2739 *      OF THE TRAPPING MME BRAND, IT IS NECESSARY TO FIRST            210
                               2740 *      CHECK TO SEE IF A MEMORY REQUEST IS CURRENTLY IN OPERATION.    220
                               2741 *      IF SO, THEN THIS REQUEST IS PUT TO SLEEP BY QUEUEING IT        230
                               2742 *      ON THE Q$CORE QUEUE.  WHEN THE MEMORY REQUEST COMPLETES        240
                               2743 *      THE NEXT ITEM (IF ANY) ON Q$CORE IS AWAKENED.  CORE ALLO-      250
                               2744 *      CATION IS ON A FIRST FIT BASIS.  THIS IS TO ALLOW HOLES        260
                               2745 *      TO FLOW TOWARD HIGHER MEMORY.                                  270
                               2746 *                                                          280
                               2747 *      CALL BY                                             290
                               2748 *              TSX L,R$GETC                                300
                               2749 *      CALL WITH                                           310
                               2750 *              C(AL) = NUMBER-OF-WORDS-REQUESTED           320
                               2751 *              C(T)  = TRAP BLOCK                          330
                               2752 *      CALLS                                               340
                               2753 *              ENQ CORE                                    350
                               2754 *              R$MORE                                      360
                               2755 *              R$RELC                                      370
                               2756 *      DEQ     CORE                                        380
                               2757 *      RETURNS WITH                                        390
                               2758 *              C(AU) = ADDRESS                             400
                               2759 *              C(AL) = NUMBER OF WORDS                     410
                               2760 *      EXIT TO 0,L                                         420
                               2761 *      PRESERVES ALL REGISTERS EXCEPT C(A)                 430
                               2762 *      USES TEMPORARIES T$TEM9 THRU T$TEM16                440
                               2763 *                                                          450
                      001152   2764        USE     CODE                                        460
                      001152   2765 GETC   BSS     0                                           470
    001152  000010 7530 11     2766        SREG    T$TEM16,T       SAVE ALL REGISTERS          480
                      001153   2767        ENQ     CORE            AND GET ON THE CORE QUEUE   490
    001153  005204 7170 00                 XED     Q$CORE+Q$XENQ
                      001154   2768 GETC1  BSS     0                                           500
    001154  000014 2350 11     2769        LDA     T$TEM12,T       RESTORE A AND               510
    001155  000007 0350 07     2770        ADLA    7,DL            ROUND UP LENGTH             520
    001156  777770 3750 07     2771        ANA     =0777770,DL     TO A MULTIPLE OF 8          530
    001157  777777 6000 00     2772        TZE     $ERROR          ***BLEWIT                   540
    001160  004001 1150 07     2773        CMPA    $RQMAX,DL       HOW MUCH IS REQUESTED       550
 END OF BINARY CARD IOS00031
    001161  777777 6030 00     2774        TRC     $ERROR          TOO MUCH!                   560
    001162  000014 7550 11     2775        STA     T$TEM12,T       SAVE NEW LENGTH             570
    001163  000000 6350 05     2776        EAA     0,AL            MOVE LENGTH TO AU           580
```

R CORE MANAGEMENT -- ALLOCATION

```
001164  000016 7550 11              STA    T$TEM10,T       SAVE FOR INDEX REGISTER OPERATIONS
                   001165     GETC3  BSS    0
001165  000014 2350 11              LDA    T$TEM12,T       GET BACK REQUEST
001166  005344 2240 00              LDX    Z,FIRST         GET PTR TO FIRST FREE BLOCK
                   001167     GETC6  BSS    0
001167  000302 5002 00              RPL    ,TNC,TZE        RUN DOWN LINKED LIST
001170  000000 1150 14              CMPA   0,Z                FOR A BLOCK BIG ENOUGH
001171  001176 6000 00              TZE    GETC7           FOUND A BLOCK THAT IS 'JUST RIGHT'
001172  001176 6020 00              TNC    GETC7           MORE THAT BIG ENOUGH
001173  000000 2240 14              LDX    Z,0,Z           GET BACK PTR OF WHERE WE LEFT OFF
001174  001167 6010 00              TNZ    GETC6           IF NOT AT END OF LIST, KEEP SEARCHING
001175  001241 7100 00              TRA    MORE            WON'T FIT ANYWHERE, GET MORE CORE
001176  000014 7440 11        GETC7  STX    Z,T$TEM12,T     SAVE PTR TO BLOCK TO RETURN TO CALLER
                              *
                              *       Z POINTS TO BLOCK TO DELINK
                              *
001177  000001 2230 14              LDX    Y,LINKB,Z       ASSIGN Y TO PREDECESSOR
001200  000000 2220 14              LDX    X,LINKF,Z       AND X TO SUCCESSORE
001201  000000 7200 14              LXL    0,LEN,Z         GET THE LENGTH OF THIS BLOCK
001202  000016 1200 11              SBLX   0,T$TEM10,T     MINUS THE AMOUNT REQUIRED
001203  001212 6000 00              TZE    GETC4           AH HA, BLOCK JUST FITS
001204  777777 6040 00              TMI    $ERROR          ***BLEWIT
001205  000016 0240 11              ADLX   Z,T$TEM10,T     SET Z TO START OF EXCESS OF BLOCK
001206  000000 4400 14              SXL    0,LEN,Z         SET THE LENGTH OF THE REMAINING BLOCK
END OF BINARY CARD IOS00032
001207  000000 7420 14              STX    X,LINKF,Z       SET FORWARD LINK OF REMAINING BLOCK
001210  000001 7440 12              STX    Z,LINKB,X        "  BACKWARD  "  "        "       "
001211  000000 6220 14              EAX    X,0,Z           COPY Z INTO X FOR FUDGE
001212  000000 7420 13        GETC4  STX    X,LINKF,Y       SET FORWARD LINK OF PREDECESSOR
001213  000001 4500 12              STZ    LINKB,X         ***DBG
001214  000001 7430 12              STX    Y,LINKB,X       SET BACKWARD LINK OF SELF
                   001215            DEQ    CORE            SEE IF ANYONE ELSE WANTS CORE
001215  005206 7170 00              XED    Q$CORE+Q$XDEQ
                              *
                              *       ZERO BLOCK FOR USER
                              *
001216  000014 2350 11              LDA    T$TEM12,T       GET LENGTH
001217  777777 3750 07              ANA    -1,DL           MASK TO COUNT
001220  777777 6240 05              EAX    Z,-1,AL         SAVE COUNT MINUS ONE IN Z
001221  000010 7350 00              ALS    10-2            PUT COUNT IN REPEAT FIELD
001222  001400 6200 05              EAX    0,B$ABIT+B$BBIT,AL  TERMINATE CONDITIONS AND COUNT
001223  000000 4310 07              FLD    0,DL            CLEAR AQ
001224  000014 2220 11              LDX    X,T$TEM12,T     GET STARTING ADDRESS
001225  000002 6230 12              EAX    Y,2,X           SET SET INDEX REGISTER
001227  000000 5602 04        GETC8  RPDX   ,4              ZERO MEMORY LIKE MAD
001230  000000 7570 12              STAQ   0,X             CHONK
001231  000000 7570 13              STAQ   0,Y             CHONK
001232  002000 1240 03              SBLX   Z,$1K,DU        REMOVE A K AS DONE
001233  001227 6030 00              TRC    GETC8           TEST FOR MORE
                              *
```

```
                                 2825 *       FINISH UP                                                          1070
                                 2826 *                                                                          1080
                         001234  2827 GETC5   BSS     0                                                          1090
    001234  000016 3220 11       2828         LCX     X,T$TEM10,T    GET COMPLEMENT OF LENGTH TO RETURN              1100
 END OF BINARY CARD IOS00033
    001235  005300 0420 00       2829         ASX     X,$AVAIL       REDUCE AVAIL BY THAT AMOUNT                     1110
    001236  777777 6040 00       2830         TMI     $ERROR         ***BLEWIT                                       1120
    001237  000010 0730 11       2831         LREG    T$TEM16,T      RESTORE REGISTERS                               1130
    001240  000000 7100 17       2832         TRA     0,L            RETURN                                          1140
```

```
                          001241        USE     CODE
                                        HEAD    R
                                   *
                                   *
                                   *                                REQUEST MORE
                                   *
                                   *    THIS SUBROUTINE DOES A REQUEST FOR MORE MEMORY.  IT ASKS
                                   *    FOR ONE CORE UNIT.  UPON THE SUCCESSFUL COMPLETION OF THE
                                   *    REQUEST, THE NEW MEMORY IS LINKED ON THE FREE LIST BY
                                   *    CALLING RELC.
                                   *
                                   *    CALL BY
                                   *            TRA MORE  (ONLY 'GETC' SHOULD CALL THIS ROUTINE)
                                   *    CALL WITH
                                   *            C(T) = TRAP-BLOCK
                                   *    CALLS
                                   *            $CHSEG
                                   *            R$RELC
                                   *    EXIT    TO R$GETC3
                                   *    DESTROYS <ALL REGISTERS>
                                   *    USES NO LOCAL TEMPORARIES
                                   *
                          001241   MORE BSS     0
001241   005303 2220 00                 LDX     X,$MTOP             GET CURRENT TOP OF MEMORY
001242   001000 0220 03                 ADLX    X,$MQUAN,DU         ADD ONE CORE UNIT
001243   005303 7420 00                 STX     X,$MTOP             SAVE NEW EXPECTED TOP
001244   001443 7070 00                 TSX     L,MREQ              REQUEST MEMORY
                          001245   MORE1 BSS    0
001245   005303 2350 00                 LDA     $MTOP               GET BACK OLD TOP OF MEMORY
001246   001000 1350 03                 SBLA    $MQUAN,DU           IN AU
001247   001000 2750 07                 ORA     $MQUAN,DL           AND LENGTH IN AL
                          001250        RELC    A                   NOW DO A RELEASE TO LINK ON FREE LIST
001250   001252 7070 00                 TSX     L,R$RELC
001251   001165 7100 00                 TRA     GETC3               TRY AGAIN
                                   *$*  DISK    RELC
```

```
                        001252             USE     CODE
                                           HEAD    R
                                     *
                                     *
                                     *                             DE-ALLOCATION
                                     *
                                     *     THIS ROUTINE LINKS THE 'RELEASED' BLOCK OF MEMORY INTO
                                     *     THE FREE MEMORY LIST ACCORDING TO THE BLOCK'S ADDRESS.
                                     *     SINCE THE FREE LIST IS ORDERED BY BLOCK ADDRESSES FROM
                                     *     LOWEST TO HIGHEST, IN ORDER TO MAKE INSERTIONS AND DE-
                                     *     LETIONS EASIEST THERE IS ASSOCIATED WITH EACH BLOCK A
                                     *     FORWARD AND BACKWARD POINTER AS WELL AS A COUNT OF THE
                                     *     TOTAL NUMBER OF WORDS IN THE BLOCK.
                                     *
                                     *     CALL BY
                                     *             TSX L,R$RELC
                                     *     CALL WITH
                                     *             C(AU) = BLOCK-ADDRESS
                                     *             C(AL) = LENGTH
                                     *     CALLS
                                     *             R$MEMCK (CONDITIONALLY)
                                     *     EXIT TO 0,L
                                     *     PRESERVES ALL REGISTERS EXCEPT C(A)
                                     *     USES LOCAL TEMPORARIES MREG THRU MREG+7
                                     *
                        001252             USE     CODE
                        001252        RELC BSS     0               RELEASE A BLOCK OF MEMORY
      001252  005370 7530 00               SREG    MREG            SAVE REGISTERS
      001253  777777 3750 03               ANA     -1,DU           ISOLATE RELEASE ADDRESS
      001254  005376 7550 00               STA     TEMP            SAVE FOR TEST
      001255  005302 2350 00               LDA     $MTOPO          TEST TO SEE IF ADDRESS
      001256  005303 2360 00               LDQ     $MTOP           OF BLOCK TO BE RELEASED
      001257  005376 1110 00               CWL     TEMP            IS IN BUFFER AREA
      001260  777777 6010 00               TNZ     $ERROR          BLEWIT
   END OF BINARY CARD IOS00034
      001261  005376 2220 00               LDX     X,TEMP          GET INSERT ADDRESS
      001262  005374 7200 00               LXL     0,MREG+4        GET LENGTH
      001263  777777 6000 00               TZE     $ERROR          ***BLEWIT
      001264  005376 0400 00               ASX     0,TEMP          ADD TO STARTING ADDRESS
      001265  000000 4400 12               SXL     0,LEN,X         SAVE IN BLOCK
      001266  005376 1110 00               CWL     TEMP            TEST IF ALL IS IN RANGE
      001267  777777 6010 00               TNZ     $ERROR          ***BLEWIT
      001270  005344 2220 00               LDX     X,FIRST         GET POINTER TO FIRST FREE BLOCK OF FREE LIST
      001271  001273 7100 00               TRA     *+2             ENTER SEARCH LOOP
                                     *
                                     *     LOCATE WHERE TO INSERT BLOCK IN FREE LIST ACCORDING TO ADDRESS
                                     *
      001272  000000 2220 12               LDX     X,LINKF,X       GET PTR TO NEXT BLOCK ON FREE LIST
      001273  001306 6000 00               TZE     RELC2           DID WE JUST FALL OFF THE END OF THE LIST?
      001274  005374 1020 00               CMPX    X,INSRT         NO,ARE WE POINTING PAST THE HOLE?
```

```
001275  001272 6020 00      2918          TNC     *-3              NO, LOOK AGAIN                               600
                            2919 *                                                                              610
                            2920 *                                                                              620
                            2921 *                                                                              630
                            2922 *        LOCATED POSITION OF BLOCK WITH XR-X POINTING TO SUCCESSOR             640
                            2923 *                                                                              650
                  001276    2924 RELC1    BSS     0                                                             660
001276  005374 2240 00      2925          LDX     Z,INSRT          NOW Z POINTS TO INSERT                       670
001277  000001 2230 12      2926          LDX     Y,LINKB,X        *** Y POINTS TO PREDECESSOR                  680
                            2927                                   *** X POINTS TO SUCCESSOR                    690
001300  000001 4500 14      2928          STZ     LINKB,Z          ***DRG                                       750
001301  000001 7430 14      2929          STX     Y,LINKB,Z        SET BACKWARD LINK OF INSERT                  760
001302  000000 7420 14      2930          STX     X,LINKF,Z         "  FORWARD  "   "   "                       770
001303  000001 7440 12      2931          STX     Z,LINKB,X        RESET SUCCESSOR'S BACKWARD LINK              780
001304  000000 7440 13      2932          STX     Z,LINKF,Y        AND FINISH LINKING BY RESETTING PREDECESSOR'
                            2933                                   ***FORWARD LINK.                             800
001305  001310 7100 00      2934          TRA     RELC3            NOW DO SOME RE-COMBINING                     810
                            2935 *                                                                              820
                            2936 *        RELEASED BLOCK FITS BETWEEN THE LAST FREE BLOCK AND 'LAST'            830
                            2937 *                                                                              840
                  001306    2938 RELC2    BSS     0                                                             850
001306  005346 6220 00      2939          EAX     X,LAST           SO MAKE X POINT TO SUCCESSOR                 860
 END OF BINARY CARD IOS00035
001307  001276 7100 00      2940          TRA     RELC1            AND TREAT NORMALLY                           870
                            2941 *                                                                              880
                            2942 *                                                                              890
                            2943 *        NOW THAT THE BLOCK HAS BEEN CORRECTLY LINKED ON THE FREE              900
                            2944 *        LIST TRY TO RECOMBINE WITH ITS BUDDIES.                               910
                            2945 *        THERE ARE FOUR (4) CASES TO CONSIDER:                                 920
                            2946 *                                                                              925
                            2947 *                CASE I: THERE EXISTS A BUDDY ABOVE IN MEMORY                  930
                            2948 *                CASE II:THERE EXISTS A BUDDY BELOW IN MEMORY                  940
                            2949 *                CASE III:ALL OF THE ABOVE                                     950
                            2950 *                CASE IV: NONE OF THE ABOVE                                    960
                            2951 *                                                                              970
                  001310    2952 RELC3    BSS     0                                                             980
001310  001350 7070 00      2953          TSX     L,RELC4          CASE I: TRY REJOINING WITH BUDDY ABOVE       990
001311  000000 6220 14      2954          EAX     X,0,Z            LET X POINT TO INSERT                        1000
001312  000000 6240 13      2955          EAX     Z,0,Y            LET Z POINT TO ITS PREDECESSOR               1010
001313  005374 7440 00      2956          STX     Z,INSRT          NOW CALL THE PREDECESSOR INSERT              1020
001314  001350 7070 00      2957          TSX     L,RELC4          FUDGE CASE II TO CASE I                      1030
```

```
                                  2959 *                                                                      1050
                                  2960 *       WE'RE DONE.  BLOCK IS CORRECTLY LINKED IN LIST.                1060
                                  2961 *       GRAB CALLER'S REGISTERS AND RETURN.                            1070
                                  2962 *                                                                      1080
   001315  005374 7220 00         2963         LXL     X,MREG+4        GET BACK AMOUNT RELEASED               1090
   001316  005300 0420 00         2964         ASX     X,$AVAIL        STATISTIC:  FREE CORE                  1100
   001317  005303 2220 00         2965         LDX     X,$MTOP         GET TOP OF MEMORY                      1110
   001320  005302 1220 00         2966         SBLX    X,$MTOPO        COMPUTE BUFFER SIZE                    1120
   001321  001000 1220 03         2967         SBLX    X,$MQUAN,DU     MINUS A CORE QUANTUM                   1130
   001322  001344 6020 00         2968         TNC     RELCX           OK, FORGET ABOUT IT                    1140
   001323  005301 1020 00         2969         CMPX    X,$MEMRQ        COMPARE IT TO MEMORY REQUIREMENTS      1150
   001324  001344 6020 00         2970         TNC     RELCX           CAN WE AFFORD TO RELEASE SOME MEMORY?  1160
   001325  005212 2340 00         2971         SZN     Q$BUSY+Q$CORE   BUT FIRST IS MEMORY BUSY?              1170
   001326  001344 6010 00         2972         TNZ     RELCX           DON'T UNDER ANY CIRCUMSTANCES TRY TO CALL MEMCK.
   001327  005350 2210 03         2973         LDX     T,TRAP,DU       GET DUMMY TCB                          1190
   001330  005350 7260 07         2974         LXL     J,TRAP,DL       AND DUMMY JCB                          1200
                       001331     2975         BRANCH  NOPASS,MEMCK    OK TO SET UP TASK TO RELEASE MEMORY    1210
   001331  001467 7000 00                      TSX     0,T$GETT
   001332  000000 6220 11                      EAX     X,0,T
   001333  000005 2210 12                      LDX     T,T$LINK,X
   001334  000000 6210 12                      EAX     T,0,X
 END OF BINARY CARD IOS00036
   001335  001365 6200 00                      EAX     0,MEMCK
   001336  000004 7400 11                      STX     0,T$TRA,T
   001337  000004 6200 11                      EAX     0,Q$OFFST,T
   001340  005162 7170 00                      XED     Q$XADD+Q$TASK
   001341  000005 2210 12                      LDX     T,T$LINK,X
                       001342                  BUGXR   (0,X)
                       525206          BUGBUG  SET     BUGBUG+1
   001342  525206 2200 03                      LDX     0,BUGBUG,DU
   001343  525206 2220 03                      LDX     X,BUGBUG,DU
                       001344     2976 RELCX   BSS     0               EXIT                                   1220
   001344  005370 0730 00         2977         LREG    MREG            RESTORE REGISTERS                      1230
                       001345     2978         BUGA                    REMIND HIM THAT A IS INVALID           1240
                       525207          BUGBUG  SET     BUGBUG+1
   001345  525207 2350 03                      LDA     BUGBUG,DU
   001346  525207 2750 07                      ORA     BUGBUG,DL
   001347  000000 7100 17         2979         TRA     0,L             RETURN TO CALLER                       1250
```

```
                                  2981 *                                                                  1270
                                  2982 *           THIS SUBROUTINE TRIES TO RECOMBINE A BLOCK OF MEMORY    1280
                                  2983 *           AND ITS BUDDY ABOVE (IF IT IS ALSO IN THE FREE LIST)    1290
                                  2984 *           INTO A SINGLE BLOCK.                                    1300
                                  2985 *                                                                   1310
                                  2986 *           CALL BY                                                 1320
                                  2987 *                     TSX L.RELC4                                   1330
                                  2988 *           CALL WITH                                               1340
                                  2989 *                     C(Z) = INSERT-BLOCK-ADDRESS                   1350
                                  2990 *                     C(X) = INSERT'S SUCCESSOR                     1360
                                  2991 *           EXIT TO 0.L                                             1370
                                  2992 *           DESTROYS 0.X.Z                                          1380
                                  2993 *                                                                   1390
                         001350   2994 RELC4  BSS     0                                                    1400
  001350  000000 7200 14          2995        LXL     0.LEN.Z          COMPUTE ADDRESS OF BUDDY IN         1410
  001351  005374 0200 00          2996        ADLX    0.INSRT          ...UPPER MEMORY                     1420
  001352  000000 1000 14          2997        CMPX    0.LINKF.Z        IS IT PART OF THE FREE LIST?        1430
  001353  000000 6010 17          2998        TNZ     0.L              NO. SO RETURN TO CALLER             1440
                                  2999 *                                                                   1450
                                  3000 *      FOUND BUDDY                                                  1460
                                  3001 *                                                                   1470
  001354  000000 2200 12          3002        LDX     0.LINKF.X        RESET INSERT'S FORWARD LINK TO      1480
  001355  000000 7400 14          3003        STX     0.LINKF.Z          THAT OF BUDDY'S.                  1490
  001356  000001 7440 10          3004        STX     Z.LINKB.0        RESET PREDECESSOR'S BACKWARD POINTER 1500
  001357  000000 7200 12          3005        LXL     0.LEN.X          GET THE LENGTH OF BUDDY             1510
  001360  005376 7400 00          3006        STX     0.TEMP              SAVE IT                          1520
  001361  000000 7200 14          3007        LXL     0.LEN.Z          GET LENGTH OF INSERT                1530
  001362  005376 0200 00          3008        ADLX    0.TEMP           TO GET TOTAL LENGTH                 1540
END OF BINARY CARD IOS00037
  001363  000000 4400 14          3009        SXL     0.LEN.Z          AND RESET LENGTH OF BLOCK           1550
  001364  000000 7100 17          3010        TRA     0.L              RETURN TO CALLER                    1560
                                  3011 *                                                                   1570
                                  3012 *                                                                   1580
                         005350   3013        USE     STORE                                                1590
                         005350   3014        EIGHT                                                        1600
                         005350   3015 TRAP   BSS     T$LEN            DUMMY TCB WHEN CREATING MEMCK       1610
                         005370   3016 MREG   EQU     TRAP+T$LEN-8     TEMP REGISTER STORAGE               1620
                         005374   3017 INSRT  EQU     MREG+4           POINTER TO INSERT BLOCK             1630
                         005376   3018 TEMP   EQU     MREG+6           FOR SCRATCH WORK                    1640
                         001365   3019        USE     PREVIOUS                                             1650
                                  3020 *$*    DISK    MEMCK                                                1660
```

```
                  001365          USE     CODE
                                  HEAD    R
                         *
                         *
                         *                               MEMORY RELEASE
                         *
                         *        THIS TASK CHECKS TO SEE IF IT CAN GIVE BACK MEMORY TO THE
                         *        SYSTEM.  IF SO,  IT RETURNS BLOCKS IN MULTIPLES OF $MQUAN.
                         *        THE ALGORITHM IS AS FOLLOWS:
                         *        GIVEN:
                         *                MEMRQ -- AMOUNT OF MEMORY NEEDED THOUGH NOT NECESSARIYLY IN USE
                         *                TOTAL -- TOTAL BUFFER AREA SIZE
                         *                AVAIL -- AMOUNT OF FREE CORE THOUGH NOT NECESSARILY CONTIGOUS
                         *                USED  -- AMOUNT OF CORE BUSY (TOTAL-AVAIL)
                         *        REQUIREMENT:
                         *                USED <= MEMRQ
                         *        ENTERED WHEN:
                         *                TOTAL-MEMRQ => MQUAN
                         *        RESTRICTIONS:
                         *                (1)  CAN RELEASE ONLY THE LAST PHYSICAL BLOCK
                         *                (2)  A RELEASE CAN BE EFFECTED ONLY IF AFTER THE RELEASE
                         *                     THERE IS AT LEAST ONE BLOCK OF MEMORY THAT IS OF
                         *                     MEMRQ-USED SIZE.
                         *        MEMCK IS AN ASYNCHRONOUS TASK, HENCE IT MAY ALL TEMP'S
                         *        USES NO LOCAL TEMPORARIES
                         *
                  001365   MEMCK  BSS     0
                  001365          ENQ     CORE              GET ON CORE QUEUE
 001365  005204 7170 00           XED     Q$CORE+Q$XENQ
 001366  005303 2220 00           LDX     X,$MTOP           GET TOP OF BUFFER AREA
 001367  005302 1220 00           SBLX    X,$MTOPO          MINUS BOTTOM = TOTAL
 001370  000027 7420 11           STX     X,T$TEMP1,T       C(T$TEMP1,T) = TOTAL
 001371  005300 1220 00           SBLX    X,$AVAIL          MINUS AVAIL = USED
 001372  777777 6040 00           TMI     $ERROR            ***BLEWIT
 001373  000026 7420 11           STX     X,T$TEMP2,T       C(T$TEMP2,T) = USED
 001374  005301 2350 00           LDA     $MEMRQ            GET AMOUNT OF MEMORY REQUIRED
 001375  000026 1350 11           SBLA    T$TEMP2,T         MINUS USED = AMOUNT NEEDED STILL
 001376  000025 7550 11           STA     T$TEMP3,T         SAVE IT
 001377  001440 6040 00           TMI     MEMX              EXIT
 001400  001440 6000 00           TZE     MEMX              EXIT IF CONDITIONS CHANGED ON US
 001401  000022 7710 00           ARL     36-18             MOVE TO AL FOR RPL
                         *
                         *        SEARCH FOR BLOCK OF 'NEEDED' SIZE
                         *
 001402  005344 2240 00           LDX     Z,FIRST           GET POINTER TO FIRST FREE BLOCK
 001403  000302 5002 00    MEM1   RPL     ,TNC,TZE          RUN DOWN LINKED LIST
 001404  000000 1150 14           CMPA    0,Z               FOR A BLOCK BIG ENOUGH
 001405  001412 6000 00           TZE     MEM2              FOUND A BLOCK THAT IS 'JUST RIGHT'
 001406  001412 6020 00           TNC     MEM2              MORE THAN BIG ENOUGH
 001407  000000 2240 14           LDX     Z,0,Z             GET BACK PTR OF WHERE WE LEFT OFF
```

R CORE MANAGEMENT -- MEMORY RELEASE

```
END OF BINARY CARD IOS00038
   001410  001403 6010 00     3071        TNZ     MEM1             IF NOT AT END OF LIST. KEEP SEARCHING
 . 001411  001440 7100 00     3072        TRA     MEMX             OTHERWISE EXIT
                              3073 *
                              3074 *      Z POINTS TO BLOCK NEEDED EVENTUALLY
                              3075 *
   001412  000000 7230 14     3076 MEM2   LXL     Y.LEN.Z          GET THE LENGTH OF THIS BLOCK
   001413  005346 6220 00     3077        EAX     X.LAST           GET PTR TO LAST BLOCK
   001414  000001 1040 12     3078        CMPX    Z.LINKB.X        DOES Z POINT TO LAST BLOCK?
   001415  001420 6010 00     3079        TNZ     *+3              NO. NOT LAST
   001416  000025 1230 11     3080        SBLX    Y.T$TEMP3.T      YES. SO SUBTRACT OFF NEEDED
   001417  001422 7100 00     3081        TRA     MEM3             CONTINUE
   001420  000001 2240 12     3082        LDX     Z.LINKB.X        GET PTR TO LAST FREE BLOCK
   001421  000000 7230 14     3083        LXL     Y.LEN.Z          GET ITS LENGTH
   001422  001000 1030 03     3084 MEM3   CMPX    Y.$MQUAN.DU      COMPARE TO CORE QUANTUM
   001423  001440 6020 00     3085        TNC     MEMX             IF SMALLER. EXIT
   001424  777000 3630 03     3086        ANX     Y.-$MQUAN.DU     OTHERWISE ROUND DOWN TO MULTIPLE OF MQUAN
   001425  000024 7430 11     3087        STX     Y.T$TEMP4.T      SAVE AMOUNT FOR RELEASE
   001426  000000 7200 14     3088        LXL     0.LEN.Z          GET BACK ITS LENGTH
   001427  000024 1200 11     3089        SBLX    0.T$TEMP4.T      MINUS AMOUNT TO RELEASE
   001430  000000 4400 14     3090        SXL     0.LEN.Z          RESTORE NEW LENGTH
   001431  001434 6010 00     3091        TNZ     *+3              IS THE BLOCK NULL
   001432  000001 2200 14     3092        LDX     0.LINKB.Z        YES. RESET LAST
   001433  000001 7400 12     3093        STX     0.LINKB.X        TO POINT TO NEW LAST BLOCK
   001434  000024 3220 11     3094        LCX     X.T$TEMP4.T      GET BACK AMOUNT TO BE RELEASED
   001435  005303 0420 00     3095        ASX     X.$MTOP          SUBTRACT FROM TOP OF MEMORY PTR'
END OF BINARY CARD IOS00039
   001436  005300 0420 00     3096        ASX     X.$AVAIL         SUBTRACT FROM AVAILABLE
   001437  001443 7070 00     3097        TSX     L.MREQ           RELEASE MEMORY TO SYSYTEM
                  001440      3098 MEMX   BSS     0                DONE
                  001440      3099        DEQ     CORE             RELEASE CORE QUEUE. AWAKEN NEXT TASK
   001440  005206 7170 00                 XED     Q$CORE+Q$XDEQ
   001441  001477 7000 00     3100        TSX     0.T$RELT         AND RELEASE TCB
                  001442      3101        EXIT                     EVAPORATE
   001442  003074 7100 00                 TRA     $EXIT
                              3102 *$*    DISK    MREQ
```

```
                     001443      3104         USE      CODE                                     110
                                 3105         HEAD     R                                        120
                                 3106 *                                                         130
                                 3107 *                                 MEMOREY REQUESTS        140
                                 3108 *                                                         150
                                 3109 *      THIS SUBROUTINE RESETS THE TOP OF MEMORY TO THE ADDRESS   160
                                 3110 *      SPECIFIED BY C(MTOP).                              170
                                 3111 *                                                         180
                                 3112 *      CALL BY                                            190
                                 3113 *               TSX L,MREQ                                200
                                 3114 *      CALL WITH                                          210
                                 3115 *               C(XT) = TBLOCK-ADDRESS                    220
                                 3116 *               C(XJ) = JBLOCK-ADDRESS                    230
                                 3117 *               C(MTOP) = NEW SETTING                     240
                                 3118 *      CALLS                                              250
                                 3119 *               $CHSEG                                    260
                                 3120 *      EXITS TO 0,L                                       270
                                 3121 *      RETURNS WITH                                       280
                                 3122 *               C(XT) = TBLOCK-ADDRESS                    290
                                 3123 *               C(XJ) = JBLOCK-ADDRESS                    300
                                 3124 *               C(XL) = RESTART ADDRESS                   310
                                 3125 *               C(MTOP) = NEW SETTING                     320
                                 3126 *       USES NO LOCAL TEMPORARIES, ONLY T$TEM9            330
                                 3127 *                                                         340
    001443  000017 4470 11       3128 MREQ    SXL      L,T$TEM9,T         SAVE RETURN ADDRESS   350
    001444  005303 2200 00       3129         LDX      0,$MTOP            GET PASSED SETTING    360
    001445  000777 0200 03       3130         ADLX     0,$MQUAN-1,DU      ROUND UP TO           370
    001446  777000 3600 03       3131         ANX      0,-$MQUAN,DU       NEXT CORE MULTIPLE    380
    001447  005303 7400 00       3132         STX      0,$MTOP            AND SAVE IT           390
                     001450      3133 MREQ1   CHSEG    $BUFSEG,$MTOP      MEMORY REQUEST        400
    001450  000656 7000 00                    TSX      0,$CHSEG
    001451  000000 0000 00                    ARG      $BUFSEG
    001452  005303 0000 00                    ARG      $MTOP
                     001453      3134         CHECK    MREQ2,B$RZ,MREQ1,B$IO,MREQ1              410
    001453  000000 7200 11                    LXL      0,T$SRW1,T
    001454  000077 3600 03                    ANX      0,B$STMK,DU
    001455  001463 6000 00                    TZE      MREQ2
    001456  000003 1000 03                    CMPX     0,B$RZ,DU
    001457  001450 6000 00                    TZE      MREQ1
    001460  000004 1000 03                    CMPX     0,B$IO,DU
    001461  001450 6000 00                    TZE      MREQ1
    001462  777777 7100 00                    TRA      $ERROR
                     001463      3135 MREQ2   BSS      0                  SUCCESSFUL REQUEST    420
 END OF BINARY CARD IOS00040
    001463  000017 7270 11       3136         LXL      L,T$TEM9,T         RETRIEVE RETURN ADDRESS  430
                     001464      3137         BUGL     (T$TEM9,T)         BUG IT                440
                     525210           BUGBUG  SET      BUGBUG+1
    001464  525210 2200 03                    LDX      0,BUGBUG,DU
    001465  000017 4400 11                    SXL      0,T$TEM9,T
    001466  000000 7100 17       3138         TRA      0,L                RETURN TO CALLER      450
```

```
3139 *$*     DISK     TBLOCK                                        460
```

R TRAP MANAGEMENT -- DESCRIPTION

```
001467           USE     CODE
                 HEAD    T
       *
       *
       *         THESE MACROS GET AND RELEASE TRAP BLOCKS.
       *         CLOBBERS C(0), C(X), C(T).
       *
       *
       *                                 GETT
       *
       GETT      MACRO   <NO-ARGUMENTS>
                 TSX     0,T$GETT        CALL SUBROUTINE
                 ENDM    GETT
       *
       *
       *                                 RELT MACRO
       *
       RELT      MACRO   <NO-ARGUMENTS>
                 TSX     0,T$RELT        CALL SUBROUTINE
                 ENDM    RELT
```

```
                   001467           USE     CODE
                                    HEAD    T
                              *
                              *
                              *                               GETT
                              *
                              *     THIS SUBROUTINE GETS A TRAP BLOCK. C(XT) POINTS TO THE
                              *     CURRENT TBLOCK WHILE C(T$LINK,T) POINTS TO THE OLD ONE
                              *
                              *     ENTERED WITH
                              *             C(XT) = TCB
                              *             C(XJ) = JCB
                              *     ENTERED BY
                              *             TSX 0,R$GETT
                              *     CALLS
                              *             R$GETC
                              *     RETURNS 0,0
                              *     RETURNS WITH
                              *             C(XT) = NEW TCB
                              *             C(XJ) = JCB
                              *
                              *
                   001467     GETT  BSS     0
                   001467           GETC    (LEN,DL)          GET A BLOCK ABOUT THE SIZE OF A TBLOCK
001467  000030 2350 07              LDA     LEN,DL
001470  001152 7070 00              TSX     L,R$GETC
001471  000005 7410 01              STX     T,LINK,AU         POINT TO PREVIOUS TBLOCK
001472  000000 6210 01              EAX     T,0,AU            MAKE C(XT) POINT TO NEW BLOCK
001473  777777 6000 00              TZE     $ERROR            ***DBG
001474  777777 6040 00              TMI     $ERROR            ***DBG
001475  000006 4460 11              SXL     J,T$JCB,T         SET JCB POINTER
001476  000000 7100 10              TRA     0,0               RETURN TO CALLER
```

T TRAP MANAGEMENT -- RELT

```
                     001477      3193           USE      CODE                                      630
                                 3194           HEAD     T                                         640
                                 3195 *                                                            650
                                 3196 *                                                            660
                                 3197 *                                  RELT                      670
                                 3198 *                                                            680
                                 3199 *         THIS SUBROUTINE RELEASES THE CURRENT TRAP BLOCK.   690
                                 3200 *                                                            700
                                 3201 *         ENTERED WITH                                       710
                                 3202 *                  C(XT) = TCB                               720
                                 3203 *                  C(XJ) = JCB                               730
                                 3204 *         ENTERED BY                                         740
                                 3205 *                  TSX 0,R$RELT                              750
                                 3206 *         CALLS                                              760
                                 3207 *                  R$RELC                                    770
                                 3208 *         RETURNS TO 0,0                                     780
                                 3209 *         RETURNS WITH                                       790
                                 3210 *                  C(XJ) = JCB                               800
                                 3211 *                                                            810
                                 3212 *                                                            820
                     001477      3213 RELT      BSS      0                                         830
001477  000000 6350 11           3214           EAA      0,T           TRAP ADDRESS TO AU          840
001500  777777 6000 00           3215           TZE      $ERROR        CHECK FOR LEGAL RELEASE     850
001501  000030 2750 07           3216           ORA      LEN,DL        TRAP LEN TO AL              860
                     001502      3217           RELC     A             RELEASE IT                  870
001502  001252 7070 00                          TSX      L,R$RELC
                     001503      3218           BUGXR    T             SPPML                       880
                     525211           BUGBUG    SET      BUGBUG+1
001503  525211 2210 03                          LDX      T,BUGBUG,DU
001504  000000 7100 10           3219           TRA      0,0           RETURN TO CALLER            890
                                 3220 *$*       DISK     JBLOCKS                                   900
```

```
       T                          JOB CONTROL BLOCK MANAGEMENT -- DESCRIPTION

001505             USE      CODE
                   HEAD     J
      *
      *
      *            THESE MACROS GET AND RELEASE JOB CONTROL BLOCKS.
      *            CLOBBERS C(X0), C(XX), AND C(XJ).
      *
      *
      *                                       GETJ
      *
        GETJ       MACRO    <NO-ARGUMENTS>
                   TSX      0,J$GETJ          CALL SUBROUTINE
                   ENDM     GETJ
      *
      *
      *                                       RELJ
      *
        RELJ       MACRO    <NO-ARGUMENTS>
                   TSX      0,J$RELJ          CALL SUBROUTINE
                   ENDM     RELJ
```

```
                  001505    3243         USE     CODE                                                    320
                            3244         HEAD    J                                                       330
                            3245 *                                                                       340
                            3246 *                                                                       350
                            3247 *                                 GETJ                                  360
                            3248 *                                                                       370
                            3249 *       THIS SUBROUTINE GETS A JOB CONTROL BLOCK.  C(XJ) POINTS TO      380
                            3250 *       THE NEW JCB.                                                    390
                            3251 *                                                                       400
                            3252 *       ENTERED WITH                                                    410
                            3253 *               C(XT) = TCB                                             420
                            3254 *       ENTERED BY                                                      430
                            3255 *               TSX 0,J$GETJ                                            440
                            3256 *       CALLS                                                           450
                            3257 *               R$GETC                                                  460
                            3258 *       RETURNS 0,0                                                     470
                            3259 *       RETURNS WITH                                                    480
                            3260 *               C(XT) = TCB                                             490
                            3261 *               C(XJ) = NEW JCB                                         500
                            3262 *       CLOBBERS C(A), C(XL)                                            510
                            3263 *       USES NO TEMPS ITSELF                                            520
                            3264 *                                                                       530
                  001505    3265 GETJ    BSS     0                                                       540
                  001505    3266         GETC    (LEN,DL)        GET A BLOCK ABOUT THE SIZE OF A JBLOCK  550
  001505  000020 2350 07                 LDA     LEN,DL
END OF BINARY CARD IOS00041
  001506  001152 7070 00                 TSX     L,R$GETC
  001507  000000 6260 01    3267         EAX     J,0,AU          MAKE C(XJ) POINT TO NEW BLOCK           560
  001510  000000 7100 10    3268         TRA     0,0             RETURN TO CALLER                        570
```

J JOB CONTROL BLOCK MANAGEMENT -- RELJ

```
                      001511          USE     CODE
                                      HEAD    J
                              *
                              *
                              *                               RELJ
                              *
                              *       THIS SUBROUTINE RELEASES THE CURRENT JOB CONTROL BLOCK.
                              *
                              *       ENTERED WITH
                              *               C(XT) = TCB
                              *               C(XJ) = JCB TO BE RELEASED
                              *       ENTERED BY
                              *               TSX 0,J$RELJ
                              *       CALLS
                              *               R$RELC
                              *       RETURNS TO 0,0
                              *       RETURNS WITH
                              *               C(XT) = TCB
                              *       CLOBBERS C(A), C(XL)
                              *       USES NO TEMPS ITSELF
                              *
                      001511  RELJ    BSS     0
001511  000000 6350 16                EAA     0,J             JCB ADDRESS TO AU
001512  777777 6000 00                TZE     $ERROR          CHECK FOR LEGAL RELEASE
001513  000020 2750 07                ORA     LEN,DL          JCB LEN TO AL
                      001514          RELC    A               RELEASE IT
001514  001252 7070 00                TSX     L,R$RELC
                      001515          BUGXR   J               SPPML
                      525212  BUGBUG  SET     BUGBUG+1
001515  525212 2260 03                LDX     J,BUGBUG,DU
001516  000000 7100 10                TRA     0,0             RETURN TO CALLER
                              *$*     DISK    LOG1
```

```
001517          USE      CODE
                HEAD     C
        *
        *
        *                                LOGGING
        *
        *       THE SKEDS LOGGING DEVICE (OPERATOR'S CONSOLE) IS A SINGLE
        *       RESOURCE, AND HENCE MUST BE QUEUED FOR.  THE LOGGING
        *       ROUTINES ARE CALLED BY A SET OF MACROS WHICH INSERT SPECI-
        *       FIED INFORMATION INTO THE LOGGING MESSAGE BUFFER.
        *
        *       ALL LOGGING MESSAGES START BY CALLING THE LOGS MACRO, WHICH
        *       DOES THE QUEUEING FOR THE LOGGING DEVICE.  ALL MESSAGE
        *       SEQUENCES END WITH THE LOGX MACRO WHICH WRITES OUT THE
        *       MESSAGE AND RELEASES THE QUEUE.
        *
        *       THE FOLLOWING MACROS CAN BE USED TO CALL THE LOGGING ROUINTES:
        *
        *
        *       LOGS                     START OF LOG MESSAGE
        *       LOGC                     LOG TEXT CHARACTERS (TALLY WORD IS ARG 1)
        *       LOG                      FORCE THE FLUSHING OF OUTPUT BUFFER
        *       LOGX                     END OF MESSAGE -- EXIT AND SEND MESSAGE
        *
        *       NOTE THAT AFTER LOGS IS CALLED, THE LIST STRUCTURE SHOULD NOT
        *       BE MODIFIED UNTIL AFTER LOGX IS CALLED, AS THE LOGGING ROUTINES
        *       WILL MAINTAIN THEIR OWN LIST STRUCTURE.
```

0 LOGGING -- MACROS

```
001517          USE     CODE
                HEAD    0
      *
      *
      *                                  LOGS MACRO
      *
      *         SET UP FOR LOGGING
      *
      LOGS      MACRO   <NO-ARGUMENTS>
                TSX     L,0$LOGS
                ENDM    LOGS
      *
      *
      *                                  LOGC MACRO
      *
      *         LOG ASCII STRING POINTER TO BE FIRST ARGUMENT
      *
      LOGC      MACRO   SC POINTER ADDRESS/'A'
                INE     '#1','A'
                LDA     #1               GET SC POINTER IN A
                TSX     L,0$LOGC
                ENDM    LOGC
      *
      *
      *                                  LOG MACRO
      *
      *         FLUSH THE OUTPUT BUFFER
      *
      LOG       MACRO   <NO-ARGUMENTS>
                TSX     L,0$LOG
                ENDM    LOG
      *
      *
      *                                  LOGX MACRO
      *
      *         CLEANUP AND EXIT
      *
      LOGX      MACRO   <NO-ARGUMENTS>
                TSX     L,0$LOGX
                ENDM    LOGX
      *
      *
      *                                  LCRLF MACRO
      *
      *         LOG A CARRIAGE RETURN/ LINE FEED
      *
      LCRLF     MACRO   <NO-ARGUMENTS>
                TSX     L,0$LCRLF        CALL SUBROUTINE
                ENDM    LCRLF
      *
```

C LOGGING -- MACROS

```
*
*                                       LSP MACRO
*
*       LOG A SPACE
*
LSP     MACRO   <NO-ARGUMENTS>
        TSX     L,O$LSP                 CALL SUBROUTINE
        ENDM    LSP
*
*
*                                       LCHR MACRO
*
*       LOG A CHARACTER
*
LCHR    MACRO   <NO-ARGUMENTS>
        TSX     L,O$LCHR                CALL SUBROUTINE
        ENDM    LCHR
```

```
        0                          LOGGING SUBROUTINES -- LOGS

                    001517   3396          USE     CODE
                             3397          HEAD    0
                             3398 *
                             3399 *
                             3400 *        THE FOLLOWING SUBROUTINES ARE CALLED BY THE LOGGING MACROS.
                             3401 *
                             3402 *
                             3403 *                                 LOGS
                             3404 *
                             3405 *
                             3406 *        LOGS  ENQUEUES FOR THE CONTROL TTY.  ATFER SEIZING IT, IT
                             3407 *        LOCKS THE DEVICE, LOGS A HELLO MESSAGE, AND RETURNS TO THE
                             3408 *        CALLER.
                             3409 *
                             3410 *
                             3411 *        ENTERED WITH
                             3412 *                C(XT) = TCB
                             3413 *                C(XJ) = JCB
                             3414 *        ENTERED BY
                             3415 *                TSX L,0$LOGS
                             3416 *        CALLS
                             3417 *                Q$ENQ
                             3418 *                $LOCK
                             3419 *        RETURN TO 0,L
                             3420 *        RETURNS WITH
                             3421 *                C(XT) = TCB
                             3422 *                C(XJ) = JCB
                             3423 *                C(XL) = RESTART ADDRESS
                             3424 *        USES
                             3425 *                NO LOCALS
                             3426 *                T$TEMP7,T (UPPER)
                             3427 *
                             3428 *
                    001517   3429 LOGS     BSS     0                ENTRY POINT
001517  000021 7470 11       3430          STX     L,T$TEMP7,T      SAVE RETURN ADDRESS
                    001520   3431          ENQ     OP               WAIT FOR OP CONSOLE TO BE FREE
001520  005144 7170 00                     XED     Q$OP+Q$XENQ
                    001521   3432 LOGS1    LOCK    R$OPERN          NOW LOCK IT
001521  000736 7000 00                     TSX     0,$LOCK
001522  005665 0000 00                     ARG     R$OPERN
                    001523   3433          CHECK   LOGS2,B$BZ,LOGS1,B$LOCK,LOGS1
001523  000000 7200 11                     LXL     0,T$SRW1,T
001524  000077 3600 03                     ANX     0,B$STMK,DU
001525  001533 6000 00                     TZE     LOGS2
001526  000003 1000 03                     CMPX    0,B$BZ,DU
001527  001521 6000 00                     TZE     LOGS1
001530  000013 1000 03                     CMPX    0,B$LOCK,DU
001531  001521 6000 00                     TZE     LOGS1
END OF BINARY CARD IOS00042
001532  777777 7100 00                     TRA     $ERROR
```

```
          0                        LOGGING SUBROUTINES -- LOGS

                      001533     3434 LOGS2  ESS     0                   CONSOLE SUCCESSFULLY SEIZED          1450
001533  004675 2350 00           3435        LDA     LGTL1               INITIALIZE TALLY WORD                1460
001534  005400 7550 00           3436        STA     LGTAL                                                    1470
001535  004676 2370 00           3437        LDAQ    LM1                 PUT *IOS* MESSAGE IN BUFFER          1480
001536  005420 7570 00           3438        STAQ    LGBUF                                                    1490
001537  000021 2270 11           3439        LDX     L,T$TEMP7,T         RETRIEVE RETURN ADDRESS              1500
                      001540     3440        BUGU    (T$TEMP7,T)         BUG IT                               1510
                      525213          BUGBUG SET     BUGBUG+1
001540  525213 2200 03                       LDX     0,BUGBUG,DU
001541  000021 7400 11                       STX     0,T$TEMP7,T
001542  000000 7100 17           3441        TRA     0,L                 RETURN TO CALLER                     1520
```

0 LOGGING SUBROUTINES -- LOGC

```
                   001543     3443        USE     CODE                                                        1540
                              3444        HEAD    0                                                           1550
                              3445 *                                                                          1560
                              3446 *                                                                          1570
                              3447 *                                  LOGC                                    1580
                              3448 *                                                                          1590
                              3449 *      LOG CHARACTER STRING GIVEN BY SC POINTER IN A-REG                   1600
                              3450 *                                                                          1610
                              3451 *      ENTERED WITH                                                        1620
                              3452 *              C(XT) = TCB                                                 1630
                              3453 *              C(XJ) = JCB                                                 1640
                              3454 *              C(A)  = TALLY WORD                                          1650
                              3455 *      ENTERED BY                                                          1660
                              3456 *              TSX L.O$LOGC                                                1670
                              3457 *      CALLS                                                               1680
                              3458 *              O$LCHR                                                      1690
                              3459 *              O$LOG                                                       1700
                              3460 *      RETURNS TO 0.L                                                      1710
                              3461 *      RETURNS WITH                                                        1720
                              3462 *              C(XT) = TCB                                                 1730
                              3463 *              C(XJ) = JCB                                                 1740
                              3464 *      USES                                                                1750
                              3465 *              O$LGCTL                                                     1760
                              3466 *              T$TEMP7.T (UPPER)                                           1770
                              3467 *      CLOBBERS C(A), C(X0)                                                1780
                              3468 *                                                                          1790
 001543  000021 7470 11       3469 LOGC   STX     L.T$TEMP7.T       SAVE RETURN ADDRESS                       1800
 001544  005401 7550 00       3470        STA     LGCTL             STORE TALLY WORD                          1810
                   001545     3471 LOGC1  BSS     0                 LOOP POINT                                1820
 001545  005401 2350 52       3472        LDA     LGCTL.SC          GET NEXT CHARACTER                        1830
 001546  001550 6070 00       3473        TTF     *+2               CONTINUE IF WE GOT ONE                    1840
 001547  001552 7100 00       3474        TRA     LOGC2             EXIT IF DONE                              1850
                   001550     3475        LCHR                      LOG THE CHARACTER                         1860
 001550  001620 7070 00                   TSX     L.O$LCHR
 001551  001545 7100 00       3476        TRA     LOGC1             GET NEXT CHARACTER                        1870
                              3477 *                                                                          1880
                              3478 *      END OF TEXT STRING                                                  1890
                              3479 *                                                                          1900
                   001552     3480 LOGC2  BSS     0                                                           1910
 001552  000021 2270 11       3481        LDX     L.T$TEMP7.T       RETRIEVE RETURN ADDRESS                   1920
                   001553     3482        BUGU    (T$TEMP7.T)       BUG IT                                    1930
                   525214          BUGBUG SET     BUGBUG+1
 001553  525214 2200 03                   LDX     0.BUGBUG.DU
 001554  000021 7400 11                   STX     0.T$TEMP7.T
                   001555     3483        BUGA                      AND A-REG                                 1940
                   525215          BUGBUG SET     BUGBUG+1
 001555  525215 2350 03                   LDA     BUGBUG.DU
 001556  525215 2750 07                   ORA     BUGBUG.DL
END OF BINARY CARD IOS00043
 001557  000000 7100 17       3484        TRA     0.L               RETURN TO CALLER                          1950
```

0 LOGGING SUBROUTINES -- LOGX

```
                 001560    3466          USE     CODE
                           3487          HEAD    0
                           3488 *
                           3489 *                                LOGX
                           3490 *
                           3491 *        CLEAN UP, SEND MESSAGE, AND EXIT
                           3492 *
                           3493 *        ENTERED WITH
                           3494 *                C(XT) = TCB
                           3495 *                C(XJ) = JCB
                           3496 *        ENTER BY
                           3497 *                TSX L,O$LOGX
                           3498 *        CALLS
                           3499 *                O$LCRLF
                           3500 *                O$LOG
                           3501 *                $UNLCK
                           3502 *                Q$DEQ
                           3503 *        RETURNS TO 0,L
                           3504 *        RETURNS WITH
                           3505 *                C(XT) = TCB
                           3506 *                C(XJ) = JCB
                           3507 *        USES
                           3508 *                T$TEMP7,T (UPPER)
                           3509 *
                 001560    3510 LOGX     BSS     0                ENTRY POINT
001560 000021 7470 11      3511          STX     L,T$TEMP7,T      SAVE RETURN ADDRESS
                 001561    3512          LCRLF                    PUT A CR/LF ON MESSAGE
001561 001600 7070 00                    TSX     L,O$LCRLF
                 001562    3513          LOG                      LOG BUFFER TO TTY
001562 001645 7070 00                    TSX     L,O$LOG
                 001563    3514 LOGX1    UNLCK   R$OPFRN
001563 000746 7000 00                    TSX     0,$UNLCK
001564 005665 0000 00                    ARG     R$OPFRN
                 001565    3515          CHECK   LOGX2,B$BZ,LOGX1
001565 000000 7200 11                    LXL     0,T$SRW1,T
001566 000077 3600 03                    ANX     0,B$STMK,DU
001567 001573 6000 00                    TZE     LOGX2
001570 000003 1000 03                    CMPX    0,B$BZ,DU
001571 001563 6000 00                    TZE     LOGX1
001572 777777 7100 00                    TRA     $ERROR
                 001573    3516 LOGX2    BSS     0
                 001573    3517          DEQ     OP               RELEASE OP CONSOLE
001573 005146 7170 00                    XED     Q$OP+Q$XDEQ
001574 000021 2270 11      3518          LDX     L,T$TEMP7,T      RETRIEVE RETURN ADDRESS
                 001575    3519          BUGU    (T$TEMP7,T)      BUG IT
                 525216         BUGBUG   SET     BUGBUG+1
001575 525216 2200 03                    LDX     0,BUGBUG,DU
001576 000021 7400 11                    STX     0,T$TEMP7,T
001577 000000 7100 17      3520          TRA     0,L              RETURN TO CALLER
```

```
                   001600            USE     CODE                                       2330
                                     HEAD    0                                          2340
                               *                                                        2350
                               *                                                        2360
                               *                        LCRLF                           2370
                               *                                                        2380
                               *     PUT A CR/ LF IN OUTPUT BUFFER                      2390
                               *                                                        2400
                   001600      LCRLF BSS     0          ENTRY POINT                     2410
001600  000022 7470 11               STX     L,T$TEMP6,T SAVE RETURN ADDRESS            2420
001601  000015 2350 07               LDA     A$CR,DL    GET A CR                        2430
001602  001620 7070 00               TSX     L,LCHR     LOG IT                          2440
END OF BINARY CARD IOS00044
001603  000012 2350 03               LDA     A$LF,DU    GET A LF                        2450
001604  001620 7070 00               TSX     L,LCHR     LOG IT, ALSO                    2460
001605  000022 2270 11               LDX     L,T$TEMP6,T RETRIEVE RETURN ADDRESS        2470
                   001606            BUGU    (T$TEMP6,T) BUG IT                         2480
                   525217     BUGBUG SET     BUGBUG+1
001606  525217 2200 03               LDX     0,BUGBUG,DU
001607  000022 7400 11               STX     0,T$TEMP6,T
001610  000000 7100 17               TRA     0,L        RETURN TO CALLER                2490
                               *                                                        2500
                               *                                                        2510
                               *                        LSP                             2520
                               *                                                        2530
                               *     PUT A SPACE IN OUTPUT BUFFER                       2540
                               *                                                        2550
                   001611      LSP   BSS     0          LOG A SPACE                     2560
001611  000022 7470 11               STX     L,T$TEMP6,T SAVE RETURN ADDRESS            2570
001612  200040 2350 07               LDA     A$SP,DL    GET A SP                        2580
001613  001620 7070 00               TSX     L,LCHR     LOG IT                          2590
001614  000022 2270 11               LDX     L,T$TEMP6,T RETRIEVE RETURN ADDRESS        2600
                   001615            BUGU    (T$TEMP6,T) BUG IT                         2610
                   525220     BUGBUG SET     BUGBUG+1
001615  525220 2200 03               LDX     0,BUGBUG,DU
001616  000022 7400 11               STX     0,T$TEMP6,T
001617  000000 7100 17               TRA     0,L        RETURN TO CALLER                2620
                               *                                                        2630
                               *                                                        2640
                               *                        LCHR                            2650
                               *                                                        2660
                               *     PUT CHARACTER IN AL IN OUTPUT BUFFER               2670
                               *                                                        2680
                               *                                                        2690
                   001620      LCHR  BSS     0          LOG CHARACTER IN A              2700
001620  000177 3750 07               ANA     A$MASK,DL  ISOLATE ASCII CHARACTER         2710
001621  005400 7550 52               STA     LGTAL,SC   DROP CHARACTER INTO BUFFER      2720
001622  000000 6070 17               TTF     0,L        IF BUFFER NOT FULL, RETURN TO CALLER 2730
001623  001645 7100 00               TRA     LOG        OTHERWISE FORCE BUFFER          2740
```

```
        0                    LOGGING -- SPECIAL SUBROUTINES

                           3565 *                                                          2760
                           3566 *                               LOCTF                      2770
                           3567 *                                                          2780
                           3568 *      LOG C(Q) AS 12 OCTAL DIGITS                         2790
                           3569 *                                                          2800
                   001624  3570 LOCF   BSS    0                                            2810
  001624  000014 3200 03   3571        LCX    0,12,DU       LOG 12 DIGITS                  2820
  001625  001632 7100 00   3572        TRA    LOCT          DO IT                          2830
                           3573 *                                                          2840
                           3574 *                                                          2850
                           3575 *                               LOCTU                      2860
                           3576 *                                                          2870
                           3577 *      LOG C(QU) AS 6 OCTAL DIGITS                         2880
                           3578 *                                                          2890
                   001626  3579 LOCTU  BSS    0                                            2900
  001626  000006 3200 03   3580        LCX    0,6,DU        LOG 6 DIGITS                   2910
  001627  001632 7100 00   3581        TRA    LOCT          DO IT                          2920
                           3582 *                                                          2930
                           3583 *                                                          2940
                           3584 *                               LOCTL                      2950
                           3585 *                                                          2960
                           3586 *      LOG C(QL) AS 6 OCTAL DIGITS                         2970
                           3587 *                                                          2980
                   001630  3588 LOCTL  BSS    0                                            2990
  001630  000000 6360 06   3589        EAQ    0,QL          MOVE NUMBER TO QL              3000
END OF BINARY CARD IOS00045
  001631  001626 7100 00   3590        TRA    LOCTU         PRINT AS QU                    3010
                           3591 *                                                          3020
                           3592 *                                                          3030
                           3593 *                               LOCT                       3040
                           3594 *                                                          3050
                           3595 *      LOGS OCTAL DIGITS                                   3060
                           3596 *      ENTER WITH                                          3070
                           3597 *             C(Q) = DIGITS TO LOG (LEFT JUSTIFIED)        3080
                           3598 *             C(0) = NUMBER OF DIGITS                      3090
                           3599 *                                                          3100
                   001632  3600 LOCT   BSS    0                                            3110
  001632  000022 7470 11   3601        STX    L,T$TEMP6,T   SAVE RETURN ADDRESS            3120
  001633  000000 2350 07   3602 LOCT1  LDA    0,DL          CLEAR A                        3130
  001634  000003 7370 00   3603        LLS    3             GET NEXT DIGIT IN A            3140
  001635  100060 0750 07   3604        ADA    A$D0,DL       CONVERT TO ASCII DIGIT         3150
  001636  001620 7070 00   3605        TSX    L,LCHR        LOG IT                         3160
  001637  000001 0600 03   3606        ADX    0,1,DU        BUMP COUNT                     3170
  001640  001633 6040 00   3607        TMI    LOCT1         TES FOR DONE                   3180
  001641  000022 2270 11   3608        LDX    L,T$TEMP6,T   RETRIEVE RETURN ADDRESS        3190
                   001642  3609        BUGU   (T$TEMP6,T)   PUT IT                         3200
                   525221       BUGBUG SET    BUGBUG+1
  001642  525221 2200 03               LDX    0,BUGBUG,DU
  001643  000022 7400 11               STX    0,T$TEMP6,T
  001644  000000 7100 17   3610        TRA    0,L           RETURN TO CALLER               3210
```

```
         0                  LOGGING SUBROUTINES -- LOG

                  001645   3612          USE     CODE                                                   3230
                           3613          HEAD    0                                                      3240
                           3614 *                                                                       3250
                           3615 *                                                                       3260
                           3616 *                                  LOG                                  3270
                           3617 *                                                                       3280
                           3618 *        FLUSHES THE BUFFER TO OP CONSOLE AND RESETS TALLY WORD         3290
                           3619 *                                                                       3300
                  001645   3620 LOG      BSS     0                 ENTRY POINT                          3310
    001645  005410 7530 00  3621         SREG    LGREG             SAVE REGISTERS                       3320
    001646  004674 2350 00  3622         LDA     LGTLO             GET OUTPUT TALLY PROTOTYPE           3330
    001647  005400 1350 00  3623         SBLA    LGTAL             COMPUTE NUMBER OF CHARACTERS TO SEND 3340
    001650  000040 0350 07  3624         ADLA    =040,DL           TAKE OF BORROW, IF ANY               3350
    001651  777700 3750 07  3625         ANA     $TALMK,DL         MASK TO CHARACTER COUNT              3360
    001652  001670 6000 00  3626         TZE     LOG2              IF ZERO, NOTHING TO DO               3370
    001653  000014 7350 00  3627         ALS     18-6              RIGHT JUSTIFY COUNT IN AU            3380
    001654  005402 7550 00  3628         STA     LNCHR             AND SAVE IT                          3390
                  001655   3629 LOG1     APEND   R$OPFRN,(LGBUF,DU),LNCHR,(2,DU)                        3400
    001655  000551 7000 00               TSX     0,$APEND
 END OF BINARY CARD IOS00046
    001656  005665 0000 00               ARG     R$OPFRN
    001657  005420 0000 03               ARG     LGBUF,DU
    001660  005402 0000 00               ARG     LNCHR
    001661  000002 0000 03               ARG     2,DU
                  001662   3630          CHECK   LOG2,B$BZ,LOG1                                         3410
    001662  000000 7200 11               LXL     0,T$SRW1,T
    001663  000077 3600 03               ANX     0,B$STMK,DU
    001664  001670 6000 00               TZE     LOG2
    001665  000003 1000 03               CMPX    0,B$BZ,DU
    001666  001655 6000 00               TZE     LOG1
    001667  777777 7100 00               TRA     $ERROR
                           3631 *                                                                       3420
                           3632 *        MESSAGE OUT OK ON OP CONSOLE                                   3430
                           3633 *                                                                       3440
                  001670   3634 LOG2     BSS     0                                                      3450
    001670  004674 2350 00  3635         LDA     LGTLO             RE-INITIALIZE OUTPUT TALLY           3460
    001671  005400 7550 00  3636         STA     LGTAL                                                  3470
    001672  005410 0730 00  3637         LREG    LGREG             RESTORE THE REGISTERS                3480
    001673  000000 7100 17  3638         TRA     0,L               RETURN TO CALLER                     3490
                           3639 *                                                                       3500
                           3640 *                                                                       3510
                           3641 *        CONSTANT AND STORAGE AREA                                      3520
                           3642 *                                                                       3530
                  004674   3643          USE     CONST                                                  3540
    004674  005420 0170 40  3644 LGTLO   TALLYB  LGBUF,120         TALLY FOR FILLING BUFFER             3550
    004675  005422 0160 40  3645 LGTL1   TALLYB  LGBUF+2,120-8     TALLY INCLUDING INITIAL HEADER       3560
                  004676   3646          EVEN                                                           3570
    004676  177007015012    3647 LM1     OCT     177007015012      GRTCH  BELL  LF  CR                  3580
    004677  111117123040    3648 LM2     UASCI   1,IOS             HEADER MESSAGE                       3590
                  001674   3649          USE     PREVIOUS                                               3600
```

```
                  005400   3650        USE     STORE                                    3610
                  005400   3651 LGTAL  BSS     1          WORKING TALLY WORD            3620
                  005401   3652 LGCTL  BSS     1          TALLY WORD FOR LOGC PICKUP    3630
005402  000000 000000      3653 LNCHR  ZERO    **,        NUMBER OF CHARACTERS TO LOG   3640
                  005410   3654        EIGHT                                            3650
                  005410   3655 LGREG  BSS     8          TEMP REGISTER STORAGE         3660
                  005420   3656        EVEN                                             3670
                  005420   3657 LGBUF  BSS     128/4      BUFFER FOR LOGGING            3680
                  001674   3658        USE     PREVIOUS                                 3690
                           3659 *$*    DISK    OPCON1                                   3700
```

```
001674          USE      CODE
                HEAD     0
         *
         *
         *                               OPERATOR INTERFACE
         *
         *      THE SKEDS INPUT DEVICE (OPERATOR'S CONSOLE) IS A SINGLE
         *      RESOURCE, AND HENCE MUST BE QUEUED FOR.  (SEE LOGGING.)
         *      THE INPUT ROUTINES ARE CALLED BE A SET OF MACROS WHICH
         *      REMOVE SPECIFIED INFORMATION FROM THE INPUT MESSAGE BUFFER.
         *
         *      THE FOLLOWING MACROS CAN BE USED TO CALL THE INPUT ROUTINES:
         *
         *      ICHR                     INPUT NEXT CHAR FROM INPUT STREAM
         *      INBLK                    INPUT NEXT NON-BLANK CHAR
         *      ICMD                     INPUT THREE LETTER COMMAND
         *      IDELM                    INPUT THE HIGHEST PRIORITY DELIMITER
```

```
001674         USE      CODE
               HEAD     0
      *
      *
      *                                   ICHR
      *
      *        INPUT NEXT CHARACTER FROM INPUT STREAM
      *
      ICHR     MACRO    <NO-ARGUMENTS>
               TSX      L,0$ICHR          CALL SUBROUTINE
               ENDM     ICHR
      *
      *
      *                                   INBLK
      *
      *        INPUT NEXT NON-BLANK CHARACTER FROM INPUT STREAM
      *
      INBLK    MACRO    <NO-ARGUMENTS>
               TSX      L,0$INBLK         CALL SUBROUTINE
               ENDM     INBLK
      *
      *
      *                                   ICMD
      *
      *        INPUT THREE LETTER COMMAND FROM INPUT STREAM
      *
      ICMD     MACRO    <NO-ARGUMENTS>
               TSX      L,0$ICMD          CALL SUBROUTINE
               ENDM     ICMD
      *
      *
      *                                   IDELM
      *
      *        INPUT THE HIGHEST PRIORITY DELIMITER
      *
      IDELM    MACRO    <NO-ARGUMENTS>
               TSX      L,0$IDELM         CALL SUBROUTINE
               ENDM     IDELM
```

0 OERATOR INTERFACE -- ICHR

```
                001674    3718          USE     CODE
                          3719          HEAD    0
                          3720 *
                          3721 *
                          3722 *                                  ICHR
                          3723 *
                          3724 *        GET NEXT CHARACTER THROWING AWAY CONTROL CHARACTERS
                          3725 *        EXCEPT FOR CARRIAGE RETURNS AND TURNING ON BITS
                          3726 *        FOR SPECIAL CHARACTERS FROM TABLE A$ASCTB
                          3727 *
                          3728 *        ENETER WITH
                          3729 *                C(XT) = TCB
                          3730 *                C(XJ) = JCB
                          3731 *        ENTER BY
                          3732 *                TSX L.0$ICHR
                          3733 *        RETURNS TO 0.L
                          3734 *        RETURNS WITH
                          3735 *                C(XT) = TCB
                          3736 *                C(XJ) = JCB
                          3737 *                C(AL) = NEXT CHARACTER
                          3738 *
                001674    3739 ICHR     BSS     0                   ENTRY POINT
END OF BINARY CARD IOS00047
    001674  005460 2340 00    3740          SZN     RFLG                TEST RE-READ FLAG
    001675  001713 6010 00    3741          TNZ     ICHR1               IF ON. GET LAST CHARACTER
    001676  005462 2350 52    3742          LDA     ITAL.SC             OTHERWISE GET NEXT CHARACTER
    001677  000177 3750 07    3743          ANA     A$MASK.DL           MASK OFF GARBAGE
    001700  001702 6070 00    3744          TTF     *+2                 GOT A CHARACTER
    001701  000004 2350 07    3745          LDA     A$EOT-B$DELIM+B$TERM.DL   FAKE END OF LINE
    001702  000004 1150 07    3746          CMPA    A$EOT-B$DELIM-B$TERM.DL TEST FOR END OF LINE
    001703  001716 6000 00    3747          TZE     ICHR2               YES
    001704  000040 1150 07    3748          CMPA    A$SP-B$DELIM.DL     *CHECK FOR CONTROL CHARACTERS
    001705  001676 6020 00    3749          TNC     ICHR+2              IGNORE THEM
    001706  000077 1150 07    3750          CMPA    A$LA-2.DL           SEE IF SPECIAL
    001707  001711 6030 00    3751          TRC     *+2                 NOPE
    001710  004560 2350 05    3752          LDA     A$ASCTB-32.AL       GET SPECIAL BITS FROM TABLE
    001711  005461 7550 00    3753 ICHR3    STA     LTCHR               UPDATE LAST CHARACTER
    001712  000000 7100 17    3754          TRA     0.L                 RETURN TO CALLER
    001713  005461 2350 00    3755 ICHR1    LDA     LTCHR               GET LAST CHARACTER INSTEAD
    001714  005460 4500 00    3756          STZ     RFLG                TURN OFF THE RE-READ FLAG
    001715  000000 7100 17    3757          TRA     0.L                 RETURN TO CALLER
    001716  600000 2750 07    3758 ICHR2    ORA     B$DELIM+B$TERM.DL *OR IN SPECIAL BITS
    001717  001711 7100 00    3759          TRA     ICHR3
                          3760 *
                          3761 *
                005460    3762          USE     STORE
                005460    3763 RFLG     BSS     1                   RE-READ FLAG
                005461    3764 LTCHR    BSS     1                   LAST CHARACTER READ
                001720    3765          USE     PREVIOUS
```

```
                    001720     3767         USE      CODE                                                           1170
                               3768         HEAD     0                                                              1180
                               3769 *                                                                               1190
                               3770 *                                                                               1200
                               3771 *                                     INBLK                                     1210
                               3772 *                                                                               1220
                               3773 *       GETS NEXT (NON-BLANK) CHARACTER IN AL                                   1230
                               3774 *                                                                               1240
                               3775 *       ENTER WITH                                                              1250
                               3776 *                C(XT) = TCB                                                    1260
                               3777 *                C(XJ) = JCB                                                    1270
                               3778 *       ENTER BY                                                                1280
                               3779 *                TSX L.O$INBLK                                                  1290
                               3780 *       CALLS                                                                   1300
                               3781 *                O$ICHR                                                         1310
                               3782 *       RETURNS TO 0.L                                                          1320
                               3783 *       RETURNS WITH                                                            1330
                               3784 *                C(XT) = TCB                                                    1340
                               3785 *                C(XJ) = JCB                                                    1350
                               3786 *                C(AL) = NEXT NON-BLANK CHARACTER                               1360
                               3787 *                                                                               1370
                               3788 *                                                                               1380
                    001720     3789 INBLK   BSS      0                    ENTRY POINT                               1390
    001720  000021 4470 11     3790         SXL      L.T$TEMP7.T          SAVE RETURN ADDRESS                       1400
                    001721     3791         ICHR                          GET NEXT CHARACTER                        1410
  END OF BINARY CARD IOS00048
    001721  001674 7070 00                  TSX      L.O$ICHR
    001722  200040 1150 07     3792         CMPA     A$SP,DL              TEST FOR A BLANK (SPACE)                  1420
    001723  001721 6000 00     3793         TZE      *-2                  IF SO. IGNORE IT                          1430
    001724  000021 7270 11     3794         LXL      L.T$TEMP7.T          RETRIEVE RETURN ADDRESS                   1440
                    001725     3795         BUGL     (T$TEMP7.T)          BUG IT                                    1450
                    525222          BUGBUG SET      BUGBUG+1
    001725  525222 2200 03                  LDX      0.BUGBUG.DU
    001726  000021 4400 11                  SXL      0.T$TEMP7.T
    001727  000000 7100 17     3796         TRA      0.L                  RETURN TO CALLER                          1460
```

O OPERATOR INTERFACE -- IDELM

```
                 001730   3798        USE     CODE
                          3799        HEAD    O
                          3800 *
                          3801 *
                          3802 *                               IDELM
                          3803 *
                          3804 *      THIS SUBROUTINE LOOKS FOR THE HIGHEST PRIORITY DELIMITER
                          3805 *
                          3806 *      ENTER WITH
                          3807 *              C(XT) = TCB
                          3808 *              C(XJ) = JCB
                          3809 *      ENTER BY
                          3810 *              TSX L.O$IDELM
                          3811 *      CALLS
                          3812 *              O$INBLK
                          3813 *      RETURNS TO 0.L
                          3814 *      RETURNS WITH
                          3815 *              C(XT) = TCB
                          3816 *              C(XJ) = JCB
                          3817 *              C(AL) = DELIMITING CHARACTER
                          3818 *      USES
                          3819 *              NO LOCALS
                          3820 *              T$TEMP5.T (LOWER)
                          3821 *      CLOBBERS C(X0)
                          3822 *
                 001730   3823 IDELM  BSS     0
 001730  000023 4470 11   3824        SXL     L.T$TEMP5.T     SAVE RETURN ADDRESS
 001731  005461 2350 00   3825        LDA     LTCHR           GET BACK LAST CHARACTER
 001732  200000 3150 07   3826        CANA    B$DELIM.DL      MUST HAVE BEEN A DELIMITER
 001733  002105 6000 00   3827        TZE     FORER           IF NOT, FORMAT ERROR
 001734  005460 0540 00   3828        AOS     RFLG            BACK OVER IT
                 001735   3829        INBLK                   GET NEXT NON-BLANK
 001735  001720 7070 00               TSX     L.O$INBLK
 001736  400000 3150 07   3830        CANA    B$TERM.DL       SEE IF TERMINATOR
 001737  001742 6010 00   3831        TNZ     *+3             IF SO, BACK OVER IT
 001740  200000 3150 07   3832        CANA    B$DELIM.DL      SEE IF DELIMITER
 001741  001743 6010 00   3833        TNZ     *+2             IF SO RETURN
 001742  005460 0540 00   3834        AOS     RFLG            MUST HAVE BEEN A SPACE
 001743  000023 7270 11   3835        LXL     L.T$TEMP5.T     RETRIEVE RETURN ADDRESS
                 001744   3836        BUGL    (T$TEMP5.T)     BUG IT
                 525223        BUGBUG SET     BUGBUG+1
 001744  525223 2200 03               LDX     0.BUGBUG.DU
 001745  000023 4400 11               SXL     0.T$TEMP5.T
END OF BINARY CARD IOS00049
 001746  000000 7100 17   3837        TRA     0.L             RETURN TO CALLER
```

```
        O                              OPERATOR INTERFACE -- GET COMMAND

                   001747     3839        USE     CODE                                                        1890
                              3840        HEAD    O                                                           1900
                              3841 *                                                                          1910
                              3842 *                                                                          1920
                              3843 *                                ICMD                                      1930
                              3844 *                                                                          1940
                              3845 *      PICK UP A 4 CHARACTER OR LESS SYMBOL FOR COMMAND NAME.              1950
                              3846 *      THROW AWAY ALL ALPHABETIC CHARACTERS AFTER THE FIRST 4              1960
                              3847 *      UNTIL REACHING A DIGIT, DELIMITER, OR TERMINATOR.                   1970
                              3848 *                                                                          1980
                   001747     3849 ICMD   BSS     0                 ENTRY POINT                               1990
   001747  000022 4470 11     3850        SXL     L,T$TEMP6,T       SAVE RETURN ADDRESS                       2000
   001750  004700 2360 00     3851        LDQ     BLNKS             PAD Q WITH BLANKS                         2010
   001751  000044 2230 03     3852        LDX     Y,36,DU           GET UP TO FOUR 9-BIT CHARACTERS           2020
                   001752     3853        INBLK                     GET FIRST NON-BLANK CHARACTER             2030
   001752  001720 7070 00                 TSX     L,O$INBLK
   001753  001755 7100 00     3854        TRA     *+2               ENTER LOOP                                2040
                   001754     3855 ICMD1  ICHR                      GET NEXT CHARACTER                        2050
   001754  001674 7070 00                 TSX     L,O$ICHR
   001755  740000 3150 07     3856        CANA    B$TERM+B$DELIM+B$DIGIT+B$OPR,DL  SEARCH FOR END OF COMMAND060
   001756  001766 6010 00     3857        TNZ     ICMD2             EXIT, IF FOUND                            2070
   001757  000033 7350 00     3858        ALS     36-9              MOVE TO TOP OF A-REG                      2080
   001760  000011 7770 00     3859        LLR     9                 NOW INTO BOTTOM OF Q-REG                  2090
   001761  000011 1230 03     3860        SBLX    Y,9,DU            DECREMENT 9-BIT CHARACTER COUNTER         2100
   001762  001754 6010 00     3861        TNZ     ICMD1             TEST FOR DONE                             2110
                   001763     3862        ICHR                      DONE, FIND END OF COMMAND                 2120
   001763  001674 7070 00                 TSX     L,O$ICHR
   001764  740000 3150 07     3863        CANA    B$TERM+B$DELIM+B$DIGIT+B$OPR,DL                             2130
   001765  001763 6000 00     3864        TZE     *-2               KEEP SEARCHING FOR END                    2140
   001766  005460 0540 00     3865 ICMD2  AOS     RFLG              SET RE-READ FLAG                          2150
   001767  000000 7760 13     3866        QLR     0,Y               POSITION LEFT JUSTIFIED, BLANKED FILL     2160
   001770  000044 7370 00     3867        LLS     36                MOVE INTO A-REG                           2170
   001771  000022 7270 11     3868        LXL     L,T$TEMP6,T       RETRIEVE RETURN ADDRESS                   2180
                   001772     3869        BUGL    (T$TEMP6,T)       BUG IT                                    2190
                   525224          BUGBUG SET     BUGBUG+1
   001772  525224 2200 03                 LDX     0,BUGBUG,DU
 END OF BINARY CARD IOS00050
   001773  000022 4400 11                 SXL     0,T$TEMP6,T
   001774  000000 7100 17     3870        TRA     0,L               RETURN TO CALLER                          2200
                              3871 *                                                                          2210
                              3872 *                                                                          2220
                   004700     3873        USE     CONST                                                       2230
   004700  040040040040       3874 BLNKS  UASCI   1,                                                          2240
                   001775     3875        USE     PREVIOUS                                                    2250
                              3876 *$*    DISK    OPCON                                                       2260
```

```
                   001775    3878         USE     CODE                                                  110
                             3879         HEAD    0                                                     120
                             3880 *                                                                     130
                             3881 *                                                                     140
                             3882 *                                   ENTER                             150
                             3883 *                                                                     160
                             3884 *       THE OPERATOR INTERFACE IS ENTERED FROM THE NOTIFY SERVICE     170
                             3885 *       ROUTINE ON RECEIPT OF A CAUSE FROM THE JOB STREAM SCHEDULER   180
                             3886 *       TO READ THE CONTROL TTY.  IT SETS UP THE NECESSARY TASK TO    190
                             3887 *       TALK WITH THE OPERATOR, AND EXITS TO CONTINUE NOTIFY SERVICE. 200
                             3888 *                                                                     210
                             3889 *                                                                     220
                   001775    3890 ENTER   BSS     0                                                     230
                   001775    3891         BRANCH  NOPASS,INPUT      CREATE TASK TO CONVERSE WITH OP     240
 001775  001467 7000 00                   TSX     0,T$GETT
 001776  000000 6220 11                   EAX     X,0,T
 001777  000005 2210 12                   LDX     T,T$LINK,X
 002000  000000 6210 12                   EAX     T,0,X
 002001  002011 6200 00                   EAX     0,INPUT
 002002  000004 7400 11                   STX     0,T$TRA,T
 002003  000004 6200 11                   EAX     0,Q$OFFST,T
 002004  005162 7170 00                   XED     Q$XADD+Q$TASK
 002005  000005 2210 12                   LDX     T,T$LINK,X
                   002006                 BUGXR   (0,X)
                   525225         BUGBUG  SET     BUGBUG+1
 002006  525225 2200 03                   LDX     0,BUGBUG,DU
 002007  525225 2220 03                   LDX     X,BUGBUG,DU
 002010  003171 7100 00      3892         TRA     C$NSRVX           RE-ISSUE NOTIFY                     250
```

```
          0                         OPERATOR INTERFACE -- INITIALIZE

                    002011   3894          USE    CODE                                                    270
                             3895          HEAD   0                                                       280
                             3896 *                                                                       290
                             3897 *                                                                       300
                             3898 *                                 INITIALIZATION                        310
                             3899 *                                                                       320
                             3900 *        INITIALIZE IN ORDER TO TALK WITH OPERATOR                      330
                             3901 *                                                                       340
                    002011   3902 INPUT    BSS    0                 ENTER HERE TO SET CONVERSATIONAL MODE 350
                    002011   3903          LOGS                     SEIZE CTY, LOCK IT, SEND IDENTIFIER   360
   002011  001517 7070 00                  TSX    L,0$LOGS
                    002012   3904 INP1     LOG                                                            370
   002012  001645 7070 00                  TSX    L,0$LOG
   002013  000200 2350 03    3905          LDA    IBFSZ*4,DU        GET NUMBER OF CHARACTERS TO READ      380
   002014  005463 7550 00    3906          STA    INCHR             SETUP FOR READ                        390
                             3907 *                                                                       400
                             3908 *        READ INPUT FROM CTY                                            410
                             3909 *                                                                       420
                    002015   3910 INP2     READ   R$OPFRN,(IBUF,DU),INCHR,(2,DU)                          430
END OF BINARY CARD IOS00051
   002015  000536 7000 00                  TSX    0,$READ
   002016  005665 0000 00                  ARG    R$OPFRN
   002017  005464 0000 03                  ARG    IBUF,DU
   002020  005463 0000 00                  ARG    INCHR
   002021  000002 0000 03                  ARG    2,DU
                    002022   3911          CHECK  INP3,B$BZ,INP2,B$TRO,INP2                               440
   002022  000000 7200 11                  LXL    0,T$SRW1,T
   002023  000077 3600 03                  ANX    0,B$STMK,DU
   002024  002032 6000 00                  TZE    INP3
   002025  000003 1000 03                  CMPX   0,B$BZ,DU
   002026  002015 6000 00                  TZE    INP2
   002027  000035 1000 03                  CMPX   0,B$TRO,DU
   002030  002015 6000 00                  TZE    INP2
   002031  777777 7100 00                  TRA    $ERROR
                             3912 *                                                                       450
                             3913 *        INFORMATION READ                                               460
                             3914 *                                                                       470
                    002032   3915 INP3     BSS    0                                                       480
   002032  000000 2350 11    3916          LDA    T$SRW1,T          GET NUMBER OF CHARACTER RECEIVED      490
   002033  777777 3750 03    3917          ANA    -1,DU             ISOLATE NUMBER                        500
   002034  002015 6000 00    3918          TZE    INP2              NOTHING, READ AGAIN                   510
   002035  000014 7710 00    3919          ARL    18-6              MOVE LENGTH TO CHARACTER TALLY POSITION 520
   002036  000140 0750 07    3920          ADA    $TAL+$TALYB,DL    SET TALLY BYTE BIT AND BUMP TALLY ONE 530
   002037  005464 2750 03    3921          ORA    IBUF,DU           SET IN ADDRESS                        540
   002040  005462 7550 00    3922          STA    ITAL              SAVE AS WORKING TALLY                 550
                    002041   3923 INP4     BSS    0                                                       560
   002041  005526 4500 00    3924          STZ    ERCMD             RESET COMMAND ERROR COUNT             570
   002042  005461 4500 00    3925          STZ    LTCHR             RESET LAST CHARACTER RECEIVED         580
END OF BINARY CARD IOS00052
   002043  005460 4500 00    3926          STZ    RFLG              RESET RE-READ FLAG                    590
```

```
         0                              OPERATOR INTERFACE -- INITIALIZE

002044  002045 7100 00        3927         TRA      LSCAN            GO TO LINE SCAN
                              3928 *
                  005462      3929         USE      STORE
                  000040      3930 IBFSZ   EQU      32               LENGTH OF INPUT BUFFER
005462  005464 0024 40        3931 ITAL    TALLYB   IBUF,4*ILEN,0    INPUT TALLY WORD
005463  000000 000000         3932 INCHR   ZERO     **,              NUMBER OF CHARACTERS TO READ
                              3933 *
                              3934 *
                  005464      3935 IBUF    BSS      0                INPUT AREA
005464  107105124040          3936         UASCI    4,GET LP01# EXIT
005470  004004004004          3937         OCT      004004004004     FAKE END OF READ
                  000005      3938 ILEN    EQU      *-IBUF           SIZE OF FAKE READ (WORDS)
                  005471      3939         BSS      IBFSZ-ILEN       RESERVE REST OF BUFFER
005524  004004004004          3940         OCT      004004004004     PAD THE END
                  002045      3941         USE      PREVIOUS
                              3942 *$*     DISK     LSCAN
```

```
                    002045          USE     CODE
                                    HEAD    0
                                *
                                *
                                *                          LINE SCAN
                                *
                                *       THIS ROUTINE RECOGNIZES COMMANDS.  A TERMINATOR
                                *       ON THE LAST COMMAND OF AMPERSAND IS TAKEN TO
                                *       INDICATE A REPETITION OF THAT COMMAND, ELSE
                                *       A NEW COMMAND IS PICKED UP AND EXECUTED.
                                *
                    002045       LSCAN  BSS     0
 002045  005460 4500 00                 STZ     RFLG       CLEAR THE RE-READ FLAG
 002046  005461 2350 00                 LDA     LTCHR      GET LAST CHARACTER
 002047  600046 1150 07                 CMPA    A$AMPER,DL SEE IF REPEAT COMMAND
 002050  005525 6000 51                 TZE     LCOMD,I    YES--DO IT
                    002051       LSCN1  BSS     0
 002051  600004 1150 07                 CMPA    A$EOT,DL   CHECK FOR CARRIAGE RETURN
 002052  002070 6000 00                 TZE     LSCNX      YES--'MORE?'
                    002053              INBLK              GET NEXT NON-BLANK CHARACTER IN A
 002053  001720 7070 00                 TSX     L,O$INBLK
 002054  400000 3150 07                 CANA    B$TERM,DL  SEE IF EXTRA TERMINATOR
 002055  002051 6010 00                 TNZ     LSCN1      IS SO--CHECK IT OUT
END OF BINARY CARD IOS00053
 002056  005460 0540 00                 AOS     RFLG       BACK UP OVER IT
                    002057       LSCN3  BSS     0
                    002057              ICMD               GET COMMAND
 002057  001747 7070 00                 TSX     L,O$ICMD
 002060  005526 0540 00                 AOS     ERCMD      ONE MORE COMMAND
 002061  000000 2220 03                 LDX     X,0,DU     INITIALIZE X FOR RPT
 002062  012300 5202 02                 RPT     CMDLN,2,TZE
 002063  004717 1150 12                 CMPA    CMDTB,X    SEARCH TABLE
 002064  002075 6010 00                 TNZ     CMDER      NO SUCH COMMAND
 002065  777777 2220 12                 LDX     X,-1,X     GET POINTER FROM TABLE
 002066  005525 7420 00                 STX     X,LCOMD    SAVE POINTER TO ROUTINE
 002067  000000 7100 12                 TRA     0,X        BRANCH ON IT
                                *
                                *
                                *       POINTER TO LAST COMMAND FOR REPEATING
                                *
                    005525              USE     STORE
                    525226       BUGBUG SET     BUGBUG+1
 005525  525226 0000 00          LCOMD  ARG     BUGBUG     POINTER TO LAST COMMAND
 005526  000000000000            ERCMD  DEC     0          COMMAND NUMBER CURRENTLY ACTIVE
                    002070              USE     PREVIOUS
```

```
         0                     OPERATOR INTERFACE -- MORE

                002070   3987         USE    CODE                                              540
                         3988         HEAD   0                                                 550
                         3989 *                                                                560
                         3990 *                                                                570
                         3991 *                                MORE?                           580
                         3992 *                                                                590
                         3993 *       THIS ROUTINE ASKS THE OP FOR MORE COMMANDS.              600
                         3994 *       NOTE THAT IT RELEASES AND THEN SEIZES THE CONTROL TTY    610
                         3995 *       TO ALLOW ANY OTHER WAITING TASKS TO LOG THEIR MESSAGES.  620
                         3996 *                                                                630
                002070   3997 LSCNX   BSS    0                                                 640
                002070   3998         LOGX                     RELEASE CONTROL -- ALLOW OTHERS TO SPEAK 650
 002070  001560 7070 00               TSX    L.O$LOGX
                002071   3999         LOGS                     SEIZE CONTROL AGAIN             660
 002071  001517 7070 00               TSX    L.O$LOGS
                002072   4000         LOGC   MORMS             'MORE? '                        670
 002072  004701 2350 00               LDA    MORMS
 002073  001543 7070 00               TSX    L.O$LOGC
 002074  002012 7100 00  4001         TRA    INP1              READ REPLY                      680
                         4002 *                                                                690
                004701   4003         USE    CONST                                             700
 004701  004702 0013 40  4004 MORMS   TALLYB *+1,10+1,0        "'MORE? '                       710
 004702  177177015012    4005         OCT    177177015012      WARM UP TTY                     720
END OF BINARY CARD IOS00054
 004703  115117122105    4006         UASCI  2,MORE?                                           730
                002075   4007         USE    PREVIOUS                                          740
```

0 OPERATOR INTERFACE -- COMMAND/FORMAT ERRORS

```
                       002075   4009          USE     CODE                                          760
                                4010          HEAD    0                                             770
                                4011 *                                                              780
                                4012 *                                                              790
                                4013 *                          CMDER                               800
                                4014 *                                                              810
                                4015 *        THIS ROUTINE IS ENTERED WHEN A COMMAND CAN NOT BE RECOGNIZED.  820
                                4016 *        IT TELLS WHICH COMMAND WAS INVALID (BY NUMBER) AND ASKS     830
                                4017 *        FOR MORE INPUT.                                       840
                                4018 *                                                              850
                       002075   4019 CMDER    BSS     0                                             860
                       002075   4020          LOGC    CERR              'COMMAND ERROR #'           870
    002075  004705 2350 00                    LDA     CERR
    002076  001543 7070 00                    TSX     L.O$LOGC
    002077  005526 2350 00      4021          LDA     ERCMD             GET NUMBER OF THIS COMMAND  880
    002100  000007 3750 07      4022          ANA     7,DL              ISOLATE NUMBER              890
    002101  100060 2750 07      4023          ORA     A$D0,DL           MAP INTO ASCII DIGIT        900
                       002102   4024          LCHR                      PRINT IT                    910
    002102  001620 7070 00                    TSX     L.O$LCHR
                       002103   4025          LOG                       FLUSH BUFFER                920
    002103  001645 7070 00                    TSX     L.O$LOG
    002104  002070 7100 00      4026          TRA     LSCNX             MORE?                       930
                                4027 *                                                              940
                       004705   4028          USE     CONST                                         950
    004705  004706 0024 40      4029 CERR     TALLYB  *+1,19+1,0        'COMMAND ERROR #'           960
    004706  177177015012        4030          OCT     177177015012                                  970
    004707  103117115115        4031          UASCI   4,COMMAND ERROR #                             980
                       002105   4032          USE     PREVIOUS                                      990
                                4033 *                                                              1000
                                4034 *                                                              1010
                                4035 *                          FORER                               1020
                                4036 *                                                              1030
                                4037 *        THIS ROUTINE IS ENTERED WHEN ONE OR MORE PARAMETERS FOR A  1040
                                4038 *        VALID COMMAND CAN NOT BE RECOGNIZED.  IT SCREAMS AND THEN  1050
                                4039 *        CALLS  'CMDER'.                                       1060
                                4040 *                                                              1070
                       002105   4041 FORER    BSS     0                 FORMAT ERROR                1080
                       002105   4042          LOGC    FERR              FORMAT ERROR                1090
    002105  004713 2350 00                    LDA     FERR
    002106  001543 7070 00                    TSX     L.O$LOGC
    002107  002075 7100 00      4043          TRA     CMDER             EXIT TO COMMAND ERROR       1100
                                4044 *                                                              1110
                       004713   4045          USE     CONST                                         1120
 END OF BINARY CARD IOS00055
    004713  004714 0021 40      4046 FERR     TALLYB  *+1,16+1,0        'FORMAT ERROR'              1130
    004714  177177015012        4047          OCT     177177015012                                  1140
    004715  106117122115        4048          UASCI   2,FORMAT ERROR                                1150
                       002110   4049          USE     PREVIOUS                                      1160
                                4050 *$*      DISK    COMMANDS                                      1170
```

0 OPERATOR INTERFACE -- COMMAND TABLE

```
                   002110          USE     CODE
                                   HEAD    0
                             *
                             *
                             *                             COMMAND TABLE
                             *
                             *     THESE COMMANDS ARE IN ALPHABETICAL ORDER
                             *
                   004717          USE     CONST
                   004717    CMDTB BSS     0               BACKWARD COMMAND TABLE
004717 105130111124                UASCI   1,EXIT          EXIT -- RELEASE TTY
004720 002372 0000 00              ARG     OPX
004721 107105124040                UASCI   1,GET           REOPEN CLOSED PERIPHERAL
004722 002110 0000 00              ARG     GET
004723 113111114114                UASCI   1,KILL          KILL A PERIPHERAL DEVICE
004724 002220 0000 00              ARG     KILL
004725 122105114105                UASCI   1,RELEASE       CLOSE A PERIPHERAL
004726 002253 0000 00              ARG     REL
004727 122105123124                UASCI   1,RESTART       RESTART A PERIPHERAL
004730 002277 0000 00              ARG     STRT
                   000012    CMDLX EQU     *-CMDTB         TABLE LENGTH
                   000005    CMDLN EQU     CMDLX/2         NUMBER OF TABLE ENTRIES
                   002110          USE     PREVIOUS
                             *$*   DISK    OGET
```

```
                          002110   4077        USE     CODE
                                   4078        HEAD    0
                                   4079 *
                                   4080 *
                                   4081 *                                GET COMMAND
                                   4082 *
                                   4083 *      RE-OPENS THE PERIPHERAL IDENTIFIED BY THE FOUR LETTER
                                   4084 *      ABBREVIATION.  THE FORMAT IS:
                                   4085 *
                                   4086 *      GET <PERIPHERAL-NAME>:...:<PERIPHERAL-NAME>
                                   4087 *
                                   4088 *
                          002110   4089 GET    BSS     0
    002110  002332 7070 00         4090        TSX     L.PERI            GET PERIPHERAL INFORMATION
    002111  000023 7420 11         4091        STX     X.T$TEMP5.T       SAVE TYPE AND
    002112  000023 4440 11         4092        SXL     Z.T$TEMP5.T       DEVICE POINTERS
                                   4093 *
                                   4094 *      CHECK IF CURRENTLY CLOSED
                                   4095 *
    002113  000002 2350 14         4096        LDA     R$FLAG.Z          GET PERIPHERAL FLAGS
    002114  200000 3150 03         4097        CANA    R$CLOSE.DU        CHECK IF CLOSED
    002115  002206 6000 00         4098        TZE     GET5              IGNORE COMMAND IF NOT
 END OF BINARY CARD IOS00056
    002116  000000 2350 14         4099        LDA     R$ABBR.Z          GET ITS NAME
    002117  005157 1150 00         4100        CMPA    Q$OP00+Q$ABBR     DON'T ALLOW OP TO
    002120  002206 6000 00         4101        TZE     GET5              FOOL WITH HIS CONSOLE
    002121  005535 7550 00         4102        STA     PTN+6             CONSTRUCT TREE-NAME
    002122  000002 2200 12         4103        LDX     0.R$ELT.X         GET ELEMENT SIZE
    002123  000027 7400 11         4104        STX     0.T$TEMP1.T       SAVE FOR OPEN
                          002124   4105 GET1   OPEN    (PTN.DU).(PTS.DU).(1.DU).(T$TEMP1.T).(0.DU)
    002124  000667 7000 00                     TSX     0.$OPEN
    002125  005527 0000 03                     ARG     PTN.DU
    002126  000014 0000 03                     ARG     PTS.DU
    002127  000001 0000 03                     ARG     1.DU
    002130  000027 0000 11                     ARG     T$TEMP1.T
    002131  000000 0000 03                     ARG     0.DU
                          002132   4106        CHECK   GET2.B$BZ.GET1.B$LOCK.GET4
    002132  000000 7200 11                     LXL     0.T$SRW1.T
    002133  000077 3600 03                     ANX     0.B$STMK.DU
    002134  002142 6000 00                     TZE     GET2
    002135  000003 1000 03                     CMPX    0.B$BZ.DU
    002136  002124 6000 00                     TZE     GET1
    002137  000013 1000 03                     CMPX    0.B$LOCK.DU
    002140  002204 6000 00                     TZE     GET4
    002141  777777 7100 00                     TRA     $ERROR
                                   4107 *
                                   4108 *      PERIPHERAL OPENED
                                   4109 *
                          002142   4110 GET2   BSS     0
    002142  000023 2220 11         4111        LDX     X.T$TEMP5.T       RESTORE PERIPHERAL POINTERS
```

```
          0                          OPERATOR INTERFACE -- GET COMMAND

     002143  000023 7240 11      4112          LXL      Z,T$TEMP5,T                                         460
  END OF BINARY CARD IOS00057
     002144  000000 2200 11      4113          LDX      0,T$SRW1,T        GET FRN                           470
     002145  000001 7400 14      4114          STX      0,R$FRN,Z         SAVE IN TABLE                     480
     002146  000025 7400 11      4115          STX      0,T$TEMP3,T       SAVE FOR PASS IN CAUSE            490
     002147  000004 0540 12      4116          AOS      R$OMAX,X          BUMP OUR MAX COUNT                500
     002150  000005 0540 12      4117          AOS      R$AVAIL,X         BUMP NUMBER AVAILABLE             510
                      002151     4118          DECRM    (R$OPER,X)        DECREMENT OPERATOR HOLDINGS       520
     002151  000001 3360 07                    LCQ      1,DL
     002152  000006 0560 12                    ASQ      R$OPER,X
     002153  777777 6040 00      4119          TMI      $ERROR            ***PROBLEM                        530
     002154  077777 2200 03      4120          LDX      0,-1-R$BUSY-R$CLOSE-R$RSVE,DU *SET OFF A FEW FLAGS  540
     002155  000002 3400 14      4121          ANSX     0,R$FLAG,Z        IN THE PERIPHERAL TABLE           550
                                 4122 *                                                                     560
                                 4123 *        SEND PERIPHERAL TO SUBMODULE                                 570
                                 4124 *                                                                     580
     002156  000001 2350 12      4125          LDA      R$STATE,X         GET STATE                         590
     002157  000027 7550 11      4126          STA      T$TEMP1,T         SAVE FOR CAUSE                    600
     002160  000000 2350 14      4127          LDA      R$ABBR,Z          GET NAME OF DEVICE                610
     002161  000007 3750 07      4128          ANA      7,DL              MASK TO UNIT NUMBER               620
     002162  000022 7350 00      4129          ALS      18                MOVE TO AU                        630
     002163  002000 2750 03      4130          ORA      B$,GET,DU         COMMAND:  GET                     640
     002164  000026 7550 11      4131          STA      T$TEMP2,T         SAVE AS MESSAGE                   650
     002165  000002 7200 12      4132          LXL      0,R$ACC,X         GET ACCESSES TO PASS              660
     002166  000024 7400 11      4133          STX      0,T$TEMP4,T                                         670
                      002167     4134 GET3     BSS      0                                                   680
                      002167     4135          CAUSE    C$FRN2,(1,DU),(T$TEMP1,T),(T$TEMP2,T)               690
                      002167     4136          ETC      (T$TEMP3,T),(T$TEMP4,T)                             700
     002167  000767 7000 00                    TSX      0,$CAUSE
     002170  006221 0000 00                    ARG      C$FRN2
     002171  000001 0000 03                    ARG      1,DU
  END OF BINARY CARD IOS00058
     002172  000027 0000 11                    ARG      T$TEMP1,T
     002173  000026 0000 11                    ARG      T$TEMP2,T
     002174  000025 0000 11                    ARG      T$TEMP3,T
     002175  000024 0000 11                    ARG      T$TEMP4,T
                      002176     4137          CHECK    GET4,B$BZ,GET3 CHECK STATUS                         710
     002176  000000 7200 11                    LXL      0,T$SRW1,T
     002177  000077 3600 03                    ANX      0,B$STMK,DU
     002200  002204 6000 00                    TZE      GET4
     002201  000003 1000 03                    CMPX     0,B$BZ,DU
     002202  002167 6000 00                    TZE      GET3
     002203  777777 7100 00                    TRA      $ERROR
                      002204     4138 GET4     BSS      0                                                   720
     002204  000000 2220 11      4139          LDX      X,T$SRW1,T        GET NUMBER OF PEOPLE NOTIFIED     730
     002205  002167 6000 00      4140          TZE      GET3              NONE, RE-SEND                     740
                      002206     4141 GET5     BSS      0                                                   750
                                 4142 *                                                                     760
                                 4143 *        PERIPHERAL PASSED, TELL OPERATOR                             770
                                 4144 *                                                                     780
```

```
           C                             OPERATOR INTERFACE -- GET COMMAND

                          002206     4145          LOGC    OPMS              TELL OPERATOR PERIPHERAL OPENED:          790
     002206  004731 2350 00                        LDA     OPMS
     002207  001543 7070 00                        TSX     L.0$LOGC
                          004731     4146          USE     CONST                                                       800
     004731  004732 0030 40          4147 OPMS     TALLYB  *+1,23+1,0        'PERIPHERAL OPENED:  '                    810
     004732  177177015012            4148          OCT     177177015012                                                820
     004733  120105122111            4149          UASCI   5,PERIPHERAL OPENED:                                        830
                          002210     4150          USE     PREVIOUS                                                    840
  END OF BINARY CARD IOS00059
     002210  000023 7240 11          4151          LXL     Z,T$TEMP5,T       RESTORE DEVICE POINTER                    850
     002211  000000 6350 14          4152          EAA     R$ABBR,Z          POINT TO ABBREVIATION                     860
     002212  000540 2750 07          4153          ORA     4*$TAL+$TAL+$TALYB,DL  ABBREVIATIONS ARE FOUR CHARACTERS    870
                          002213     4154          LOGC    A                                                           880
     002213  001543 7070 00                        TSX     L.0$LOGC
                                     4155 *                                                                            890
                                     4156 *        OPERATOR INFORMED. SEE IF MORE                                      900
                                     4157 *                                                                            910
                          002214     4158          IDELM                     GET DELIMITER                             920
     002214  001730 7070 00                        TSX     L.0$IDELM
     002215  200073 1150 07          4159          CMPA    A$SCOL,DL         SEMI-COLON?                               930
     002216  002110 6000 00          4160          TZE     GET               GET NEXT                                  940
     002217  002045 7100 00          4161          TRA     LSCAN             OTHERWISE GET NEXT COMMAND                950
                                     4162 *                                                                            960
                                     4163 *                                                                            970
                          005527     4164          USE     STORE                                                       980
                          005527     4165 PTN      BSS     0                 PERIPHERAL TREE-NAME                      990
     005527  104105126111            4166          UASCI   6,DEVICE                                                    1000
     005535  117120060060            4167          UASCI   1,OP00            DEVICE TO BE OPENED GOES HERE             1010
     005536  040040040040            4168          UASCI   5,                                                          1020
                          000014     4169 PTS      EQU     *-PTN             TREE-SIZE                                 1030
                          002220     4170          USE     PREVIOUS                                                    1040
                                     4171 *$*      DISK    OKILL                                                       1050
```

```
                   002220   4173         USE     CODE
                            4174         HEAD    0
                            4175 *
                            4176 *
                            4177 *                                KILL COMMAND
                            4178 *
                            4179 *       FORCES THE IMMEDIATE TERMINATION OF THE PERIPHERAL IDENTIFIED
                            4180 *       BY THE FOUR LETTER ABBREVIATION.  THE FORMAT IS:
                            4181 *
                            4182 *       KILL <PERIPHERAL-NAME>;...;<PERIPHERAL-NAME>
                            4183 *
                   002220   4184 KILL    BSS     0
 END OF BINARY CARD IOS00060
     002220  002332 7070 00  4185        TSX     L,PERI            GET PERIPHERAL INFORMATION
     002221  000023 7420 11  4186        STX     X,T$TEMP5,T       SAVE TYPE AND
     002222  000023 4440 11  4187        SXL     Z,T$TEMP5,T       DEVICE POINTERS
                            4188 *
                            4189 *       CHECK IF PERIPHERAL IS IN USE
                            4190 *
     002223  000002 2350 14  4191        LDA     R$FLAG,Z          GET PERIPHERAL FLAG
     002224  400000 3150 03  4192        CANA    R$BUSY,DU         CHECK IF BUSY
     002225  002250 6000 00  4193        TZE     KILL1             EXIT IF NOT
                            4194 *
                            4195 *       SEND MESSAGE TO KILL THIS JOB
                            4196 *
                            4197                                   SEND MESSAGE TO ABORT JOB IN PROGRESS
     002226  000001 2350 12  4198        LDA     R$STATE,X         GET STATE FOR CAUSE
     002227  000027 7550 11  4199        STA     T$TEMP1,T         SAVE FOR COMMUNICATIONS ROUTINE
     002230  000000 2350 14  4200        LDA     R$ABBR,Z          GET NAME OF PERIPHERAL TO CLOSE
     002231  000007 3750 07  4201        ANA     7,DL              MASK TO UNIT NUMBER
     002232  000022 7350 00  4202        ALS     18                POSITION IN MESSAGE FIELD
     002233  004000 2750 03  4203        ORA     B$,KILL,DU        COMMAND:  KILL
     002234  000026 7550 11  4204        STA     T$TEMP2,T         SAVE FOR COMMUNICATIONS
                   002235   4205         BRANCH  PASS,C$MESSX      CREATE TASK TO SEND MESSAGE
     002235  001467 7000 00              TSX     0,T$GETT
     002236  000000 6220 11              EAX     X,0,T
     002237  000005 2210 12              LDX     T,T$LINK,X
     002240  003140 6200 00              EAX     0,C$MESSX
     002241  000004 7400 11              STX     0,T$TRA,T
     002242  000004 6200 11              EAX     0,Q$OFFST,T
     002243  005162 7170 00              XED     Q$XADD+Q$TASK
     002244  000000 6210 12              EAX     T,0,X
                   002245                BUGXR   (0,X)
                   525227        BUGBUG  SET     BUGBUG+1
     002245  525227 2200 03              LDX     0,BUGBUG,DU
 END OF BINARY CARD IOS00061
     002246  525227 2220 03              LDX     X,BUGBUG,DU
                            4206 *
                            4207 *       MESSAGE SENT, SEE IF MORE
                            4208 *
```

OPERATOR INTERFACE -- KILL COMMAND

```
                           002247     4209         IDELM                    GET DELIMITER
  002247   001730 7070 00                          TSX      L,0$IDELM
                           002250     4210 KILL1   BSS      0
  002250   200073 1150 07             4211         CMPA     A$SCOL,DL       SEMI-COLON?
  002251   002220 6000 00             4212         TZE      KILL            YES, GET NEXT PERIPHERAL NAME
  002252   002045 7100 00             4213         TRA      LSCAN           NO, GET NEXT COMMAND
                                      4214 *$*     DISK     OREL
```

0 OPERATOR INTERFACE -- CLOSE COMMAND

```
                    002253      USE     CODE
                                HEAD    0
                           *
                           *
                           *                              CLOSE COMMAND
                           *
                           *            CLOSES THE PERIPHERAL IDENTIFIED BY THE FOUR LETTER
                           *            ABBREVIATION.  THE FORMAT IS:
                           *
                           *            RELE <PERIPHERAL-NAME>; ...;<PERIPHERAL-NAME>
                           *
                    002253     REL      BSS     0
  002253  002332 7070 00                TSX     L.PERI          GET PERIPHERAL INFORMATION
  002254  000023 7420 11                STX     X.T$TEMP5.T     SAVE TYPE AND
  002255  000023 4440 11                SXL     Z.T$TEMP5.T     DEVICE POINTERS
                           *
                           *            CHECK FOR ALREADY CLOSED
                           *
  002256  000002 2350 14                LDA     R$FLAG.Z        GET PERIPHERAL FLAGS
  002257  200000 3150 03                CANA    R$CLOSE.DU      CHECK IF CLOSED
  002260  002273 6010 00                TNZ     REL2            EXIT IF SO
  002261  100000 2750 03                ORA     R$RSVE.DU       ASK TO HAVE IT RESERVED
  002262  000002 7550 14                STA     R$FLAG.Z        RESTORE FLAGS
  002263  400000 3150 03                CANA    R$BUSY.DU       CHECK IF CURRENTLY BUSY
  002264  002273 6010 00                TNZ     REL2            BUSY--IT WILL CLOSE EVENTUALLY
                           *
                           *            DEVICE IDLE -- SEIZE IT
                           *
                    002265     REL1     BSS     0
  002265  500000 2350 03                LDA     R$BUSY+R$RSVE.DU *MARK IT BUSY AND RESERVED
  002266  000002 2550 14                ORSA    R$FLAG.Z
  002267  000002 4460 14                SXL     J.R$ALLC.Z      AND ALLOCATED TO US
  002270  000006 4460 11                SXL     J.T$JCB.T       FUDGE
  002271  000023 2350 11                LDA     T$TEMP5.T       GET RELEASE PARAMTERS
  002272  002447 7070 00                TSX     L.R$RELP        CALL RELEASE
                           *
                           *            PERIPHERAL RELEASED. CHECK IF MORE
                           *
                    002273     REL2     BSS     0
                    002273              IDELM                   GET DELIMITER
END OF BINARY CARD IOS00062
  002273  001730 7070 00                TSX     L.O$IDELM
  002274  200073 1150 07                CMPA    A$SCOL.DL       SEMI-COLON?
  002275  002253 6000 00                TZE     REL             YES. GET NEXT PERIPHERAL NAME
  002276  002045 7100 00                TRA     LSCAN           NO. GET NEXT COMMAND
                           *$*          DISK    OSTRT
```

```
       0                          OPERATOR INTERFACE -- START COMMAND

                     002277    4261        USE     CODE
                               4262        HEAD    0
                               4263 *
                               4264 *
                               4265 *                         START COMMAND
                               4266 *
                               4267 *     STARTS (RESTARTS) THE PERIPHERAL IDENTIFIED BY THE FOUR LETTER
                               4268 *     ABBREVIATION.  THE FORMAT IS:
                               4269 *
                               4270 *     START <PERIPHERAL-NAME>:...:<PERIPHERAL-NAME>
                               4271 *
                     002277    4272 STRT   BSS     0
   002277  002332 7070 00      4273        TSX     L.PERI           GET PERIPHERAL INFORMATION
   002300  000023 7420 11      4274        STX     X.T$TEMP5.T      SAVE TYPE AND
   002301  000023 4440 11      4275        SXL     Z.T$TEMP5.T      DEVICE PTRS
                               4276 *
                               4277 *     CHECK IF PERIPHERAL IS IN USE
                               4278 *
   002302  000002 2350 14      4279        LDA     R$FLAG.Z         GET PERIPHERAL FLAG
   002303  400000 3150 03      4280        CANA    R$BUSY.DU        CHECK IF BUSY
   002304  002326 6000 00      4281        TZE     STRT2            EXIT IF NOT
                               4282 *
                               4283 *     GET RESTART ADDRESS (CURRENTLY NOT IMPLEMENTED)
                               4284 *
                               4285 *
                               4286 *     SEND MESSAGE TO SUBMODULE TO RESTART
                               4287 *
   002305  000001 2350 12      4288        LDA     R$STATE.X        GET STATE FOR CAUSE
   002306  000027 7550 11      4289        STA     T$TEMP1.T        SAVE FOR COMMUNICATIONS ROUTINE
   002307  000000 2350 14      4290        LDA     R$ABBR.Z         GET NAME OF PERIPHERAL TO RESTART
   002310  000007 3750 07      4291        ANA     7.DL             MASK TO UNIT NUMBER
   002311  000022 7350 00      4292        ALS     18               POSITION IN MESSAGE FIELD
   002312  012000 2750 03      4293        ORA     B$.STRT.DU       COMMAND: RESTART
   002313  000026 7550 11      4294        STA     T$TEMP2.T        SAVE MESSAGE
                     002314    4295        BRANCH  PASS.C$MESSX     CREATE TASK TO SEND MESSAGE
   002314  001467 7000 00              TSX     0.T$GETT
   002315  000000 6220 11              EAX     X.0.T
   002316  000005 2210 12              LDX     T.T$LINK.X
   002317  003140 6200 00              EAX     0.C$MESSX
 END OF BINARY CARD IOS00063
   002320  000004 7400 11              STX     0.T$TRA.T
   002321  000004 6200 11              EAX     0.Q$OFFST.T
   002322  005162 7170 00              XED     Q$XADD+Q$TASK
   002323  000000 6210 12              EAX     T.0.X
                     002324            BUGXR   (0.X)
                     525230     BUGBUG SET     BUGBUG+1
   002324  525230 2200 03              LDX     0.BUGBUG.DU
   002325  525230 2220 03              LDX     X.BUGBUG.DU
                     002326    4296 STRT2  BSS     0
                               4297 *
```

```
          0                                OPERATOR INTERFACE -- START COMMAND

                                   4298 *      MESSAGE SENT, SEE IF MORE                                          480
                                   4299 *                                                                          490
                        002326     4300         IDELM                          GET DELIMITER                      500
      002326  001730 7070 00                    TSX      L.0$IDELM
      002327  200073 1150 07       4301         CMPA     A$SCOL.DL             SEMI-COLON?                        510
      002330  002277 6000 00       4302         TZE      STRT                  YES. GET NEXT PERIPHERAL NAME      520
      002331  002045 7100 00       4303         TRA      LSCAN                 NO. GET NEXT COMMAND               530
                                   4304 *$*     DISK     OPCON2                                                   540
```

```
                     4306          HEAD     0
          002332     4307          USE      CODE
                     4308 *
                     4309 *
                     4310 *                                 PERI SUBROUTINE
                     4311 *
                     4312 *        GET PERIPHERAL ABBREVIATION FROM INPUT STREAM.  IT SETS
                     4313 *        C(XX) TO THE DEVICE HEADER AND C(XZ) TO THE DEVICE TABLE
                     4314 *        ENTRY.  IF IT FINDS NO SUCH DEVICE, IT LOGS A MESSAGE
                     4315 *        SAYING SO.
                     4316 *
                     4317 *        PERIPHERAL ABBREVIATION IS OF THE FORM:
                     4318 *        <PER. ABBR.>:='2 ALPHABETICS'+'2 DIGITS' (E.G. LP01)
                     4319 *
                     4320 *        RETURNS WITH
                     4321 *                 C(XX) = PTR TO DEVICE HEADER
                     4322 *                 C(XZ) = PTR TO DEVICE UNIT
                     4323 *
          002332     4324 PERI     BSS      0
002332  000022 4470 11   4325      SXL      L,T$TEMP6,T     SAVE RETURN ADDRESS
002333  000044 3230 03   4326      LCX      Y,36,DU         LOOK FOR FOUR 9-BIT CHARACTERS
          002334     4327          INBLK                    GET FIRST NON-BLANK CHARACTER
002334  001720 7070 00             TSX      L,O$INBLK
002335  002337 7100 00   4328      TRA      *+2             ENTER LOOP
          002336     4329 PERI1    ICHR                     GET NEXT CHARACTER
002336  001674 7070 00             TSX      L,O$ICHR
002337  640000 3150 07   4330      CANA     B$TERM+B$DELIM+B$OPR,DL 'TEST FOR DELIMITER
002340  002346 6010 00   4331      TNZ      PERI2           END OF ABBREVIATION
002341  000033 7350 00   4332      ALS      36-9            MOVE TO AU (LEFT JUSTIFIED)
002342  000011 7770 00   4333      LLR      9               NOW TO BOTTOM OF Q
002343  000011 0230 03   4334      ADLX     Y,9,DU          BUMP 9-BIT COUNTER
002344  002336 6040 00   4335      TMI      PERI1           LOOK IF MORE
END OF BINARY CARD IOS00064
002345  002347 7100 00   4336      TRA      *+2             DON'T BACK UP
002346  005460 0540 00   4337 PERI2 AOS     RFLG            BACK OVER LAST CHARACTER
          002347     4338          INBLK                    GET NEXT NON-BLANK
002347  001720 7070 00             TSX      L,O$INBLK
002350  600000 3150 07   4339      CANA     B$TERM+B$DELIM,DL 'LOOK FOR TERMINATOR OR DELIMITER
002351  002105 6000 00   4340      TZE      FORER           FORMAT ERROR
                     4341 *
                     4342 *        FIND TABLE ENTRY
                     4343 *
002352  005540 2220 03   4344      LDX      X,R$DEVHP-R$DHLEN,DU  POINT TO DEVICE HEADERS
          002353     4345 PERI3    BSS      0               LOOP HERE
002353  000010 0220 03   4346      ADLX     X,R$DHLEN,DU    BUMP TO NEXT HEADER
002354  005620 1020 03   4347      CMPX     X,R$XDVHR,DU    TEST FOR DONE
002355  002105 6030 00   4348      TRC      FORER           FORMAT ERROR
002356  000000 2240 12   4349      LDX      Z,R$PTR,X       GET PTR TO UNIT TABLE .
002357  000003 2350 12   4350      LDA      R$MAX,X         GET NUMBER TO SEARCH
002360  000012 7350 00   4351      ALS      18-8            SET UP REPEAT
```

```
         0                              OPERATOR INTERFACE -- PERI SUBROUTINE

002361  001100 6200 05        4352          EAX     0,B$ABIT+B$IZE,AL *TERMINATE CONDITIONS     570
002362  000000 5202 03        4353          RPTX    ,R$DEVLN        LOOK FOR ABBREVIATION       580
002363  000000 1160 14        4354          CMPQ    R$ABBR,Z                                    590
002364  002353 6010 00        4355          TNZ     PERI3           LOOP IF NO MATCH            600
002365  000003 1240 03        4356          SBLX    Z,R$ABBR+R$DEVLN,DU *RESET INDEX            610
002366  000022 7270 11        4357          LXL     L,T$TEMP6,T     RETRIEVE RETURN ADDRESS     620
                    002367    4358          BUGL    (T$TEMP6,T)     BUG IT                      630
                    525231         BUGBUG SET     BUGBUG+1
002367  525231 2200 03                      LDX     0,BUGBUG,DU
002370  000022 4400 11                      SXL     0,T$TEMP6,T
002371  000000 7100 17        4359          TRA     0,L             RETURN TO CALLER            640
                              4360 *$*      DISK    OPEXIT                                      650
```

```
                 002372     4362         USE     CODE                                         110
                            4363         HEAD    0                                            120
                            4364 *                                                            130
                            4365 *                                                            140
                            4366 *                               EXIT                         150
                            4367 *                                                            160
                            4368 *       VARIOUS WAYS OF EXITING THE OPERATOR INTERFACE       170
                            4369 *                                                            180
                 002372     4370 OPX     BSS     0                                            190
                 002372     4371         LOGC    MOK             LOG OK MESSAGE               200
 END OF BINARY CARD IOS00065
  002372  004740 2350 00                 LDA     MOK
  002373  001543 7070 00                 TSX     L.O$LOGC
                 002374     4372         LOGX                    SEND MESAGE                  210
  002374  001560 7070 00                 TSX     L.O$LOGX
                 002375     4373         RELT                    RELEASE TRAP BLOCK           220
  002375  001477 7000 00                 TSX     O.T$RELT
                 002376     4374         EXIT                    AND EVAPORATE                230
  002376  003074 7100 00                 TRA     $EXIT
                            4375 *                                                            240
                            4376 *                                                            250
                 004740     4377         USE     CONST                                        260
  004740  004741 0010 40    4378 MOK     TALLYB  *+1,8           OK MESSAGE TALLY             270
  004741  177015012117      4379         OCT     177015012117,113015012177  RB CR LF O K CR LF RB  280
                 002377     4380         USE     PREVIOUS                                     290
                            4381 *$*     DISK    PERMAC                                       300
```

```
002377         USE     CODE
               HEAD    R
*
*
*                                      DESCRIPTION
*
*              THESE MACROS GET A RELEASE A SINGLE PERIPHERAL
*
*
*                                      GETP
*
*              GET A PERIPHERAL
*
GETP           MACRO   TYPE/ 'A'
               INE     '#1','A'
               LDA     #1              GET TYPE IN AU
               TSX     L,R$GETP        CALL SUBROUTINE
               ENDM    GETP
*
*
*                                      RELP MACRO
*
*              RELEASE A PERIPHERAL
*
RELP           MACRO   DEVICE NUMBER/ 'A'
               INE     '#1','A'
               LDA     #1              GET DEVICE NUMBER IN AL
               TSX     L,R$RELP        CALL SUBROUTINE
               ENDM    RELP
*$*            DISK    GETP
```

```
                 002377             USE     CODE
                                    HEAD    R
                              *
                              *
                              *                         GETP
                              *
                              *     GETP GETS A SINGLE PERIPHERAL UNIT OF A SPECIFIED TYPE.
                              *     IF THERE ARE MORE UNITS OF THE SAME TYPE STILL AVAILABLE
                              *     AFTER THE GET, THEN THE NEXT JOB WAITING FOR THE SAME
                              *     PERIPHERAL TYPE IS AWAKENED.
                              *
                              *     CALL BY
                              *             TSX L,R$GETP
                              *     CALL WITH
                              *             C(J) = JOB NUMBER
                              *             C(T) = TBLOCK ADDRESS
                              *             C(L) = RETURN ADDRESS
                              *             C(AU) = PERIPHERAL TYPE
                              *     CALLS
                              *             NONE
                              *     RETURNS WITH
                              *             C(J) = JOB NUMBER
                              *             C(T) = TBLOCK ADDRESS
                              *             C(AU) = PTR TO DEVICE HEADER
                              *             C(AL) = DEVICE NUMBER
                              *     EXIT TO 0,L
                              *     USES
                              *             NO LOCAL TEMPORARIES
                              *             T$TEMP1,T$TEMP2
                              *
                 002377         GETP    BSS     0               ENTRY POINT
  002377  000004 4470 11                SXL     L,T$TRA,T       SAVE RETURN ADDRESS
  002400  000027 7550 11                STA     T$TEMP1,T       SAVE TYPE
  002401  000027 2240 11                LDX     Z,T$TEMP1,T     GET PERIPHERAL TYPE
  002402  000007 3640 03                ANX     Z,7,DU          MASK TO TYPE ONLY
  002403  004743 2240 14                LDX     Z,TABLE,Z       POINT TO PERIPHERAL TYPE TABLE
  002404  777777 6000 00                TZE     $ERROR          NO SUCH PERIPHERAL
  002405  000027 7440 11                STX     Z,T$TEMP1,T     SAVE DEVICE HEADER PTR
                              *
                              *     LOOP TO LOOK FOR FREE DEVICE
                              *
  002406  000003 3360 14                LCQ     MAX,Z           GET NUMBER TO CHECK COMPLEMENTED
  002407  777777 6000 00                TZE     $ERROR          ***BLEWIT
  002410  000000 2200 14                LDX     0,PTR,Z         GET POINTER TO UNITS
  002411  700000 2350 03                LDA     BUSY+CLOSE+RSVE,DU *GET BITS TO CHECK
  002412  000002 3150 10        GETP1   CANA    FLAG,0          IS THIS UNIT FREE?
END OF BINARY CARD IOS00066
  002413  002420 6000 00                TZE     GETP2           YES, TAKE IT
  002414  000003 0200 03                ADLX    0,R$DEVLN,DU    STEP TO NEXT DEVICE
  002415  000001 0760 07                ADQ     1,DL            TEST FOR DONE
```

```
002416  002412 6040 00      4463          TMI    GETP1           NO. SO CONTINUE SEARCH               600
002417  777777 7100 00      4464          TRA    $ERROR          ***SOMEBODY FORGOT TO RESERVE 'EM!   610
                            4465 *                                                                 620
                            4466 *                                                                 630
                            4467 *        FOUND THE REQUESTED UNIT                                 640
                            4468 *                                                                 650
                002420      4469 GETP2    BSS    0                                                 660
002420  000027 4400 11      4470          SXL    0,T$TEMP1,T     SAVE POINTER TO DEVICE               670
002421  000002 4460 10      4471          SXL    J,ALLC,0        MARK IT ALLOCATED TO US              680
002422  400000 2220 03      4472          LDX    X,BUSY,DU       SET THE BUSY BIT ON                  690
002423  000002 2420 10      4473          ORSX   X,FLAG,0                                          700
                002424      4474          DECRM  (AVAIL,Z)       DECREMENT NUMBER NOW FREE            710
002424  000001 3360 07                    LCQ    1,DL
002425  000005 0560 14                    ASQ    AVAIL,7
002426  777777 6040 00      4475          TMI    $ERROR          ***BLEWIT                            720
002427  000027 2350 11      4476          LDA    T$TEMP1,T       GET RETURN WORDS FOR USER            730
                002430      4477          BUGXR  (X,Y,Z,Q)                                         740
                525232           BUGBUG   SET    BUGBUG+1
002430  525232 2220 03                    LDX    X,BUGBUG,DU
002431  525232 2230 03                    LDX    Y,BUGBUG,DU
002432  525232 2240 03                    LDX    Z,BUGBUG,DU
002433  525232 2250 03                    LDX    Q,BUGBUG,DU
002434  000004 7270 11      4478          LXL    L,T$TRA,T       RETRIEVE RETURN                      750
                002435      4479          BUGL   (T$TRA,T)       BUG IT                               760
                525233           BUGBUG   SET    BUGBUG+1
002435  525233 2200 03                    LDX    0,BUGBUG,DU
002436  000004 4400 11                    SXL    0,T$TRA,T
002437  000027 2350 11      4480          LDA    T$TEMP1,T       GET RETURN WORD FOR CALLER           770
                002440      4481          BUG    (T$TEMP1,T)     BUG IT                               780
                525234           BUGBUG   SET    BUGBUG+1
002440  525234 2200 03                    LDX    0,BUGBUG,DU
 END OF BINARY CARD IOS00067
002441  000027 7400 11                    STX    0,T$TEMP1,T
002442  000027 4400 11                    SXL    0,T$TEMP1,T
                002443      4482          BUG    (T$TEMP2,T)                                       790
                525235           BUGBUG   SET    BUGBUG+1
002443  525235 2200 03                    LDX    0,BUGBUG,DU
002444  000026 7400 11                    STX    0,T$TEMP2,T
002445  000026 4400 11                    SXL    0,T$TEMP2,T
002446  000000 7100 17      4483          TRA    0,L             RETURN TO CALLER                     800
                            4484 *$*      DISK   RES3                                              810
```

```
                 002447    4486        USE    CODE
                           4487        HEAD   R
                           4488 *
                           4489 *
                           4490 *                              RELP
                           4491 *
                           4492 *      RELP RELEASES A PERIPHERAL UNIT OF A SPECIFIED TYPE.
                           4493 *
                           4494 *      CALL WITH
                           4495 *             TSX L.R$RELP
                           4496 *      CALL WITH
                           4497 *             C(J) = JOB NUMBER
                           4498 *             C(T) = TBLOCK ADDRESS
                           4499 *             C(L) = RETURN ADDRESS
                           4500 *             C(AU) = PTR TO DEVICE HEADER
                           4501 *             C(AL) = DEVICE NUMBER ADDRESS
                           4502 *      CALLS
                           4503 *             $CLOSE
                           4504 *             $NOTIF
                           4505 *             $CAUSE
                           4506 *             O$LOGS
                           4507 *             O$LOGC
                           4508 *             O$LOGX
                           4509 *      RETURNS WITH
                           4510 *             C(J) = JOB NUMBER
                           4511 *             C(T) = TBLOCK ADDRESS
                           4512 *      EXITS TO 0,L
                           4513 *
                           4514 *      EXITS WITH PERIPHERAL DEALLOCATED.
                           4515 *
                 002447    4516 RELP   BSS    0                ENTRY POINT
002447  000004 4470 11     4517        SXL    L,T$TRA,T        SAVE RETURN ADDRESS
                           4518 *
                           4519 *      PERFORM CONSISTANCY CHECKS
                           4520 *
002450  000027 7550 11     4521        STA    T$TEMP1,T        SAVE PASSED INFORMATION
002451  000027 7220 11     4522        LXL    X,T$TEMP1,T      GET DEVICE ADDRESS
002452  000002 2340 12     4523        SZN    FLAG,X           SHOULD BE BUSY
002453  777777 6000 00     4524        TZE    $ERROR           IT ISN'T!
002454  777777 2360 03     4525        LDQ    -1,DU            SET FOR LOWER HALF COMPARE
002455  000002 2350 12     4526        LDA    ALLC,X           GET ALLOCATED JOB NUMBER
002456  000006 2110 11     4527        CMK    T$JCB,T          CHECK FOR CORRECT J NUMBER
002457  777777 6010 00     4528        TNZ    $ERROR           SHOULD BE THE SAME
                 002460    4529        BUGL   (ALLC,X)         OK, DESTROY IT
                 525236         BUGBUG SET    BUGBUG+1
002460  525236 2200 03                 LDX    0,BUGBUG,DU
002461  000002 4400 12                 SXL    0,ALLC,X
002462  000027 2240 11     4530        LDX    Z,T$TEMP1,T      GET PTR TO DEVICE HEADER
002463  000005 0540 14     4531        AOS    AVAIL,Z          BUMP THE NUMBER NOW FREE
                           4532 *
```

```
                                4533 *       CHECK IF CLOSE REQUESTED                              580
                                4534 *                                                             590
   002464  000002 2350 12       4535         LDA     FLAG,X            GET PERIPHERAL FLAG         600
   002465  100000 3150 03       4536         CANA    RSVE,DU           IS IT RESERVED?             610
 END OF BINARY CARD IOS00068
   002466  002545 6000 00       4537         TZE     RELP3             NO, SO CONTINUE             620
                                4538 *                                                             630
                                4539 *       CLOSE REQUESTED -- CLOSE AND LOG                      640
                                4540 *                                                             650
   002467  000006 0540 14       4541         AOS     OPER,Z            BUMP OPER COUNT ONE         660
                      002470    4542         DECRM   (AVAIL,Z)         DECREMENT ONCE FOR THE 'NOW FREE'   670
   002470  000001 3360 07                    LCQ     1,DL
   002471  000005 0560 14                    ASQ     AVAIL,Z
                      002472    4543         DECRM   (AVAIL,Z)         AND AGAIN FOR THE 'CLOSE'   680
   002472  000001 3360 07                    LCQ     1,DL
   002473  000005 0560 14                    ASQ     AVAIL,Z
                      002474    4544         DECRM   (OMAX,Z)          AND OUR MAX TOO             690
   002474  000001 3360 07                    LCQ     1,DL
   002475  000004 0560 14                    ASQ     OMAX,Z
   002476  777777 6040 00       4545         TMI     $ERROR            ***PROBLEM                  700
   002477  300000 6750 03       4546         ERA     CLOSE+RSVE,DU     MARK IT CLOSED AND NOT RESERVED   710
   002500  000002 7550 12       4547         STA     FLAG,X            RESTORE FLAG WORD           720
   002501  000001 2230 12       4548         LDX     Y,FRN,X           GET FRN OF DEVICE           730
   002502  000025 7430 11       4549         STX     Y,T$TEMP3,T       SAVE FOR CLOSE              740
                      002503    4550         BUGU    (FRN,X)           BUG FRN                     750
                      525237         BUGBUG  SET     BUGBUG+1
   002503  525237 2200 03                    LDX     0,BUGBUG,DU
   002504  000001 7400 12                    STX     0,FRN,X
                                4551 *                                                             760
                                4552 *       CLOSE PERIPHERAL                                      770
                                4553 *                                                             780
                      002505    4554 RELP1   CLOSE   (T$TEMP3,T)       CLOSE THE DEVICE            790
   002505  000704 7000 00                    TSX     0,$CLOSE
   002506  000025 0000 11                    ARG     T$TEMP3,T
                      002507    4555         CHECK   RELP2,B$BZ,RELP1                              800
   002507  000000 7200 11                    LXL     0,T$SRW1,T
   002510  000077 3600 03                    ANX     0,B$STMK,DU
   002511  002515 6000 00                    TZE     RELP2
   002512  000003 1000 03                    CMPX    0,B$BZ,DU
   002513  002505 6000 00                    TZE     RELP1
 END OF BINARY CARD IOS00069
   002514  777777 7100 00                    TRA     $ERROR
                      002515    4556 RELP2   BSS     0                                             810
                                4557 *       SEE WHO HAS IT?                                       820
                                4558 *                                                             830
                                4559 *       SEND RELEASE MESSAGE TO SUB-MODULE                    840
                                4560 *                                                             850
   002515  000027 2220 11       4561         LDX     X,T$TEMP1,T       RESTORE TYPE PTR            860
   002516  000001 2350 12       4562         LDA     STATE,X           GET STATE FOR CAUSE         870
   002517  000025 7550 11       4563         STA     T$TEMP3,T                                     880
```

```
    002520  000027 7240 11      4564          LXL     Z,T$TEMP1,T       RESTORE DEVICE UNIT PTR        890
    002521  000000 2350 14      4565          LDA     ABBR,Z            GET NAME                       900
    002522  000007 3750 07      4566          ANA     7,DL              MASK TO UNIT NUMBER            910
    002523  000022 7350 00      4567          ALS     18                MOVE TO AU                     920
    002524  006000 2750 03      4568          ORA     B$.REL.DU         COMMAND: RELEASE               930
    002525  000024 7550 11      4569          STA     T$TEMP4,T         SAVE FOR CAUSE                 940
                     002526     4570          BRANCH  NOPASS,C$MESSX,(T$TEMP3,T),(T$TEMP4,T)           950
    002526  001467 7000 00                    TSX     0,T$GETT
    002527  000000 6220 11                    EAX     X,0,T
    002530  000005 2210 12                    LDX     T,T$LINK,X
    002531  000025 2360 11                    LDQ     T$TEMP3,T
    002532  000027 7560 12                    STQ     T$TEMP1,X
    002533  000024 2360 11                    LDQ     T$TEMP4,T
    002534  000026 7560 12                    STQ     T$TEMP2,X
    002535  000000 6210 12                    EAX     T,0,X
    002536  003140 6200 00                    EAX     0,C$MESSX
    002537  000004 7400 11                    STX     0,T$TRA,T
    002540  000004 6200 11                    EAX     0,Q$OFFST,T
    002541  005162 7170 00                    XED     Q$XADD+Q$TASK
 END OF BINARY CARD IOS00070
    002542  000005 2210 12                    LDX     T,T$LINK,X
                     002543                   BUGXR   (0,X)
                     525240          BUGBUG SET     BUGBUG+1
    002543  525240 2200 03                    LDX     0,BUGBUG,DU
    002544  525240 2220 03                    LDX     X,BUGBUG,DU
                     002545     4571 RELP3    BSS     0                                                960
                                4572 *                                                                 970
                                4573 *        FINISH UP                                                980
                                4574 *                                                                 990
    002545  400000 2350 03      4575          LDA     BUSY,DU           UNSET THE BUSY BIT             1000
    002546  000027 7220 11      4576          LXL     X,T$TEMP1,T       RESTORE PTR TO DEVICE UNIT     1010
    002547  000002 6550 12      4577          ERSA    FLAG,X                                           1020
    002550  000004 7270 11      4578          LXL     L,T$TRA,T         RETRIEVE RETURN                1030
                     002551     4579          BUGL    (T$TRA,T)         BUG IT                         1040
                     525241          BUGBUG SET     BUGBUG+1
    002551  525241 2200 03                    LDX     0,BUGBUG,DU
    002552  000004 4400 11                    SXL     0,T$TRA,T
                     002553     4580          BUGXR   (X,Y,Z,Q)         BUG REGISTERS                  1050
                     525242          BUGBUG SET     BUGBUG+1
    002553  525242 2220 03                    LDX     X,BUGBUG,DU
    002554  525242 2230 03                    LDX     Y,BUGBUG,DU
    002555  525242 2240 03                    LDX     Z,BUGBUG,DU
    002556  525242 2250 03                    LDX     Q,BUGBUG,DU
                     002557     4581          BUGA                                                     1060
                     525243          BUGBUG SET     BUGBUG+1
    002557  525243 2350 03                    LDA     BUGBUG,DU
    002560  525243 2750 07                    ORA     BUGBUG,DL
                     002561     4582          BUGQ                                                     1070
                     525244          BUGBUG SET     BUGBUG+1
    002561  525244 2360 03                    LDQ     BUGBUG,DU
```

```
002562  525244 2760 07                      ORQ    BUGBUG,DL
002563  000000 7100 17        4583          TRA    0,L                  RETURN TO CALLER          1080
                      002564  4584          EXIT                        WAIT FOR REPLY            1090
002564  003074 7100 00                      TRA    $EXIT
                              4585 *$*      DISK   PERIPH                                         1100
```

```
                      002565          USE    CODE
                                      HEAD   R
                                *
                                *
                                *                              PERIPHERAL TYPE TABLE
                                *
                                *     THE PERIPHERAL TYPE TABLE HAS AN ENTRY FOR EACH
                                *     TYPE OF RESOURCE.  THUS THE ENTRY POINTER POINTS TO THE
                                *     RESOURCE DEVICE HEADER.  THE HEADER CONTIANS SUCH INFORMATION
                                *     AS THE TOTAL NUMBER OF DEVICES OF THIS TYPE, THE NUMBER CUR-
                                *     RENTLY AVAILABLE, A POINTER TO THE FIRST DEVICE, ETC.  THE
                                *     PERIPHERAL DEVICE ITSELF CONTAINS SUFFICIENT INFORMATION FOR
                                *     ANY OPERATION ON THE PERIPHERAL.
                                *
                                *
                                *     TABLE OF PERIPHERAL TYPES
                                *
                      004743          USE    CONST
                      004743    TABLE BSS    0
   004743  000000 0000 00             ARG    0                0 = INVALID
   004744  005550 0000 00       TABCP ARG    CPTAB            1 = CARD PUNCH
END OF BINARY CARD IOS00071
   004745  005560 0000 00       TABCR ARG    CRTAB            2 = CARD READER
   004746  005570 0000 00       TABLP ARG    LPTAB            3 = LINE PRINTER
   004747  005600 0000 00       TABMT ARG    MTTAB            4 = MAGNETIC TAPE
   004750  005610 0000 00       TABOP ARG    OPTAB            5 = OPERATOR'S CONSOLE
   004751  000000 0000 00             ARG    0                6 = INVALID
   004752  000000 0000 00             ARG    0                7 = INVALID
                      002565          USE    PREVIOUS
                                *
                                *
                      000001    TYPCP EQU    TABCP-TABLE      TYPE:  CARD PUNCH
                      000002    TYPCR EQU    TABCR-TABLE      TYPE:  CARD READER
                      000003    TYPLP EQU    TABLP-TABLE      TYPE:  LINE PRINTER
                      000004    TYPMT EQU    TABMT-TABLE      TYPE:  MAG TAPE
                      000005    TYPOP EQU    TABOP-TABLE      TYPE:  OPERATOR'S CONSOLE
                                *
                                *
                                *     COMMUNICATIONS STATES
                                *
                                      HEAD
                      001000    CPST  EQU    R$TYPCP*512      CP STATE FOR COMMUNICATIONS
                      002000    CRST  EQU    R$TYPCR*512      CR STATE
                      003000    LPST  EQU    R$TYPLP*512      LP STATE
                      004000    MTST  EQU    R$TYPMT*512      MT STATE
                      005000    OPST  EQU    R$TYPOP*512      OP STATE
                                      HEAD   R
```

```
002565          USE     CODE
                HEAD    R
     *
     *
     *                                  PERIPHERAL HEADER TABLE
     *
     *          A DEVICE HEADER IS THE ITEM THAT AN ENTRY IN THE TABLE OF
     *          PERIPHERAL TYPES POINTS TO.  THE HEADER CONTAINS THE POINTER
     *          TO THE DEVICES OF A CERTAIN TYPE (I.E. LINE PRINTERS).
     *          IT ALSO CONTAINS SUCH INFORMATION AS THE CONFIGURATION OF THE
     *          SYSTEM (MAXIMUM NUMBER OF DEVICES OF A CERTAIN TYPE).
     *          LASTLY, IT CONTAINS A POINTER TO THE CORRESPONDING QUEUE.
     *
     *
     *          FORMAT OF DEVICE HEADER
     *
000000     PTR    EQU     0                  POINTER TO DEVICE TABLE
000001     STATE  EQU     PTR+1              STATE CORRES. TO RECIPIENT
000002     ELT    EQU     STATE+1            (UPPER) ELEMENT SIZE FOR DEVICE
000002     ACC    EQU     ELT                (LOWER) ACCESSES
000003     MAX    EQU     ELT+1              MAXIMUM IN SYSTEM
000004     OMAX   EQU     MAX+1              OUR MAXIMUM
000005     AVAIL  EQU     OMAX+1             NUMBER CURRENTLY AVAILABLE (FREE)
000006     OPER   EQU     AVAIL+1            NUMBER OPERATOR HOLDS
000007     SPARE  EQU     OPER+1             SPARE
000010     DHLEN  EQU     SPARE+1-PTR        DEVICE HEADER LENGTH
     *
     *
     *          DEVICE HEADER GENERATING MACRO
     *
           DEVHDR MACRO   NAME,ELT,ACC,MAX,OMAX,AVAIL,OPER
           #1TAB  ARG     #1
                  ZERO    $#1ST,0            STATE
                  ZERO    #2,#3              ELEMENT SIZE/ ACCESSES
                  VFD     36/#4              MAX
                  VFD     36/#5              OUR MAX
                  VFD     36/#6              AVAILABLE (FREE)
                  VFD     36/#7              NUMBER OPERATOR HOLDS
                  DEC     0                  SPARE
                  ENDM    DEVHDR
```

```
         R                                RESOURCE ALLOCATION -- PERIPHERAL HEADER TABLE

                              4675 *                                                                               990
                              4676 *                                                                              1000
                              4677 *         HERE IS THE LIST OF DEVICE HEADERS IN ALPHABETICAL ORDER.            1010
                              4678 *                                                                              1020
                      005543  4679           USE     STORE                                                        1030
                      005550  4681           EIGHT                                                                1050
                      005550  4682 DEVHR     BSS     0                   START OF DEVICE HEADER LIST              1060
                      005550  4683           DEVHDR  CP,6,B$AP,CPMAX,0,0,CPMAX                                    1070
    005550  005620 0000 00         CPTAB     ARG     CP
    005551  001000 000000                    ZERO    $CPST,0
    005552  000006 000004                    ZERO    6,B$AP
    005553  000000000001                     VFD     36/CPMAX
    005554  000000000000                     VFD     36/0
    005555  000000000000                     VFD     36/0
    005556  000000000001                     VFD     36/CPMAX
    005557  000000000000                     DEC     0
                      005560  4684           DEVHDR  CR,6,B$RD,CRMAX,0,0,CRMAX                                    1080
    005560  005623 0000 00         CRTAB     ARG     CR
    005561  002000 000000                    ZERO    $CRST,0
    005562  000006 000020                    ZERO    6,B$RD
    005563  000000000001                     VFD     36/CRMAX
    005564  000000000000                     VFD     36/0
  END OF BINARY CARD IOS00072
    005565  000000000000                     VFD     36/0
    005566  000000000001                     VFD     36/CRMAX
    005567  000000000000                     DEC     0
                      005570  4685           DEVHDR  LP,6,B$AP,LPMAX,0,0,LPMAX                                    1090
    005570  005626 0000 00         LPTAB     ARG     LP
    005571  003000 000000                    ZERO    $LPST,0
    005572  000006 000004                    ZERO    6,B$AP
    005573  000000000002                     VFD     36/LPMAX
    005574  000000000000                     VFD     36/0
    005575  000000000000                     VFD     36/0
    005576  000000000002                     VFD     36/LPMAX
    005577  000000000000                     DEC     0
                      005600  4686           DEVHDR  MT,36,B$RD+B$AP,MTMAX,0,0,MTMAX                              1100
    005600  005634 0000 00         MTTAB     ARG     MT
    005601  004000 000000                    ZERO    $MTST,0
    005602  000044 000024                    ZERO    36,B$RD+B$AP
    005603  000000000010                     VFD     36/MTMAX
    005604  000000000000                     VFD     36/0
    005605  000000000000                     VFD     36/0
    005606  000000000010                     VFD     36/MTMAX
    005607  000000000000                     DEC     0
                      005610  4687           DEVHDR  OP,9,B$ALL,OPMAX,0,0,OPMAX                                   1110
    005610  005664 0000 00         OPTAB     ARG     OP
    005611  005000 000000                    ZERO    $OPST,0
    005612  000011 000037                    ZERO    9,B$ALL
  END OF BINARY CARD IOS00073
    005613  000000000001                     VFD     36/OPMAX
```

```
005614 000000000000                   VFD   36/0
005615 000000000000                   VFD   36/0
005616 000000000001                   VFD   36/OPMAX
005617 000000000000                   DEC   0
              005620    4688 XDVHR   EQU   *                END OF DEVICE HEADER TABLE        1120
              000050    4689 DHRLN   EQU   *-DEVHR          LENGTH OF TABLE                   1130
              002565    4690         USE   PREVIOUS                                           1140
```

```
002565          USE     CODE
                HEAD    R
     *
     *
     *                                   PERIPHERAL DEVICE TABLE
     *
     *          THERE IS A ONE-TO-ONE CORRESPONDENCE BETWEEN A 'DEVICE' AND
     *          PHYSICAL DEVICE IN THE MACHINE ROOM.  THE DEVICE CONTAINS THE
     *          NAME OF THE DEVICE. THE FRN WHEN OPEN, FLAG BITS TELLING ITS
     *          CURRENT STATUS. AND IF BUSY. WHO IS RESPONSIBLE.
     *
     *
     *          FORMAT OF PERIPHERAL DEVICE
     *
     *
000000     ABBR   EQU     0               FOUR CHARACTER ASCII ABBREVIATION FOR PERIPHERAL
000001     FRN    EQU     ABBR+1          (UPPER) FRN OF PERIPHERAL WHEN OPEN
000002     FLAG   EQU     FRN+1           (UPPER) FLAG BITS FOR THE PERIPHERAL
000002     ALLC   EQU     FLAG            (LOWER) JOB NUMBER USING IT WHEN BUSY
000003     DEVLN  EQU     ALLC+1-ABBR     DEVICE ENTRY LENGTH
     *
     *
     *          BITS FOR FLAG
     *
400000     BUSY   EQU     B$SIGN          ON IF NOT ALLOCATABLE
200000     CLOSE  EQU     BUSY/2          ON IF CLOSED
100000     RSVE   EQU     CLOSE/2         ON IF OPERATOR REQUESTED A CLOSE
     *
     *
     *          DEVICE GENERATING MACRO
     *
           DEVICE MACRO   NAME
                  UASCI   1,#1
                  ARG     0
                  ZERO    BUSY+CLOSE,0
                  ENDM    DEVICE
     *
     *
           DEVT   MACRO   NAME,(LIST OF DEVICE NUMBERS)
           #1     BSS     0
                  IDRP    #2
           SET    SET     #2
                  DEVICE  #1#2
                  IDRP
           #1MAX  SET     SET+1
                  ENDM    DEVT
```

```
                               4739 *                                                                   1630
                               4740 *                                                                   1640
                               4741 *       HERE IS THE TABLE OF DEVICES IN ALPHABETICAL ORDER.         1650
                               4742 *                                                                   1660
                               4743 *                                                                   1670
                     005620    4744         USE     STORE                                               1680
                     005620    4745 PERT    BSS     0              START OF PERIPHERAL TABLE            1690
                     005620    4746 DEVTB   BSS     0              START OF DEVICE TABLE                1700
                     005620    4747         DEVT    CP.(00)        CARD PUNCHES                         1710
                     005620         CP      BSS     0
                     000000         SET     SET     00
                     005620                 DEVICE  CP00
  005620  103120060060                      UASCI   1.CP00
  005621  000000 0000 00                    ARG     0
  005622  600000 000000                     ZERO    BUSY+CLOSE.0
                     000001         CPMAX   SET     SET+1
                     005623    4748         DEVT    CR.(00)        CARD READERS                         1720
                     005623         CR      BSS     0
                     000000         SET     SET     00
                     005623                 DEVICE  CR00
  005623  103122060060                      UASCI   1.CR00
  005624  000000 0000 00                    ARG     0
  005625  600000 000000                     ZERO    BUSY+CLOSE.0
                     000001         CRMAX   SET     SET+1
                     005626    4749         DEVT    LP.(00.01)     LINE PRINTERS                        1730
                     005626         LP      BSS     0
                     000000         SET     SET     00
                     005626                 DEVICE  LP00
  005626  114120060060                      UASCI   1.LP00
  005627  000000 0000 00                    ARG     0
  005630  600000 000000                     ZERO    BUSY+CLOSE.0
                     000001         SET     SET     01
                     005631                 DEVICE  LP01
  005631  114120060061                      UASCI   1.LP01
  005632  000000 0000 00                    ARG     0
  005633  600000 000000                     ZERO    BUSY+CLOSE.0
                     000002         LPMAX   SET     SET+1
                     005634    4750         DEVT    MT.(00.01.02.03.04.05.06.07)   MAG TAPES            1740
                     005634         MT      BSS     0
                     000000         SET     SET     00
                     005634                 DEVICE  MT00
  005634  115124060060                      UASCI   1.MT00
  005635  000000 0000 00                    ARG     0
  005636  600000 000000                     ZERO    BUSY+CLOSE.0
                     000001         SET     SET     01
                     005637                 DEVICE  MT01
  005637  115124060061                      UASCI   1.MT01
 END OF BINARY CARD IOS00074
  005640  000000 0000 00                    ARG     0
  005641  600000 000000                     ZERO    BUSY+CLOSE.0
```

```
                    000002        SET   SET     02
                    005642              DEVICE  MT02
005642 115124060062                     UASCI   1,MT02
005643 000000 0000 00                   ARG     0
005644 600000 000000                    ZERO    BUSY+CLOSE,0
                    000003        SET   SET     03
                    005645              DEVICE  MT03
005645 115124060063                     UASCI   1,MT03
005646 000000 0000 00                   ARG     0
005647 600000 000000                    ZERO    BUSY+CLOSE,0
                    000004        SET   SET     04
                    005650              DEVICE  MT04
005650 115124060064                     UASCI   1,MT04
005651 000000 0000 00                   ARG     0
005652 600000 000000                    ZERO    BUSY+CLOSE,0
                    000005        SET   SET     05
                    005653              DEVICE  MT05
005653 115124060065                     UASCI   1,MT05
005654 000000 0000 00                   ARG     0
005655 600000 000000                    ZERO    BUSY+CLOSE,0
                    000006        SET   SET     06
                    005656              DEVICE  MT06
005656 115124060066                     UASCI   1,MT06
005657 000000 0000 00                   ARG     0
005660 600000 000000                    ZERO    BUSY+CLOSE,0
                    000007        SET   SET     07
                    005661              DEVICE  MT07
005661 115124060067                     UASCI   1,MT07
005662 000000 0000 00                   ARG     0
005663 600000 000000                    ZERO    BUSY+CLOSE,0
                    000010        MTMAX SET     SET+1
                    005664   4751         DEVT    OP,(00)         OPERATOR'S CONSOLE      1750
                    005664        OP    BSS     0
                    000000        SET   SET     00
                    005664              DEVICE  OP00
005664 117120060060                     UASCI   1,OP00
005665 000000 0000 00                   ARG     0
END OF BINARY CARD IOS00075
005666 600000 000000                    ZERO    BUSY+CLOSE,0
                    000001        OPMAX SET     SET+1
                    000047   4752 TT    SET     *-PERT          PERT LENGTH             1760
                    000015   4753 PERTL EQU     TT/DEVLN        NUMBER OF ENTRIES       1770
                             4754 *                                                     1780
                    005665   4755 OPFRN EQU     OP+FRN          OP FRN                  1790
                    002565   4756       USE     PREVIOUS                                1800
                             4757 *$*   DISK    SCHED                                   1810
```

```
                    002565          USE     CODE
                                    HEAD    R
                              *
                              *
                              *                             MAIN
                              *
                              *     THIS IS THE SCHEDULER FOR THE ENTIRE MONITOR JOB-RUN SYSTEMS.
                              *     IT CHECKS THE IMPUT QUEUE LISTS FOR POSSIBLE JOB INITIATION.
                              *     TO DO THIS, IT POINTS TO A QUEUE LIST AND CALLS THE  RUN
                              *     PART OF THE SCHEDULER.  RUN  WILL TRY TO START EVERY WAITING
                              *     JOB ON THE SPECIFIED QUEUE LIST.  SO   MAIN  MERELY POINTS TO
                              *     LIST, AND THE SUBROUTINES DO THE REST.
                              *
                              *     CURRENTLY THERE ARE ONLY TWO INPUT QUEUE LIST:
                              *             Q$INP1 -- GENERAL ALL JOBS GO HERE EXCEPT...
                              *             Q$INP2 -- JOBS WHICH ARE SHORT PRINT JOBS.
                              *             Q$INP3 -- JOBS WHICH ARE MEDIUM PRINTER JOBS (NOT IMPLEMENTED)
                              *
                              *     THIS IS AN ASYNCHRONOUS TASK
                              *
                    002565    SKED  BSS     0
002565  005240 6250 00              EAX     Q,Q$INP2        POINT TO SHORT PRINT JOB QUEUE LIST
002566  002575 7070 00              TSX     L,SKEDR         RUN 'EM
002567  005220 6250 00              EAX     Q,Q$INP1        POINT TO ALL OTHER JOB QUEUE LIST
002570  002575 7070 00              TSX     L,SKEDR         LET 'EM GO, TOO
002571  005260 6250 00              EAX     Q,Q$INP3        POINT TO MEDIUM PRINT JOB QUEUE LIST
002572  002575 7070 00              TSX     L,SKEDR         LET HER RIP.
                    002573          RELT                    DONE, RELEASE TCB
002573  001477 7000 00              TSX     0,T$RELT
                    002574          EXIT                    POOF.
002574  003074 7100 00              TRA     $EXIT
                              *$*   DISK    SCHEDULE
```

```
                     002575    4790        USE     CODE
                               4791        HEAD    R
                               4792 *
                               4793 *
                               4794 *                                RUN
                               4795 *
                               4796 *      THIS SUBROUTINE CHECKS A SPECIFIED QUEUE-LIST TO SEE IF
                               4797 *      ANY OF THE WAITING JOBS CAN HAVE ALL OF THEIR RESOURCE REQUESTS
                               4798 *      SATISFIED.  IF SO, THEN THAT JOB IS REMOVED FROM THE LIST AND
                               4799 *      PLACED ON THE Q$TASK QUEUE IN ORDER TO RUN.
                               4800 *
                               4801 *      ALL JOBS ARE ONE-STEP PROCESSES.  THIS MEANS THAT THE JOBS
                               4802 *      CONSIST OF A SINGLE ACTIVITY.  THUS IN ORDER TO AVOID DEAD-
                               4803 *      LOCK SITUATIONS, ALL RESOURCES ARE ALLOCATED COLLECTIVELY WHEN
                               4804 *      A JOB IS TO START; OTHERWISE THE JOB MUST WAIT TILL ALL HIS
                               4805 *      REQUESTS FOR SYSTEM RESOURCES CAN BE SATISFIED SIMULTANEOUSLY.
                               4806 *
                               4807 *      CALL WITH
                               4808 *              C(XT) = TCB
                               4809 *              C(XJ) = JCB (PHONY)
                               4810 *      CALL BY
                               4811 *              TSX L,R$SKEDR
                               4812 *      RETURNS 0,L
                               4813 *      RETURNS WITH
                               4814 *              C(XT) = TCB
                               4815 *      USES
                               4816 *              TEMP3  (UPPER) QADDRESS
                               4817 *              TEMP3  (LOWER) RETURN ADDRESS
                               4818 *              TEMP4  PTR TO NEXT RESOURCE NEED
                               4819 *              TEMP5  CURRENT NEED
                               4820 *              TEMP6  COUNTER OF CURRENT NEED
                               4821 *              TEMP8  (UPPER) PTR TO PREDECESSOR  (Q$LINK)
                               4822 *              TEMP8  (LOWER) PTR TO CURRENT  (Q$OFFST)
                               4823 *
                     002575    4824 SKEDR  BSS     0                ENTRY POINT
002575  000025 4470 11         4825        SXL     L,T$TEMP3,T      SAVE RETURN ADDRESS
002576  000025 7450 11         4826        STX     Q,T$TEMP3,T      SAVE PTR TO QUEUE-LIST
                               4827 *
                               4828 *      IS THIS LIST EMPTY?
                               4829 *
002577  000001 6200 15         4830        EAX     0,Q$FIRST+1,Q    ADDRESS OF FIRST ELEMENT
002600  000001 1000 15         4831        CMPX    0,Q$LAST,Q       DOES LAST POINT TO IT?
002601  002724 6000 00         4832        TZE     SKDRX            IF SO, EXIT
002602  000000 6200 15         4833        EAX     0,Q$FIRST,Q      GET PTR TO WHAT IS TO BE PREDECESSOR
                     002603    4834 SKDR0  BSS     0
002603  000020 7400 11         4835        STX     0,T$TEMP8,T      SAVE IT FOR RE-LINKING
002604  000000 2200 10         4836        LDX     0,0,0            OFFSET PTR TO BLOCK
002605  777777 6000 00         4837        TZE     $ERROR           ***PROBLEM
002606  000020 4400 11         4838 SKR0.1 SXL     0,T$TEMP8,T      SAVE PTR TO CURRENT BLOCK
002607  777774 6200 10         4839        EAX     0,-Q$OFFST,0     RELATE TO BEGINNING
```

```
                               *
                               *      CHECK IF RESOURCE REQUIREMENTS CAN BE SATISFIED
                               *
END OF BINARY CARD IOS00076
     002610  000006 7260 10           LXL    J,T$JCB,0       GET PTR TO ASSOCIATED JOB DESCRIPTOR
     002611  000013 6230 16           EAX    Y,J$RES,J       GET PTR TO RESOURCE REQUEST BLOCK
     002612  000024 7430 11      SKDR1 STX   Y,T$TEMP4,T     SAVE IT
     002613  000000 2240 13           LDX    Z,0,Y           GET TYPE OF NEXT RESOURCE REQUESTED
     002614  002645 6040 00           TMI    SKDR4           END OF LIST--RUN THIS JOB
     002615  004743 2220 14           LDX    X,TABLE,Z       GET PTR TO DEVICE HEADER
     002616  777777 6000 00           TZE    $ERROR          ***PROBLEM
     002617  000000 2350 13           LDA    0,Y             GET NUMBER REQUIRED
     002620  777777 3750 07           ANA    -1,DL           MASK TO NUMBER ONLY
     002621  000003 1150 12           CMPA   MAX,X           TEST AGAINST MAX
     002622  002624 6000 00           TZE    *+2             OK
     002623  777777 6030 00           TRC    $ERROR          ***THAT'S SOMEBODY ELSE
     002624  002625 7100 14           TRA    *+1,Z           BRANCH THRU TABLE ON TYPE
     002625  777777 7100 00           TRA    $ERROR          0 = ILLEGAL
     002626  002735 7100 00           TRA    SCPCK           1 = CARD PUNCH
     002627  002735 7100 00           TRA    SCRCK           2 = CARD READER
     002630  002741 7100 00           TRA    SLPCK           3 = LINE PRINTER
     002631  002735 7100 00           TRA    SMTCK           4 = MAG TAPE
     002632  777777 7100 00           TRA    $ERROR          5 = OP CONSOLE (ALLOW NO ONE TO OWN IT)
     002633  777777 7100 00           TRA    $ERROR          6 = INVALID
     002634  777777 7100 00           TRA    $ERROR          7 = INVALID
                     002635      SKDR2 BSS   0               RETURN HERE IF RESOURCES AVAILABLE
     002635  000000 6230 05           EAX    Y,0,AL          MOVE COUNT TO Y
END OF BINARY CARD IOS00077
     002636  000024 0230 11           ADLX   Y,T$TEMP4,T     ADD IN OLD POINTER
     002637  002612 7100 00           TRA    SKDR1           AND LOOP
                     002640      SKDR3 BSS   0               RETURN HERE IF THIS JOB CAN'T START
     002640  000004 6230 10           EAX    Y,Q$OFFST,0     GET OFFSET PTR TO THIS BLOCK
     002641  000001 1030 15           CMPX   Y,Q$LAST,Q      IS IT THE LAST?
     002642  002724 6000 00           TZE    SKDRX           IF SO, EXIT
     002643  000003 6200 10           EAX    0,Q$LINK,0      OTHERWISE GET PTR TO NEXT
                               *
     002644  002603 7100 00           TRA    SKDR0           GET NEXT BLOCK
                               *
                               *      ALLOCATE RESOURCES
                               *
                     002645      SKDR4 BSS   0               START THIS JOB
     002645  003064 7070 00           TSX    L,J$JNUMB       GET A JOB NUMBER
     002646  000007 4420 16           SXL    X,J$JCB,J       SAVE AS STATE FOR COMMUNICATIONS
     002647  000000 6350 12           EAA    0,X             MOVE JOB NUMBER
     002650  000022 7710 00           ARL    18              INTO AL RIGHT JUSTIFIED
     002651  000010 2360 16           LDQ    J$MESS,J        GET MESSAGE
     002652  000004 7360 00           QLS    4               LEFT JUSTIFIED
     002653  000004 7730 00           LRL    4               MOVE INTO QU
     002654  000010 7560 16           STQ    J$MESS,J        COMPLETE MESSAGE, SAVE
     002655  000013 6230 16           EAX    Y,J$RES,J       GET POINTER TO RESOURCE NEED BLOCK
```

```
   002656  000024 7430 11         4888          STX     Y,T$TEMP4,T       SAVE
                                  4889 *
                         002657   4890 SKDR5    BSS     0
   002657  000000 2350 13         4891          LDA     0,Y               GET NEXT RESOURC REQUIREMENT
   002660  002707 6040 00         4892          TMI     SKDR8             TEST FOR DONE
   002661  000023 7550 11         4893          STA     T$TEMP5,T         SAVE IT
   002662  777777 3750 07         4894          ANA     -1,DL             MASK TO COUNT ONLY
   002663  777777 6000 00         4895          TZE     $ERROR            ***BLEWIT
END OF BINARY CARD IOS00078
   002664  000000 5310 00         4896          NEG                       COMPLEMENT
   002665  000022 7550 11         4897          STA     T$TEMP6,T         SAVE AS LOOP COUNTER
   002666  000023 2350 11         4898 SKDR6    LDA     T$TEMP5,T         GET BACK TYPE
   002667  002670 7100 01         4899          TRA     *+1,AU            BRANCH ON TYPE
   002670  777777 7100 00         4900          TRA     $ERROR            0 = ILLEGAL
   002671  002751 7100 00         4901          TRA     SCPAL             1 = CARD PUNCH
   002672  002751 7100 00         4902          TRA     SCRAL             2 = CARD READER
   002673  002753 7100 00         4903          TRA     SLPAL             3 = LINE PRINTER
   002674  002751 7100 00         4904          TRA     SMTAL             4 = MAG TAPE
   002675  777777 7100 00         4905          TRA     $ERROR            5 = OP CONSOLE (ALLOW NO ONE TO OWN IT)
   002676  777777 7100 00         4906          TRA     $ERROR            6 = INVALID
   002677  777777 7100 00         4907          TRA     $ERROR            7 = INVALID
                         002700   4908 SKDR7    BSS     0
   002700  000024 2230 11         4909          LDX     Y,T$TEMP4,T       RESTORE RESOURCE PTR
   002701  000000 7550 13         4910          STA     0,Y               SAVE ALLOCATED RESOURCE IN JCB
   002702  000001 0230 03         4911          ADLX    Y,1,DU            BUMP ONE
   002703  000024 7430 11         4912          STX     Y,T$TEMP4,T       SAVE
   002704  000022 0540 11         4913          AOS     T$TEMP6,T         BUMP LOOP COUNTER
   002705  002666 6040 00         4914          TMI     SKDR6             LOOP IF MORE DEVICES OF THIS TYPE NEEDED
   002706  002657 7100 00         4915          TRA     SKDR5             OTHERWISE GET NEXT TYPE
                                  4916 *
                                  4917 *        RESOURCES ALLOCATED, REMOVE FROM LIST
                                  4918 *
                         002707   4919 SKDR8    BSS     0
   002707  000020 7200 11         4920          LXL     0,T$TEMP8,T       GET BACK PTR TO CURRENT BLOCK
   002710  777777 2220 10         4921          LDX     X,-Q$OFFST+Q$LINK,0 *PTR TO SUCCESSOR
   002711  000020 2230 11         4922          LDX     Y,T$TEMP8,T       GET BACK PTR TO PREDECESSOR
END OF BINARY CARD IOS00079
   002712  000000 7420 13         4923          STX     X,0,Y             REMOVE FROM LIST
                                  4924 *
                                  4925 *        ADD JOB TO Q$TASK QUEUE
                                  4926 *
   002713  005162 7170 00         4927          XED     Q$XADD+Q$TASK     PLACE ON Q$TASK
                                  4928 *
                                  4929 *        TEST FOR DONE
                                  4930 *
   002714  000025 2250 11         4931          LDX     Q,T$TEMP3,T       RESTORE QADDRESS
   002715  000001 1000 15         4932          CMPX    0,Q$LAST,Q        TEST FOR LAST
   002716  002722 6010 00         4933          TNZ     SKDR9             NOPE, CONTINUE
   002717  000001 6230 13         4934          EAX     Y,-Q$LINK+Q$OFFST,Y *UPDATE
   002720  000001 7430 15         4935          STX     Y,Q$LAST,Q        Q$LAST PTR
```

```
002721  002724 7100 00        4936          TRA      SKDRX              AND EXIT                                  1570
                              4937 *                                                                              1580
                              4938 *        FUDGE IN ORDER TO LOOP THRU LIST AFTER REMOVAL                      1590
                              4939 *                                                                              1600
                 002722       4940 SKDR9    BSS      0                                                            1610
002722  777777 2200 10        4941          LDX      0,-Q$OFFST+Q$LINK,0 *MOVE TO NEXT BLOCK                      1620
002723  002606 7100 00        4942          TRA      SKR0.1             AND LOOP WITHOUT DISTURBING PREDECESSOR   1630
                              4943 *                                                                              1640
                              4944 *        DONE, BUG REGISTERS, AND RETURN                                       1650
                              4945 *                                                                              1660
                 002724       4946 SKDRX    BSS      0                                                            1670
                 002724       4947          BUGXR    (X,Y,Z,Q)          BUG REGISTERS                             1680
                 525245            BUGBUG   SET      BUGBUG+1
002724  525245 2220 03                      LDX      X,BUGBUG,DU
002725  525245 2230 03                      LDX      Y,BUGBUG,DU
002726  525245 2240 03                      LDX      Z,BUGBUG,DU
002727  525245 2250 03                      LDX      Q,BUGBUG,DU
002730  000025 7270 11        4948          LXL      L,T$TEMP3,T        RETRIEVE RETURN ADDRESS                   1690
                 002731       4949          BUG      (T$TEMP3,T)        BUG IT (BOTH HALVES)                      1700
                 525246            BUGBUG   SET      BUGBUG+1
002731  525246 2200 03                      LDX      0,BUGBUG,DU
002732  000025 7400 11                      STX      0,T$TEMP3,T
002733  000025 4400 11                      SXL      0,T$TEMP3,T
002734  000000 7100 17        4950          TRA      0,L                RETURN TO CALLER                          1710
                              4951 *$*      DISK     SKEDCK                                                       1720
```

```
                  002735        USE     CODE
                                HEAD    R
                          *
                          *
                          *                             CHECK
                          *
                          *     THESE ROUTINES DETERMINE WHETHER OR NOT ENOUGH RESOURCES
                          *     OF A SPECIFIC TYPE ARE AVAILABLE.  THE DECISION IS BASED ON
                          *     OBVIOUSLY THE NUMBER OF FREE UNITS OF THE REQUESTED TYPE AND
                          *     ON A 'JOHN HAMM' POLICY RULE (SUBJECT TO CHANGE WITHOUT
                          *     NOTIFICATION).  IF THE REQUEST CAN BE SATIFIED, THEN THE
                          *     ROUTINES EXIT TO R$SKDR2; OTHERWISE R$SKDR3.
                          *
                          *     ENTER WITH
                          *             C(A) = NUMBER REQUESTED
                          *             C(X) = PTR TO DEVICE HEADER OF TYPE REQUESTED
                          *     RETURNS
                          *             R$SKDR2  IF SATIFIABLE
                          *             R$SKDR3  OTHERWISE
                          *
                  002735  SCPCK BSS     0               CARD PUNCH CHECK
                  002735  SCRCK BSS     0               CARD READER CHECK
                  002735  SMTCK BSS     0               MAG TAPE CHECK
002735  000005 1150 12          CMPA    AVAIL,X         ARE THAT MANY AVAILABLE?
002736  002635 6000 00          TZE     SKDR2           YES, NO POLICY RULE TO CHECK HERE
END OF BINARY CARD IOS00080
002737  002635 6040 00          TMI     SKDR2           YES,            DITTO
002740  002640 7100 00          TRA     SKDR3           NO, SO SOLLY PREASE
                          *
                          *
                          *     YE OLDE LINE PRINTERS
                          *
                          *     CURRENT IMPLEMENTED ALGORITHM:
                          *     GIVEN N PRINTERS, WHERE N>1, THEN WE'LL POTENTIALLY ALLOW N-1
                          *     LONG JOBS (SEE R$LJOBS) TO RUN SIMULTANEOUSLY.  IF N=1, THEN
                          *     WE'LL RUN SHORT, LONG, AND MEDIUM JOBS IN THAT ORDER.
                          *
                  002741  SLPCK BSS     0               LINE PRINTER CHECK
002741  000002 1150 07          CMPA    2,DL            ALLOW ONLY ONE PRINTER PER JOB
002742  777777 6030 00          TRC     $ERROR          ***PROBLEM
002743  000004 2360 12          LDQ     OMAX,X          GET NUMBER OF PRINTERS WE OWN
002744  000002 1760 07          SBQ     2,DL            PART OF POLICY RULE
002745  002735 6040 00          TMI     SCPCK           ALL RIGHT TO RUN LONG JOB
002746  005674 1160 00          CMPQ    LJOBS           HAVE 2 OR MORE PRINTERS, RUN LJOB?
002747  002640 6040 00          TMI     SKDR3           HAVE OMAX-1 LONG JOBS ALREADY GOING
002750  002735 7100 00          TRA     SCPCK           OK TO TRY TO RUN A LONG JOB
```

```
                   002751  USE    CODE
                           HEAD   R
                         *
                         *
                         *                            ALLOCATE
                         *
                         *        THIS ROUTINE ALLOCATES A SPECIFIC DEVICE TYPE.  IF A LONG
                         *        PRINTER TYPE, THEN THE LONG PRINTER JOB COUNTER 'LJOB' IS
                         *        BUMPED.
                         *
                         *
                   002751  SCPAL  BSS    0             CARD PUNCH ALLOCATE
                   002751  SCRAL  BSS    0             CARD READER ALLOCATE
                   002751  SMTAL  BSS    0             MAG TAPE ALLOCATE
                   002751         GETP   A             GET THE PERIPHERAL
002751  002377 7070 00            TSX    L,R$GETP
002752  002700 7100 00            TRA    SKDR7         AND RETURN WITH IT ALLOCATED
                         *
                         *
                   002753  SLPAL  BSS    0             LINE PRINTER ALLOCATE
002753  000012 2360 16            LDQ    J$SIZE,J      GET SIZE OF THE JOB
002754  000003 2220 16            LDX    X,J$TYPE,J    GET THE OUTPUT TYPE (512 OR 320 FORMAT)
002755  000001 3620 03            ANX    X,$TYPMK,DU   MASK TO TYPE
002756  002763 7160 12            XEC    LPTB1,X       TEST FOR LONG JOB
002757  002751 6000 00            TZE    SMTAL         EXIT IF NOT A LONG PRINTER JOB
002760  002751 6020 00            TNC    SMTAL         EXIT IF NOT A LONG PRINTER JOB
002761  005674 0540 00            AOS    LJOBS         OTHERWISE BUMP LONG JOB COUNTER
002762  002751 7100 00            TRA    SMTAL         USE STANDARD ROUTINE
                         *
                         *
                   002763  LPTB1  BSS    0             PRINTER DECISION TABLE
002763  005672 1160 00            CMPQ   M.512         DECISION IF 512 FORMAT
END OF BINARY CARD IOS00081
002764  005673 1160 00            CMPQ   M.320         DECISION IF 320 FORMAT
                         *
                   005667         USE    STORE         RUN TIME PATCHABLE
                   005670         EVEN
005670  000000000235       S.512  VFD    36/5000/32+1  5000 OR LESS IS A SHORT JOB
005671  000000000254       S.320  VFD    36/5500/32+1  DITTO FOR 320 FORMAT
005672  000000001751       M.512  VFD    36/32000/32+1 32000 IS TOPS FOR MEDIUM JOB
005673  000000002106       M.320  VFD    36/35000/32+1 DITTO FOR 320 FORMAT
                         *
005674  000000000000       LJOBS  DEC    0             NUMBER OF LONG PRINTER JOBS CURRENTLY ACTIVE
                   002765         USE    PREVIOUS
                           *$*    DISK   SCRATCH
```

R UTILITY -- SCRATCH INPUT QUEUE FILE

```
                  002765        USE     CODE
                                HEAD    R
                          *
                          *
                          *                         SCRATCH INPUT QUEUE FILE
                          *
                          *     THIS IS AN ASYNCHRONOUS TASK DESIGNED TO SCRATCH AN INPUT
                          *     QUEUE FILE (E.G. PRINT-FILE-QUEUE).  PASSED THE FRN OF THE
                          *     QUEUE FILE, IT DETERMINES IF THE READ POINTER IS AT THE END
                          *     OF THE FILE.  IF SO, AND THE BUSY COUNT IS ZERO, THEN THE
                          *     SCRATCH IS ALLOWED.  OTHERWISE THE READ POINTER IS RESET AND
                          *     THE TASK IS TERMINATED.
                          *
                          *     ENTER BY
                          *             CREATED TASK R$SCR
                          *     ENTER WITH
                          *             C(T$TEMP1,T) = FRN OF INPUT QUEUE FILE
                          *             C(T$TEMP2,T) = PTR TO NCB
                          *     RETURNS
                          *             TERMINATES -- ASYNCHRONOUS TASK
                          *
                  002765    SCR         BSS     0
                  002765                LOCK    (T$TEMP1,T)       LOCK THE FILE
002765  000736 7000 00                  TSX     0,$LOCK
002766  000027 0000 11                  ARG     T$TEMP1,T
                  002767                CHECK   SCR1,B$BZ,SCR,B$LOCK,SCR
002767  000000 7200 11                  LXL     0,T$SRW1,T
002770  000077 3600 03                  ANX     0,B$STMK,DU
002771  002777 6000 00                  TZE     SCR1
002772  000003 1000 03                  CMPX    0,B$BZ,DU
002773  002765 6000 00                  TZE     SCR
002774  000013 1000 03                  CMPX    0,B$LOCK,DU
002775  002765 6000 00                  TZE     SCR
002776  777777 7100 00                  TRA     $ERROR
                  002777    SCR1        BSS     0                 FILE LOCKED
002777  000026 2220 11                  LDX     X,T$TEMP2,T       GET PTR TO NCB
003000  000032 2340 12                  SZN     C$BUSY,X          TEST COUNT
003001  003037 6010 00                  TNZ     SCR5              BUSY, EXIT
                  003002                GETC    ($QBFSZ,DL)       GET A WORKING BUFFER
END OF BINARY CARD IOS00082
003002  000100 2350 07                  LDA     $QBFSZ,DL
003003  001152 7070 00                  TSX     L,R$GETC
003004  000025 7550 11                  STA     T$TEMP3,T         SAVE FOR READ
                  003005    SCR2        READ    (T$TEMP1,T),(T$TEMP3,T),(1,DU),(0,DU)  READ
003005  000536 7000 00                  TSX     0,$READ
003006  000027 0000 11                  ARG     T$TEMP1,T
003007  000025 0000 11                  ARG     T$TEMP3,T
003010  000001 0000 03                  ARG     1,DU
003011  000000 0000 03                  ARG     0,DU
                  003012                CHECK   SCR7,B$BZ,SCR2,B$EOF,SCR3
```

```
              R                          UTILITY -- SCRATCH INPUT QUEUE FILE

      003012  000000 7200 11                  LXL     0,T$SRW1,T
      003013  000077 3600 03                  ANX     0,B$STMK,DU
      003014  003051 6000 00                  TZE     SCR7
      003015  000003 1000 03                  CMPX    0,B$BZ,DU
      003016  003005 6000 00                  TZE     SCR2
      003017  000016 1000 03                  CMPX    0,B$EOF,DU
      003020  003022 6000 00                  TZE     SCR3
      003021  777777 7100 00                  TRA     $ERROR
                         003022  5075 SCR3    BSS     0                                                430
                         003022  5076         SCR     (T$TEMP1,T),(0,DU)  SCRATCH THE FILE             440
      003022  000612 7000 00                  TSX     0,$SCR
      003023  000027 0000 11                  ARG     T$TEMP1,T
      003024  000000 0000 03                  ARG     0,DU
                         003025  5077         CHECK   SCR4,B$BZ,SCR3                                   450
      003025  000000 7200 11                  LXL     0,T$SRW1,T
      003026  000077 3600 03                  ANX     0,B$STMK,DU
      003027  003033 6000 00                  TZE     SCR4
   END OF BINARY CARD IOS00083
      003030  000003 1000 03                  CMPX    0,B$BZ,DU
      003031  003022 6000 00                  TZE     SCR3
      003032  777777 7100 00                  TRA     $ERROR
                         003033  5078 SCR4    BSS     0                 FILE SCRATCHED                 460
      003033  000026 2220 11     5079         LDX     X,T$TEMP2,T       GET BACK PTR TO NCB            470
      003034  000031 4500 12     5080         STZ     C$QFLOC,X         RESET FLOC PTR                 480
                         003035  5081         RELC    (T$TEMP3,T)       RELEASE BUFFER                 490
      003035  000025 2350 11                  LDA     T$TEMP3,T
      003036  001252 7070 00                  TSX     L,R$RELC
                         003037  5082 SCR5    BSS     0                 UNLOCK IT                      500
                         003037  5083         UNLCK   (T$TEMP1,T)                                      510
      003037  000746 7000 00                  TSX     0,$UNLCK
      003040  000027 0000 11                  ARG     T$TEMP1,T
                         003041  5084         CHECK   SCR6,B$BZ,SCR5                                   520
      003041  000000 7200 11                  LXL     0,T$SRW1,T
      003042  000077 3600 03                  ANX     0,B$STMK,DU
      003043  003047 6000 00                  TZE     SCR6
      003044  000003 1000 03                  CMPX    0,B$BZ,DU
      003045  003037 6000 00                  TZE     SCR5
      003046  777777 7100 00                  TRA     $ERROR
                         003047  5085 SCR6    BSS     0                 DONE                           530
                         003047  5086         RELT                      RELEASE TCB                    540
      003047  001477 7000 00                  TSX     0,T$RELT
                         003050  5087         EXIT                      EXIT                           550
      003050  003074 7100 00                  TRA     $EXIT
                                 5088 *                                                                560
                                 5089 *                                                                570
                                 5090 *                                                                580
                         003051  5091 SCR7    BSS     0                 FILE NOT EMPTY                 590
      003051  000000 2200 11     5092         LDX     0,T$SRW1,T        GET NUMBER OF UNITS TRANSFERRED 600
      003052  003022 6000 00     5093         TZE     SCR3              NONE, THEN SCRATCH THE FILE    610
                         003053  5094 SCR8    BSS     0                                                620
```

```
                          003053    5095        SPTR    (T$TEMP1,T),(-1,DU)  SET POINTER BACK ONE      630
   003053  000623 7000 00                       TSX     0,$SPTR
   003054  000027 0000 11                       ARG     T$TEMP1,T
   003055  777777 0000 03                       ARG     -1,DU
                          003056    5096        CHECK   SCR4,B$BZ,SCR8                                 640
 END OF BINARY CARD IOS00084
   003056  000000 7200 11                       LXL     0,T$SRW1,T
   003057  000077 3600 03                       ANX     0,B$STMK,DU
   003060  003033 6000 00                       TZE     SCR4
   003061  000003 1000 03                       CMPX    0,B$RZ,DU
   003062  003053 6000 00                       TZE     SCR8
   003063  777777 7100 00                       TRA     $ERROR
                                    5097 *$*    DISK    JOBTAB                                         650
```

R JOB NUMBER ASSIGNMENT AND TABLE

```
                        003064              USE     CODE
                                            HEAD    J
                                   *
                                   *
                                   *                                  JOB NUMBER ASSIGNMENT
                                   *
                                   *        THE JOB NUMBER IS RETURNED IN XR-X
                                   *        CLOBERS C(X0)
                                   *
                        003064     JNUMB    BSS     0
003064  000000 2220 03                      LDX     X,0,DU            INITIALIZE FOR SEARCH
003065  000000 2350 07                      LDA     0,DL              CLEAR AL
003066  040300 5202 01                      RPT     MAXJB,1,TZE
003067  005700 1150 12                      CMPA    JTAB,X            SEARCH FOR FREE JOB NUMBER
003070  777777 6010 00                      TNZ     $ERROR            ***OY VEH!
003071  005701 1220 03                      SBLX    X,JTAB+1,DU       COMPUTE JOB NUMBER
003072  005700 7460 12                      STX     J,JTAB,X          MARK IT ALLOCATED TO US
003073  000000 7100 17                      TRA     0,L               RETURN TO CALLER
                                   *
                                   *
                                   *
                                   *                                  JOB TABLE
                                   *
                                   *        THE JOB TABLE IS A TABLE OF ONE WORD ENTRIES, INDEXED
                                   *        BY THE JOB NUMBER.  INFORMATION IN THE JOB TABLE IS
                                   *        THAT WHICH MUST BE LOCATED OR MATCHED FOR CONSISTENCY
                                   *        CHECKS AND FOR DEBUGGING PURPOSES.  THE TABLE CONTAINS
                                   *        MAXJB ENTRIES.
                                   *
                        005675              USE     STORE
                        000020     MAXJB    EQU     16                MAXIMUM NUMBER OF JOBS IN THE SYSTEM
                                   *
                                   *
                        005700              EIGHT                     TO MAKE DEBUGGING EASIER
                        005700     JTAB     BSS     0                 JOB TABLE
                        005700              DUP     1,MAXJB           (UPPER) PTR TO JCB/ (LOWER) PTR TO DEVICE
005700  000000 000000                       ZERO    0,0               INITIALLY OFF
END OF BINARY CARD IOS00085
                        003074              USE     PREVIOUS
                                   *$*      DISK    XIT
```

```
        J                      EXIT

                 003074   5139         USE     CODE                                        110
                          5140         HEAD                                                120
                          5141 *                                                           130
                          5142 *                                                           140
                          5143 *                            EXIT                           150
                          5144 *                                                           160
                          5145 *       COMPLETE ONE TASK AND BEGIN ANOTHER FROM Q$TASK QUEUE 170
                          5146 *                                                           180
                          5147 *       ENTER BY                                            190
                          5148 *               TRA $EXIT                                   200
                          5149 *       RETURN TO NEXT TASK                                 210
                          5150 *       RETURNS WITH                                        220
                          5151 *               C(J) = JOB NUMBER                           230
                          5152 *               C(T) = TRAP BLOCK                           240
                          5153 *               C(L) = TASK-START-ADDRESS                   250
                          5154 *                                                           260
                 003074   5155 EXIT    BSS     0               ENTRY POINT                 270
                 003074   5156         CKPT                    DEBUGGING                   280
  003074  000474 7170 00               XED     X$CKPT
                          5157         INHIB   SAVE,ON         LOCK OUT UNWANTED INTERRUPTS 290
                 003075   5158 EXIT1   BSS     0               TIME TO FOOL WITH THE Q$TASK QUEUE 300
  003075  005161 6202 00  5159         EAX     0,Q$FIRST+1+Q$TASK  ADDRESS OF FIRST ELEMENT 310
  003076  005161 1002 00  5160         CMPX    0,Q$LAST+Q$TASK *DOES LAST POINT TO IT?     320
  003077  003135 6002 00  5161         TZE     WAIT                                        330
  003100  005160 2212 00  5162         LDX     T,Q$FIRST+Q$TASK *OFFSET POINTER TO BLOCK   340
  003101  777777 6002 00  5163         TZE     $ERROR          ***PROBLEM                  350
  003102  005161 1012 00  5164         CMPX    T,Q$LAST+Q$TASK *IS THIS LAST?              360
  003103  003106 6012 00  5165         TNZ     *+3             NO                          370
  003104  005161 6202 00  5166         EAX     0,Q$FIRST+1+Q$TASK  YES, SET THIS QUEUE     380
END OF BINARY CARD IOS00086
  003105  005161 7402 00  5167         STX     0,Q$LAST+Q$TASK *TO EMPTY STATUS            390
  003106  000004 1212 03  5168         SBLX    T,Q$OFFST,DU    RELATE THE BEGINNING OF BLOCK 400
  003107  777777 6002 00  5169         TZE     $ERROR          ***DBG                      410
  003110  777777 6042 00  5170         TMI     $ERROR          ***DBG                      420
  003111  000003 2222 11  5171         LDX     X,Q$LINK,T      GET OFFSET POINTER TO NEXT BLOCK 430
  003112  005160 7422 00  5172         STX     X,Q$FIRST+Q$TASK *AND MAKE IT NOW FIRST     440
                          5173         INHIB   RESTORE         RESUME NORMAL TELECAST      450
  003113  000006 7260 11  5174         LXL     J,T$JCB,T       RESTORE JCB POINTER         460
  003114  777777 6000 00  5175         TZE     $ERROR          ***DBG                      470
  003115  777777 6040 00  5176         TMI     $ERROR          ***DBG                      480
  003116  000004 2270 11  5177         LDX     L,T$TRA,T       AND TRANSFER ADDRESS        490
  003117  777777 6000 00  5178         TZE     $ERROR          ***DBG                      500
  003120  777777 6040 00  5179         TMI     $ERROR          ***DBG                      510
                 003121   5180         BUGU    (T$TRA,T)       BUG RETURN                  520
                 525247           BUGBUG SET   BUGBUG+1
  003121  525247 2200 03               LDX     0,BUGBUG,DU
  003122  000004 7400 11               STX     0,T$TRA,T
                 003123   5181         BUGXR   (0,X,Y,Z,Q)     BUG THE REGISTERS           530
                 525250           BUGBUG SET   BUGBUG+1
  003123  525250 2200 03               LDX     0,BUGBUG,DU
```

```
                                          EXIT

    003124  525250 2220 03                      LDX     X,BUGBUG,DU
    003125  525250 2230 03                      LDX     Y,BUGBUG,DU
    003126  525250 2240 03                      LDX     Z,BUGBUG,DU
    003127  525250 2250 03                      LDX     Q,BUGBUG,DU
                      003130     5182           BUGA                                            540
                      525251          BUGBUG    SET     BUGBUG+1
    003130  525251 2350 03                      LDA     BUGBUG,DU
    003131  525251 2750 07                      ORA     BUGBUG,DL
                      003132     5183           BUGQ                                            550
                      525252          BUGBUG    SET     BUGBUG+1
    003132  525252 2360 03                      LDQ     BUGBUG,DU
  END OF BINARY CARD IOS00087
    003133  525252 2760 07                      ORQ     BUGBUG,DL
    003134  000000 7100 17       5184 EXIT2     TRA     0,L             AND AWAAAY WE GO!       560
                                 5185 *                                                         570
                                 5186 *                                                         580
                                 5187 *                 WAIT FOR SOMETHING TO HAPPEN            590
                                 5188 *                                                         600
                                 5189 *                                                         610
                                 5190           INHIB   SAVE,ON                                 620
    003135  000017 2202 03       5191 WAIT      LDX     0,$,PAUSE,DU    PAUSE AND START AT      630
    003136  000000 0012 00       5192           MME                     ANY INTERRUPT           640
                                 5193           INHIB   RESTORE                                 650
    003137  003075 7100 00       5194           TRA     EXIT1           SKIP CHECKPOINT         660
                                 5195 *$*       DISK    NOTC                                    690
```

```
003140          USE     CODE
                HEAD    C
      *
      *
      *         COMMUNICATIONS NETWORK STRUCTURE
      *
      *              THERE EXEISTS A PRIVATE COMMUNICATIONS NETWORK AMONG THE
      *         THE MONITOR AND ALL OF ITS SUB-MODULES (I.E. PERIPHERAL DRIVERS).
      *         AT STARTUP TIME FOR THE MONITOR, IT CREATES THE NETWORK BY
      *         OPENING THREE SCRATCH EVENTS AND PASSING A FRN OF EACH TO EACH
      *         SUB-MODULES SPAWNED.  FOR THE SUB-MODULES, THESE EVENTS ARE
      *         REFERENCED BY CANONICAL NUMBERS:
      *
      *                 FRN0 --
      *         THIS IS THE COMMAND EVENT FOR THE DRIVERS.  EACH DRIVER IS
      *         ALLOWED NOTIFY ACCESS ONLY.  COMMANDS ARE CHANNELED TO THE
      *         SPECIFIED SUB-MODULE BY THE STATE WHEN CAUSED.
      *
      *                 FRN1 --
      *         THIS IS THE COMMAND REPLY EVENT FOR THE DRIVERS.  EACH DRIVER
      *         IS ALLOWED CAUSE ACCESS ONLY.  IN ORDER TO INFORM THE
      *         MONITOR THE PERIPHERAL DRIVER CAUSES THIS EVENT WITH ITS STATE.
      *
      *                 FRN2 --
      *         AS IMPLIED ABOVE, FRN0 AND FRN1 ARE AN INPUT/OUTPUT PAIR.
      *         FRN2 HOWEVER IS NOT PAIRED AT ALL.  THE MONITOR USES THIS
      *         EVENT AS A PASS EVENT SENDING FILES TO BE PROCESSED AND DEVICES
      *         DOWN TO ITS SONS.  THE SONS NEVER EVER PASS ANYTHING BACK TO
      *         THE FATHER.  THEY SIMPLY CLOSE FILES.
      *
      *         MESSAGE FORMATS:  RETURNED IN T$SRW2,T (UPPER)
      *
      *         FOR FRN2 --
      *                 BITS 0 - 3 = JOB NUMBER (WITH 0 ILLEGAL)
      *                 BITS 4     = BANNER (ON MEANS SUPPLY BANNER)
      *                 BITS 5     = OUTPUT MODE:  512/ 320 (ON MEANS 320)
      *                 BITS 6 -17 = START ADDRESS (IN ELEMENTS)
      *
      *         FOR FRN1 --
      *                 BITS 0 - 3 = MUST BE ZERO
      *                 BITS 4 - 7 = COMMAND
      *                 BITS 8 -14 = <NOT USED>
      *                 BITS 15-17 = DEVICE UNIT NUMBER (0-7)
      *
      *
      *         FOR FRN0 --
      *                 BITS 0 - 3 = JOB NUMBER (MUST BE ZERO)
      *                 BITS 4 - 7 = COMMAND
      *                 BITS 8 -14 = <NOT USED>
      *                 BITS 15-17 = DEVICE UNIT NUMBER (0 THRU 7)
```

C COMMUNICATIONS -- SEND MESSAGE

```
                  003140    5248         USE     CODE                                          630
                            5249         HEAD    C                                             640
                            5250 *                                                             650
                            5251 *                                                             660
                            5252 *                           SEND A MESSAGE                    670
                            5253 *                                                             680
                            5254 *       THIS TASK SENDS A 36 BIT MESSAGE, WHICH IS BIT CODED TO THE    690
                            5255 *       MONITOR ON THE OUTPUT EVENT FILE FRN1.  THE STATE AND          700
                            5256 *       MESSAGE ARE PASSED TO IT IN T$TEMP1,T AND T$TEMP2,T,           710
                            5257 *       RESPECTIVELY.  AFTER SUCCESFULLY TRANSMITTING THE MESSAGE      720
                            5258 *       THE TASK EVAPORATES.                                  730
                            5259 *                                                             740
                            5260 *       ENTER WITH                                            750
                            5261 *               C(XT) = TBLOCK-ADDRESS                        760
                            5262 *               C(T$TEMP1,T) = STATE                          770
                            5263 *               C(T$TEMP2,T) = MESSAGE                        780
                            5264 *       SINCE AN ASYNCHRONOUS TASK, IT CAN USE ALL TEMP'S     790
                            5265 *       CALLS ONLY  R$RELT  WHEN DONE.                        800
                            5266 *                                                             810
                  003140    5267 MESSX   BSS     0               ENTRY POINT                   820
                  003140    5268         CAUSE   FRNO,(1,DU),(T$TEMP1,T),(T$TEMP2,T),(0,DU),(0,DU)    830
 003140  000767 7000 00                  TSX     0,$CAUSE
 003141  006217 0000 00                  ARG     FRNO
 003142  000001 0000 03                  ARG     1,DU
 003143  000027 0000 11                  ARG     T$TEMP1,T
 003144  000026 0000 11                  ARG     T$TEMP2,T
 003145  000000 0000 03                  ARG     0,DU
 003146  000000 0000 03                  ARG     0,DU
                  003147    5269         CHECK   MESS1,B$BZ,MESSX                              840
 003147  000000 7200 11                  LXL     0,T$SRW1,T
 003150  000077 3600 03                  ANX     0,B$STMK,DU
 003151  003155 6000 00                  TZE     MESS1
 003152  000003 1000 03                  CMPX    0,B$BZ,DU
 003153  003140 6000 00                  TZE     MESSX
 003154  777777 7100 00                  TRA     $ERROR
 003155  000000 2220 11     5270 MESS1   LDX     X,T$SRW1,T      GET NUMBER OF PEOPLE NOTIFIED 850
 003156  003140 6000 00     5271         TZE     MESSX           NONE, RE-SEND                 860
                            5272 *                                                             870
                            5273 *       MESSAGE SENT                                          880
                            5274 *                                                             890
                  003157    5275         RELT                    RELEASE TRAP BLOCK            900
 003157  001477 7000 00                  TSX     0,T$RELT
                  003160    5276         EXIT                    EVAPORATE                     910
END OF BINARY CARD IOS00088
 003160  003074 7100 00                  TRA     $EXIT
```

```
                     003161      5279          USE     CODE                                                        940
                                 5280          HEAD    C                                                           950
                                 5281 *                                                                            960
                                 5282 *                                   CTRAP SERVICE                            970
                                 5283 *                                                                            980
                                 5284 *        THIS SUBROUTINE IS ENTERED WHENEVER AN OUTSTANDING                  990
                                 5285 *        NOTIFY IS CAUSED.  THE ROUTINE ASCERTAINS THE REASON               1000
                                 5286 *        FOR THE CAUSE AND TRANSFERS CONTROL TO THE AP-                     1010
                                 5287 *        PRIATE SUBROUTINE.                                                 1020
                                 5288 *                                                                           1030
                                 5289 *        CALL WITH                                                          1040
                                 5290 *                C(XT) = CTRAP ADDRESS                                      1050
                                 5291 *        CALLS                                                              1060
                                 5292 *                C$NSRVX  (TO RE-ISSUE NOTIFY)                              1070
                                 5293 *                OR APPROPIATE SUBROUTINE                                   1080
                                 5294 *                                                                           1090
                     003161      5295 NSRV     BSS     0                                                          1100
                     003161      5296          CHECK   NSRV1,B$TLE,NSRVX                                          1110
003161  000000 7200 11                         LXL     0,T$SRW1,T
003162  000077 3600 03                         ANX     0,B$STMK,DU
003163  003167 6000 00                         TZE     NSRV1
003164  000011 1000 03                         CMPX    0,B$TLE,DU
003165  003171 6000 00                         TZE     NSRVX
003166  777777 7100 00                         TRA     $ERROR
                     003167      5297 NSRV1    BSS     0                 IT WAS REALLY CAUSED                     1120
003167  000005 7270 11           5298          LXL     L,C$RLINK,T       WELL, WHO GETS IT?                       1130
003170  000000 7100 17           5299          TRA     0,L               LET HIM HANDLE IT                        1140
```

```
         C                          COMMUNICATIONS -- RE-ISSUE NOTIFY

                  003171      5301        USE     CODE                                                    1160
                              5302        HEAD    C                                                       1170
                              5303 *                                                                      1180
                              5304 *                                                                      1190
                              5305 *                                  RE-ISSUE NOTIFY                     1200
                              5306 *                                                                      1210
                              5307 *      THIS SUBROUTINE SIMPLY RE-ISSUES THE NOTIFY FOR                 1220
                              5308 *      ANY NCB (NOTIFY CONTROL BLOCK).  AFTER A SUCCESSFUL OPERATION   1230
                              5309 *      THE TASK EVAPORATES.                                            1240
                              5310 *                                                                      1250
                              5311 *      CALL WITH                                                       1260
                              5312 *              C(XT) = NCB ADDRESS                                     1270
                              5313 *      CALLS                                                           1280
                              5314 *              T$GETT                                                  1290
                              5315 *              $SETUP                                                  1300
                              5316 *              $NOTIF                                                  1310
                              5317 *              T$RELT                                                  1320
                              5318 *                                                                      1330
                  003171      5319 NSRVX  BSS     0                                                       1340
                  003171      5320        GETT                    GET A TRAP BLOCK                        1350
   003171  001467 7000 00                 TSX     0,T$GETT
   003172  000000 6220 11     5321        EAX     X,0,T           SAVE A PTR TO IT                        1360
   003173  000005 2210 11     5322        LDX     T,T$LINK,T      LET T POINT TO ORIGINAL TCB             1370
   003174  003161 6200 00     5323        EAX     0,NSRV          RESTART ADDRESS                         1380
                  003175      5324        SETUP                   RESTART CTRAP                           1390
   003175  000510 7170 00                 XED     $SETUP
   003176  000000 6210 12     5325        EAX     T,0,X           GET BACK PTR TO NEW TCB                 1400
   003177  000005 2230 11     5326 NSX1   LDX     3,T$LINK,T      MAKE X3 POINT TO NCB                    1410
                  003200      5327        NOTIF   (ERN,3),(STATE,3)                                       1420
   003200  000756 7000 00                 TSX     0,$NOTIF
   003201  000024 0000 13                 ARG     ERN,3
   003202  000025 0000 13                 ARG     STATE,3
                  003203      5328        CHECK   NSX2,B$BZ,NSX1                                          1430
   003203  000000 7200 11                 LXL     0,T$SRW1,T
END OF BINARY CARD IOS00089
   003204  000077 3600 03                 ANX     0,B$STMK,DU
   003205  003211 6000 00                 TZE     NSX2
   003206  000003 1000 03                 CMPX    0,B$BZ,DU
   003207  003177 6000 00                 TZE     NSX1
   003210  777777 7100 00                 TRA     $ERROR
                  003211      5329 NSX2   BSS     0               SUCCESSFULLY RE-ISSUED NOTIFY           1440
                  003211      5330        RELT                    RELEASE TBLOCK                          1450
   003211  001477 7000 00                 TSX     0,T$RELT
                  003212      5331        EXIT                    EXIT                                    1460
   003212  003074 7100 00                 TRA     $EXIT
```

```
                                  5333 *                                                                1480
                                  5334 *                                                                1490
                                  5335 *                         NOTIFY CONTROL BLOCKS                  1500
                                  5336 *                                                                1510
                                  5337 *               PRINT-FILE-EVENT                                 1520
                                  5338 *                                                                1530
                         003213   5339        NCB      LPE,LPE,$INIT,-1,0,R$LPTAB+R$OMAX,Q$INP1,R$TYPLP,1  1540
                         005720               USE      STORE
                         005720               EIGHT
                         005720         LPE   BSS      0
   005720  000000 000035                      ZERO     0,B$TRO
   005721  000000 000000                      ZERO     0,0
   005722  000000 000000                      ZERO     0,0
   005723  000000 000000                      ZERO     0,0
   005724  000000 000000                      ZERO     **,0
   005725  000000 003301                      ZERO     0,$INIT
   005726  005720 005720                      ZERO     *-C$NCB,*-C$JCB
   005727  114120105040                       UASCI    1,LPE           ASCII ABREVIATION
                         005730               DUP      1,16-4
   005730  000000000000                       DEC      0
END OF BINARY CARD IOS00090
   005744  777777 0000 00               FRLPE ARG      -1
   005745  000000 000000                      ZERO     0,0
                         005746               DUP      1,2
   005746  000000000000                       DEC      0
   005750  777777 0000 00                     ARG      -1
   005751  000000000000                       DEC      0
   005752  000000000000                       DEC      0
   005753  005574 0000 00                     ARG      R$LPTAB+R$OMAX
   005754  005220 0000 00                     ARG      Q$INP1
   005755  000003 000001                      ZERO     R$TYPLP,1
   005756  777777777777                       DEC      -1
                         003213               USE      PREVIOUS
                         005750   5340 FRLPQ  EQU      LPE+QFRN        INPUT FILE RN                    1550
                                  5341 *                                                                1560
                                  5342 *                                                                1570
                                  5343 *               PUNCH-FILE-EVENT                                 1580
                                  5344 *                                                                1590
                         003213   5345        NCB      CPE,CPE,$INIT,-1,0,R$CPTAB+R$OMAX,Q$INP1,R$TYPCP,1  1600
                         005757               USE      STORE
                         005760               EIGHT
                         005760         CPE   BSS      0
   005760  000000 000035                      ZERO     0,B$TRO
   005761  000000 000000                      ZERO     0,0
END OF BINARY CARD IOS00091
   005762  000000 000000                      ZERO     0,0
   005763  000000 000000                      ZERO     0,0
   005764  000000 000000                      ZERO     **,0
   005765  000000 003301                      ZERO     0,$INIT
   005766  005760 005760                      ZERO     *-C$NCB,*-C$JCB
```

```
           C                                COMMUNICATIONS -- NOTIFY CONTROL BLOCKS

     005767  103120105040                            UASCI   1,CPE             ASCII ABREVIATION
                         005770                      DUP     1,16-4
     005770  000000000000                            DEC     0
     006004  777777 0000 00                   FRCPE  ARG     -1
     006005  000000 000000                           ZERO    0,0
                         006006                      DUP     1,2
     006006  000000000000                            DEC     0
 END OF BINARY CARD IOS00092
     006010  777777 0000 00                          ARG     -1
     006011  000000000000                            DEC     0
     006012  000000000000                            DEC     0
     006013  005554 0000 00                          ARG     R$CPTAB+R$OMAX
     006014  005220 0000 00                          ARG     Q$INP1
     006015  000001 000001                           ZERO    R$TYPCP,1
     006016  777777777777                            DEC     -1
                         003213                      USE     PREVIOUS
                         006010         5346 FRCPQ   EQU     CPE+QFRN         INPUT FILE RN                           1610
                                        5347 *                                                                        1620
                                        5348 *                                                                        1630
                                        5349 *               JOB-STREAM-SCHEDULER-EVENT                               1640
                                        5350 *                                                                        1650
                         525253         5351 BUGBUG  SET     BUGBUG+1         ***DBG                                  1660
                         003213         5352         NCB     JSS,JSS,O$ENTER,-1,0,$BUGBUG,$BUGBUG,$BUGBUG,$BUGBUG     1670
                         006017                      USE     STORE
                         006020                      EIGHT
                         006020              JSS     BSS     0
     006020  000000 000035                           ZERO    0,B$TRO
     006021  000000 000000                           ZERO    0,0
     006022  000000 000000                           ZERO    0,0
     006023  000000 000000                           ZERO    0,0
     006024  000000 000000                           ZERO    **,0
     006025  000000 001775                           ZERO    0,O$ENTER
     006026  006020 006020                           ZERO    *-C$NCB,*-C$JCB
     006027  112123123040                            UASCI   1,JSS             ASCII ABREVIATION
                         006030                      DUP     1,16-4
     006030  000000000000                            DEC     0
 END OF BINARY CARD IOS00093
     006044  777777 0000 00                   FRJSS  ARG     -1
     006045  000000 000000                           ZERO    0,0
                         006046                      DUP     1,2
     006046  000000000000                            DEC     0
     006050  777777 0000 00                          ARG     -1
     006051  000000000000                            DEC     0
     006052  000000000000                            DEC     0
     006053  525253 0000 00                          ARG     $BUGBUG
     006054  525253 0000 00                          ARG     $BUGBUG
     006055  525253 525253                           ZERO    $BUGBUG,$BUGBUG
     006056  777777777777                            DEC     -1
                         003213                      USE     PREVIOUS
                                        5353 *                                                                        1680
```

```
                          5354 *
                          5355 *                  FRN1-EVENT-LP (INPUT)
                          5356 *
                 525254   5357 BUGBUG SET         BUGBUG+1            ***DBG
                 003213   5358        NCB         LP1,LP1,C$COMD,-1,$LPST,$BUGBUG,$BUGBUG,$BUGBUG,$BUGBUG
                 006057               USE         STORE
                 006060               EIGHT
                 006060        LP1    BSS         0
END OF BINARY CARD IOS00094
   006060  000000 000035              ZERO        0,B$TRO
   006061  000000 000000              ZERO        0,0
   006062  000000 000000              ZERO        0,0
   006063  000000 000000              ZERO        0,0
   006064  000000 000000              ZERO        **,0
   006065  000000 003213              ZERO        0,C$COMD
   006066  006060 006060              ZERO        *-C$NCB,*-C$JCB
   006067  114120061040               UASCI       1,LP1               ASCII ABREVIATION
                 006070               DUP         1,16-4
   006070  000000000000               DEC         0
   006104  777777 0000 00      FRLP1  ARG         -1
   006105  003000 000000              ZERO        $LPST,0
                 006106               DUP         1,2
END OF BINARY CARD IOS00095
   006106  000000000000               DEC         0
   006110  777777 0000 00             ARG         -1
   006111  000000000000               DEC         0
   006112  000000000000               DEC         0
   006113  525254 0000 00             ARG         $BUGBUG
   006114  525254 0000 00             ARG         $BUGBUG
   006115  525254 525254              ZERO        $BUGBUG,$BUGBUG
   006116  777777777777               DEC         -1
                 003213               USE         PREVIOUS
                          5359 *
                          5360 *
                          5361 *                  $FRN1-EVENT-CP (INPUT)
                          5362 *
                 525255   5363 BUGBUG SET         BUGBUG+1            ***DBG
                 003213   5364        NCB         CP1,CP1,C$COMD,-1,$CPST,$BUGBUG,$BUGBUG,$BUGBUG,$BUGBUG
                 006117               USE         STORE
                 006120               EIGHT
                 006120        CP1    BSS         0
   006120  000000 000035              ZERO        0,B$TRO
   006121  000000 000000              ZERO        0,0
   006122  000000 000000              ZERO        0,0
   006123  000000 000000              ZERO        0,0
   006124  000000 000000              ZERO        **,0
   006125  000000 003213              ZERO        0,C$COMD
   006126  006120 006120              ZERO        *-C$NCB,*-C$JCB
   006127  103120061040               UASCI       1,CP1               ASCII ABREVIATION
                 006130               DUP         1,16-4
```

```
          C                        COMMUNICATIONS -- NOTIFY CONTROL BLOCKS

    006130  000000000000                          DEC     0
  END OF BINARY CARD IOS00096
    006144  777777 0000 00              FRCP1     ARG     -1
    006145  001000 000000                         ZERO    $CPST,0
                        006146                    DUP     1,2
    006146  000000000000                          DEC     0
    006150  777777 0000 00                        ARG     -1
    006151  000000000000                          DEC     0
    006152  000000000000                          DEC     0
    006153  525255 0000 00                        ARG     $BUGBUG
    006154  525255 0000 00                        ARG     $BUGBUG
    006155  525255 525255                         ZERO    $BUGBUG,$BUGBUG
    006156  777777777777                          DEC     -1
                        003213                    USE     PREVIOUS
                                  5365 *
                                  5366 *
                                  5367 *          $FRN1-EVENT-CR (INPUT)
                        525256    5368 BUGBUG SET  BUGBUG+1         ***DBG
                        003213    5369            NCB     CR1,CR1,C$COMD,-1,$CRST,$BUGBUG,$BUGBUG,$BUGBUG,$BUGBUG
                        006157                    USE     STORE
  END OF BINARY CARD IOS00097
                        006160                    EIGHT
                        006160          CR1       BSS     0
    006160  000000 000035                         ZERO    0,B$TRO
    006161  000000 000000                         ZERO    0,0
    006162  000000 000000                         ZERO    0,0
    006163  000000 000000                         ZERO    0,0
    006164  000000 000000                         ZERO    **,0
    006165  000000 003213                         ZERO    0,C$COMD
    006166  006160 006160                         ZERO    *-C$NCB,*-C$JCB
    006167  103122061040                          UASCI   1,CR1            ASCII ABREVIATION
                        006170                    DUP     1,16-4
    006170  000000000000                          DEC     0
  END OF BINARY CARD IOS00098
    006204  777777 0000 00              FRCR1     ARG     -1
    006205  002000 000000                         ZERO    $CRST,0
                        006206                    DUP     1,2
    006206  000000000000                          DEC     0
    006210  777777 0000 00                        ARG     -1
    006211  000000000000                          DEC     0
    006212  000000000000                          DEC     0
    006213  525256 0000 00                        ARG     $BUGBUG
    006214  525256 0000 00                        ARG     $BUGBUG
    006215  525256 525256                         ZERO    $BUGBUG,$BUGBUG
    006216  777777777777                          DEC     -1
                        003213                    USE     PREVIOUS
                                  5370 *
                                  5371 *
                                  5372 *                  COMMUNICATIONS FRN'S
                                  5373 *
```

```
                          006217     5374          USE     STORE                                                     1890
       006217  777777 0000 00        5375 FRN0     ARG     -1                 MON:  COMMAND OUTPUT                  1900
       006220  777777 0000 00        5376 FRN1     ARG     -1                 MON:  COMMAND INPUT                   1910
       006221  777777 0000 00        5377 FRN2     ARG     -1                 MON:  COMMAND PASS                    1920
                                     5378 *                                                                         1930
                                     5379 *                                                                         1940
                                     5380 *                PROCESS-ONE-EVENT (LPCP MODULE)                          1950
                                     5381 *                                                                         1960
                          525257     5382 BUGBUG SET       BUGBUG+1           ***DBG                                1970
                          006222     5383          NCB     PR1,LPCP,$CRASH,-1,0,BUGBUG,BUGBUG,BUGBUG,BUGBUG          1980
                          006222                   USE     STORE
                          006230                   EIGHT
                          006230          PR1      BSS     0
       006230  000000 000035                       ZERO    0,BSTRO
       006231  000000 000000                       ZERO    0,0
       006232  000000 000000                       ZERO    0,0
       006233  000000 000000                       ZERO    0,0
    END OF BINARY CARD IOS00099
       006234  000000 000000                       ZERO    **,0
       006235  000000 004074                       ZERO    0,$CRASH
       006236  006230 006230                       ZERO    *-C$NCB,*-C$JCB
       006237  114120103120                        UASCI   1,LPCP               ASCII ABREVIATION
                          006240                   DUP     1,16-4
       006240  000000000000                        DEC     0
       006254  777777 0000 00             FRPR1    ARG     -1
       006255  000000 000000                       ZERO    0,0
                          006256                   DUP     1,2
       006256  000000000000                        DEC     0
       006260  777777 0000 00                      ARG     -1
       006261  000000000000                        DEC     0
    END OF BINARY CARD IOS00100
       006262  000000000000                        DEC     0
       006263  525257 0000 00                      ARG     BUGBUG
       006264  525257 0000 00                      ARG     BUGBUG
       006265  525257 525257                       ZERO    BUGBUG,BUGBUG
       006266  777777777777                        DEC     -1
                          006267                   USE     PREVIOUS
                          006260     5384 FRPP     EQU     PR1+0FRN           SPAWNED FRN                           1990
                                     5385 *                                                                         2000
                                     5386 *                                                                         2010
                                     5387 *                PORCESS-TWO-EVENT (CR MODULE)                            2020
                                     5388 *                                                                         2030
                          525260     5389 BUGBUG SET       BUGBUG+1           ***DBG                                2040
                          006267     5390          NCB     PR2,CDRD,$CRASH,-1,0,BUGBUG,BUGBUG,BUGBUG,BUGBUG          2050
                          006267                   USE     STORE
                          006270                   EIGHT
                          006270          PR2      BSS     0
       006270  000000 000035                       ZERO    0,BSTRO
       006271  000000 000000                       ZERO    0,0
       006272  000000 000000                       ZERO    0,0
```

```
          C                              COMMUNICATIONS -- NOTIFY CONTROL BLOCKS

     006273  000000 000000                    ZERO     0,0
     006274  000000 000000                    ZERO     **,0
     006275  000000 004074                    ZERO     0,.CRASH
     006276  006270 006270                    ZERO     *-C$NCB,*-C$JCB
     006277  103104122104                     UASCI    1,CDRD              ASCII ABREVIATION
                       006300                 DUP      1,16-4
     006300  000000000000                     DEC      0
  END OF BINARY CARD IOS00101
     006314  777777 0000 00   .         FRPR2 ARG      -1
     006315  000000 000000                    ZERO     0,0
                       006316                 DUP      1,2
     006316  000000000000                     DEC      0
     006320  777777 0000 00                   ARG      -1
     006321  000000000000                     DEC      0
     006322  000000000000                     DEC      0
     006323  525260 0000 00                   ARG      BUGBUG
     006324  525260 0000 00                   ARG      BUGBUG
     006325  525260 525260                    ZERO     BUGBUG,BUGBUG
     006326  777777777777                     DEC      -1
                       006327                 USE      PREVIOUS
                       006320      5391 FRCR  EQU      PR2+QFRN           SPAWNED FRN                      2060
                                   5392 *$*   DISK     COMD                                                2070
```

C COMMUNICATIONS -- COMMANDS

```
                    003213      USE    CODE
                                HEAD   C
                           *
                           *
                           *                             COMMANDS
                           *
                           *    THIS ROUTINE DECODES A COMMAND SENT TO IT FROM A SUB-MODULE
                           *    IF A LEGAL COMMAND, THE APPROPRIATE ACTION IS TAKEN.  IF
                           *    ILLEGAL, WELL IT IS INGORED (MORE OFTEN THAN NOT)
                           *
                           *
                    003213 COMD  BSS    0                ENTER HERE FROM C$NSRV
003213  000001 2360 11          LDQ    T$SRW2,T          GET COMMAND
003214  000012 7720 00          QRL    18-4-4            RIGHT JUSTIFY IN QU
003215  000017 3760 03          ANQ    =017,DU           MASK TO COMMAND
003216  000010 1160 03          CMPQ   CMAX,DU           TEST VALIDITY OF COMMAND
END OF BINARY CARD IOS00102
003217  003231 6030 00          TRC    COMDX             NOPE; EXIT
003220  003221 7100 22          TRA    *+1,QU*           BRANCH TO SUBROUTINE
003221  003231 0000 00 CMDTB    ARG    COMDX             0 = INVALID (IGNORED)
003222  003232 0000 00          ARG    GET               1 = GET A PERIPHERAL
003223  003233 0000 00          ARG    KILL              2 = KILL PERIPHERAL NOW
003224  003236 0000 00          ARG    REL               3 = RELEASE PERIPHERAL WHEN NOT BUSY
003225  003231 0000 00          ARG    COMDX             4 = INVALID (XXXX)
003226  003247 0000 00          ARG    RSTRT             5 = RESTART A PERIPHERAL
003227  777777 0000 00          ARG    $ERROR            6 = DONE (NOT HERE ON THIS STATE)
003230  003241 0000 00          ARG    READY             7 = READY PERIPHERAL NAME
                    000010 CMAX  EQU    *-CMDTB          NUMBER OF COMMANDS LEGAL
                           *
                           *
                           *
                           *                             COMDX
                           *
                           *    RE-ISSUE NOTIFY ON THIS NCB
                           *
                    003231 COMDX BSS    0
003231  003171 7100 00          TRA    NSRVX             RE-ISSUE THE NOTIFY
```

```
                     003232           USE      CODE
                                      HEAD     C
                                *
                                *
                                *                            COMMAND GET
                                *
                                *     THIS SUBROUTINE 'GETS' THE SPECIFIED PERIPHERAL.
                                *
                                *
                     003232     GET   BSS      0             IGNORE
003232  003231 7100 00                TRA      COMDX         EXPECT NO REPLY--IGNORE
```

```
                         003233   5443        USE     CODE
                                  5444        HEAD    C
                                  5445 *
                                  5446 *
                                  5447 *                              COMMAND KILL
                                  5448 *
                                  5449 *      THIS ROUTINE TELLS THE OPERATOR THAT A PARTICULAR DEVICE
                                  5450 *      HAS BEEN STOPPED AND RELEASED -- KILLED.
                                  5451 *
                                  5452 *      ENTER WITH
                                  5453 *              C(XT) = NCB-ADDRESS
                                  5454 *
                         003233   5455 KILL   BSS     0
   003233  004753 2350 00         5456        LDA     KLMS            GET MESSAGE TO SEND
   003234  000027 7550 11         5457        STA     TSTEMP1,T       SAVE FOR LOGGING
                         004753   5458        USE     CONST
   004753  004754 0024 40         5459 KLMS   TALLYB  *+1,19+1,0
   004754  120105122111           5460        UASCI   5,PERIPHERAL KILLED:
 END OF BINARY CARD IOS00103
                         003235   5461        USE     PREVIOUS
   003235  003243 7100 00         5462        TRA     RDY1            HANDLE LIKE READY
```

```
        C                            COMMUNICATIONS -- COMMAND RELEASE

                      003236             USE     CODE
                                         HEAD    C
                                *
                                *
                                *                               COMMAND RELEASE
                                *
                                *        THIS ROUTINE RELEASES A PERIPHERAL AS SOON AS IT IS FREE
                                *
                                *        ENTER WITH
                                *                C(XT) = NCB-ADDRESS
                                *
                      003236      REL    BSS     0
003236  004761 2350 00                   LDA     CLMS           GET MESSAGE TO SEND
003237  000027 7550 11                   STA     T$TEMP1,T      SAVE FOR LOGGING
                      004761             USE     CONST
004761  004762 0024 40            CLMS   TALLYB  *+1,19+1,0
004762  120105122111                     UASCI   5,PERIPHERAL CLOSED:
                      003240             USE     PREVIOUS
003240  003243 7100 00                   TRA     RDY1           HANDLE LIKE READY
```

C COMMUNICATIONS -- COMMAND READY

```
                                  5484        HEAD    C                                                 1010
                                  5485 *                                                                1020
                                  5486 *                                                                1030
                                  5487 *                                COMMAND READY                   1040
                                  5488 *                                                                1050
                                  5489 *      THIS COMMAND TELLS THE OPERATOR TO PERFORM CERTAIN ACTIONS.  1060
                                  5490 *                                                                1070
                                  5491                                                                  1080
                        003241    5492 READY  BSS     0                                                 1090
    003241  004767 2350 00        5493        LDA     RDYMS           GET MESSAGE TO SEND               1100
    003242  000027 7550 11        5494        STA     T$TEMP1,T       SAVE FOR LOGGING                  1110
                        004767    5495        USE     CONST                                             1120
    004767  004770 0011 40        5496 RDYMS  TALLYB  *+1,8+1,0                                         1130
    004770  122105101104          5497        UASCI   2,READY:                                          1140
                        003243    5498        USE     PREVIOUS                                          1150
                        003243    5499 RDY1   BSS     0               PRINT NAME                        1160
 END OF BINARY CARD IOS00104
    003243  003262 7070 00        5500        TSX     L,PERI          GET PERIPHERAL INFORMATION        1170
    003244  000026 7420 11        5501        STX     X,T$TEMP2,T     SAVE DEVICE HEADER PTR            1180
    003245  000026 4440 11        5502        SXL     Z,T$TEMP2,T     SAVE UNIT PTR                     1190
    003246  003250 7100 00        5503        TRA     LOG             SEND BUILT UP MESSAGES            1200
```

```
          C                                COMMUNICATIONS -- COMMAND RESTART

                    003247       5505          USE     CODE
                                 5506          HEAD    C
                                 5507 *
                                 5508 *
                                 5509 *                                COMMAND RESTART
                                 5510 *
                                 5511 *        THIS COMMAND RESTARTS THE JOB ON THE SPECIFIED PER-
                                 5512 *        IPHERAL TO THE SPECIFIED ELEMENT NUMBER.
                                 5513 *
                    003247       5514 RSTRT    BSS     0
 003247  003231 7100 00          5515          TRA     COMDX                   EXPECT NO REPLY--IGNORE
```

```
        C                          COMMUNICATIONS -- LOG MESSAGE TO OP

                  003250     5517        USE     CODE                                         1340
                             5518        HEAD    C                                            1350
                             5519 *                                                           1360
                             5520 *                                                           1370
                             5521 *                            LOG MESSAGE TO OP              1380
                             5522 *                                                           1390
                  003250     5523 LOG    BSS     0                                            1400
                  003250     5524        LOGS                  LOG A MESSAGE TO THE OP SAYING 1410
003250  001517 7070 00                   TSX     L.O$LOGS
                  003251     5525        LOGC    (T$TEMP1.T)   THE CANNED MESSAGE             1420
003251  000027 2350 11                   LDA     T$TEMP1.T
003252  001543 7070 00                   TSX     L.O$LOGC
003253  000026 7240 11       5526        LXL     Z.T$TEMP2.T   FROM -- GET PTR TO UNIT        1430
003254  000000 6350 14       5527        EAA     R$ABBR.Z      POINT TO ABBREVIATION          1440
003255  000540 2750 07       5528        ORA     4*$TAL+$TAL+$TALYB,DL  ABBREVIATIONS ARE 4 CHARACTERS  1450
                  003256     5529        LOGC    A             PERIPHERAL NAME                1460
003256  001543 7070 00                   TSX     L.O$LOGC
                  003257     5530        LCRLF                 NEATNESS                       1470
003257  001600 7070 00                   TSX     L.O$LCRLF
                  003260     5531        LOGX                  FINISH UP                      1480
003260  001560 7070 00                   TSX     L.O$LOGX
003261  003231 7100 00       5532        TRA     COMDX         RE-ISSUE THE NOTIFY            1490
```

C COMMUNICATIONS -- PERIPHERAL INFORMATION (PERI)

```
                          003262        USE     CODE
                                        HEAD    C
                                  *
                                  *
                                  *                                 PERI
                                  *
                                  *     THIS ROUTINE CHECKS TO SEE IF THE PERIPHERAL NAME EXITS
                                  *     AND IF SO RETURNS A POINTER TO THE DEVICE ENTRY BLOCK,
                                  *     OTHERWISE IT EXITS
                                  *
                                  *     ENTER WITH
                                  *             C(XT) = NCP-ADDRESS
                                  *     RETURN WITH
                                  *             C(XZ) = DEVICE ENTRY BLOCK
                                  *             C(X0) = TYPE
                                  *
                          003262  PERI  BSS     0
  003262  000025 2350 11                LDA     C$STATE,T          GET STATE
  003263  000011 7710 00                ARL     18-9               MAP INTO TYPE
  003264  004743 2200 01                LDX     0,R$TABLE,AU       GET PTR TO DEVICE HEADER
  003265  000000 2240 10                LDX     Z,R$PTR,0          GET PTR TO FIRST DEVICE OF THIS TYPE
END OF BINARY CARD IOS00105
  003266  000001 2350 11                LDA     T$SRW2,T           GET BACK UNIT NUMBER
  003267  000022 7710 00                ARL     36-18              RIGHT JUSTIFY IN AL
  003270  000007 3750 07                ANA     7,DL               MASK TO UNIT NUMBER
  003271  000000 2750 14                ORA     R$ABBR,Z           GET NAME
  003272  000003 3360 10                LCQ     R$MAX,0            GET THE NUMBER OF DEVICES TO CHECK
  003273  000000 1150 14          PERI1 CMPA    R$ABBR,Z           TEST FOR MATCH
  003274  000000 6000 17                TZE     0,L                RETURN WITH Z POINTING TO DEVICE
  003275  000003 0240 03                ADLX    Z,R$DEVLN,DU       NO, SKIP TO NEXT DEVICE
  003276  000001 0760 07                ADQ     1,DL               TEST FOR DONE
  003277  003273 6040 00                TMI     PERI1              NO, LOOP
  003300  003231 7100 00                TRA     COMDX              INGORE MONITOR
                                  *$*   DISK    INIT
```

```
003301          USE     CODE
                HEAD
 *
 *
 *              THE INITIALIZATION ROUTINE IS CHARGED WITH THE RESPONSIBILITY
 *              OF COORDINATING THE ACTIVITIES OF THE EXTERNAL SYSTEMS' USERS
 *              AND THE ENTIRE MONITOR STRUCTURE.  FROM THE EXTERNAL SIDE,
 *              AS USERS PLACE DESCRIPTOR ITEMS OF FILES TO BE PRINTED OR PUNCHED
 *              IN THE PRINT AND PUNCH FILE QUEUES, RESPECTIVELY, AND CAUSE THE
 *              CORRESPONDING EVENT TO INDICATE THAT ACTION, THE 'INIT' ROUTINE
 *              IS NOTIFIED.  IT THEN MUST READ IN THE DESCRIPTOR FROM THE EXERNAL
 *              FILES, CREATE A CORRESPONDING INTERNAL REPRESENTATION, AND QUEUE
 *              IT UP TO BE RUN.
 *
 *              WITH REGARDS TO ACTUAL IMPLEMENTATION, THE FOLLOWING HAPPENS:
 *              CHECK TO SEE IF WE HAVE ANY PERIPHERALS NEEDED FOR THIS
 *              INPUT FILE QUEUE.  IF NOT, THERE IS NO NEED TO READ THE DES-
 *              CRIPTOR BECAUSE WE CAN'T RUN HIM (OWN NO PERIPHERALS).
 *              OTHERWISE GET A TCB AND A JCB FOR THE NEW JOB, MOVE IN DATA FROM
 *              THE NOTIFY CONTROL BLOCK (NCB) TO THE JOB CONTROL BLOCK (JCB).
 *                      LOCK THE INPUT FILE QUEUE
 *                      UPDATE THE INPUT FILE QUEUE
 *                      GET A READ BUFFER
 *                      READ DESCRIPTOR ITEM INTO CORE
 *                      OPEN NAMED FILE
 *                      RELEASE BUFFER
 *                      FINISH DESCRIBING INPUT IN JCB
 *                      PLACE THE NEW REQUEST ON A WAIT-WANITNG-TO-RUN QUEUE
 *                      UNLOCK THE INPUT FILE QUEUE
 *                      WAKE UP THE SCHEDULER
 *                      LOOP TO HIT EOF OF INPUT FILE QUEUE
 *                      UPON HITTING THE END, RE-ISSUE THE NOTIFY
```

```
MBR    01  09-17-71   09.937       INPUT/OUTPUT SCHEDULER -- IOS MONITOR                                              PAGE     184

                                          JOB -- SET UP

                         003301       5601         USE      CODE                                                          440
                                      5602         HEAD                                                                   450
                                      5603 *                                                                              460
                                      5604 *                                                                              470
                                      5605 *                                      SETUP                                   480
                                      5606 *                                                                              490
                         003301       5607 INIT    BSS      0                                                             500
      003301  000033 2340 31          5608         SZN      C$RUN,T*            TEST TO SEE IF ANY PERIPHERAL OPENED      510
      003302  003171 6000 00          5609         TZE      C$NSRVX             EXIT IF NONE OPENED                       520
                                      5610 *                                                                              530
                                      5611 *       LOCK THE INPUT FILE QUEUE                                              540
                                      5612 *                                                                              550
                         003303       5613 INITO   BSS      0                                                             560
                         003303       5614         LOCK     (C$QFRN,T)          LOCK THE INPUT QUEUE                      570
      003303  000736 7000 00                       TSX      0,$LOCK
      003304  000030 0000 11                       ARG      C$QFRN,T
                         003305       5615         CHECK    INTO.0,B$BZ,INITO,B$LOCK,INITO                                580
      003305  000000 7200 11                       LXL      0,T$SRW1,T
      003306  000077 3600 03                       ANX      0,B$STMK,DU
      003307  003315 6000 00                       TZE      INTO.0
      003310  000003 1000 03                       CMPX     0,B$BZ,DU
      003311  003303 6000 00                       TZE      INITO
      003312  000013 1000 03                       CMPX     0,B$LOCK,DU
   END OF BINARY CARD IOS00106
      003313  003303 6000 00                       TZE      INITO
      003314  777777 7100 00                       TRA      $ERROR
                         003315       5616 INTO.0  BSS      0                                                             590
                         003315       5617         GETT                         GET A NEW TASK BLOCK                      600
      003315  001467 7000 00                       TSX      0,T$GETT
                         003316       5618         GETJ                         GET A JOB CONTROL BLOCK (JCB)             610
      003316  001505 7000 00                       TSX      0,J$GETJ
      003317  000006 4460 11          5619         SXL      J,T$JCB,T           SAVE PTR TO JCB                           620
      003320  000005 4410 16          5620         SXL      T,J$TCB,J           SAVE PTR TO TCB***DBG                     630
      003321  000005 2220 11          5621         LDX      X,T$LINK,T          GET BACK PTR TO NCB                       640
      003322  000006 7420 11          5622         STX      X,T$NCB,T           SAVE PTR TO NCB                           650
      003323  000005 7420 16          5623         STX      X,J$NCB,J           SAVE PTR TO NCB***DBG                     660
                                      5624 *                                                                              670
                                      5625 *       MOVE DATA FROM NCB TO NEW TBLOCK                                       680
                                      5626 *                                                                              690
      003324  000030 2200 12          5627         LDX      0,C$QFRN,X          GET FRN OF FILE CORRESPONDING TO          700
      003325  000000 7400 16          5628         STX      0,J$QFRN,J          EVENT NOTIFIED                            710
      003326  000031 7200 12          5629         LXL      0,C$QFLOC,X         GET FILE SOURCE (READ) PTR                720
      003327  000001 7400 16          5630         STX      0,J$QFLOC,J         AND SAVE IT                               730
      003330  000017 2200 03          5631         LDX      0,A$LP,DU           GET ACCOUNTING NUMBER FOR LINE PRINTER    740
      003331  005720 1020 03          5632         CMPX     X,C$LPE,DU          SEE IF IT IS THE LINE PRINTER             750
      003332  003334 6000 00          5633         TZE      *+2                 YES                                       760
      003333  000016 2200 03          5634         LDX      0,A$CP,DU           NO. IT IS THE CARD PUNCH                  770
      003334  000004 4400 16          5635         SXL      0,J$ACODE,J         SAVE FOR ACCOUNTING                       780
      003335  000035 6230 12          5636         EAX      Y,C$RES,X           PTR TO RESOURCE NEEDS                     790
      003336  000013 6240 16          5637         EAX      Z,J$RES,J           PTR TO WHERE THEY GO IN JCB               800
```

JOB -- SET UP

```
    003337  000035 2350 12      5638          LDA     C$RES,X          COMPUTE STATE                     810
    003340  777777 3750 03      5639          ANA     -1,DU            FOR JOB INITIATION                820
 END OF BINARY CARD IOS00107
    003341  000011 7350 00      5640          ALS     9                ***WHEW;?!!                       830
    003342  000006 7550 16      5641          STA     J$STATI,J        SAVE FOR RUN                      840
                   003343       5642 INTO.1   BSS     0                                                  850
    003343  000027 7440 11      5643          STX     Z,T$TEMP1,T      SAVE PTR                          860
    003344  000000 2350 13      5644          LDA     0,Y              GET NEXT NEED                     870
    003345  000000 7550 14      5645          STA     0,Z              SAVE IN JCB                       880
    003346  003353 6040 00      5646          TMI     INIT1            TEST FOR DONE                     890
    003347  000000 6240 05      5647          EAX     Z,0,AL           GET NUMBER REQUESTED IN Z         900
    003350  000027 0240 11      5648          ADLX    Z,T$TEMP1,T      PLUS OLD PTR                      910
    003351  000001 0230 03      5649          ADLX    Y,1,DU           BUMP Y PTR TOO                    920
    003352  003343 7100 00      5650          TRA     INTO.1           LOOP                              930
                                5651 *                                                                   940
                                5652 *        UPDATE FILE                                                950
                                5653 *                                                                   960
                   003353       5654 INIT1    UPDATE  (J$QFRN,J)       UPDATE INPUT QUEUE                970
    003353  000726 7000 00                    TSX     0,$UPDATE
    003354  000000 0000 16                    ARG     J$QFRN,J
                   003355       5655          CHECK   INIT2,B$BZ,INIT1                                   980
    003355  000000 7200 11                    LXL     0,T$SRW1,T
    003356  000077 3600 03                    ANX     0,B$STMK,DU
    003357  003363 6000 00                    TZE     INIT2
    003360  000003 1000 03                    CMPX    0,B$BZ,DU
    003361  003353 6000 00                    TZE     INIT1
    003362  777777 7100 00                    TRA     $ERROR
                                5656 *                                                                   990
                                5657 *        GET A BUFFER TO READ IN NEXT JOB                          1000
                                5658 *                                                                  1010
                   003363       5659 INIT2    BSS     0                                                 1020
                   003363       5660          GETC    (QBFSZ,DL)       GET A BUFFER                     1030
    003363  000100 2350 07                    LDA     QBFSZ,DL
    003364  001152 7070 00                    TSX     L,R$GETC
    003365  000011 7550 16      5661          STA     J$BUF,J          SAVE BUFFER POINTER              1040
                                5662 *                                                                  1050
                                5663 *                                                                  1060
                                5664 *        READ INFORMATION INTO CORE                                1070
                                5665 *                                                                  1080
                   003366       5666 INIT3    BSS     0                                                 1090
                   003366       5667          READ    (J$QFRN,J),(J$BUF,J),(1,DU),(0,DU)                1100
    003366  000536 7000 00                    TSX     0,$READ
 END OF BINARY CARD IOS00108
    003367  000000 0000 16                    ARG     J$QFRN,J
    003370  000011 0000 16                    ARG     J$BUF,J
    003371  000001 0000 03                    ARG     1,DU
    003372  000000 0000 03                    ARG     0,DU
                   003373       5668          CHECK   INIT4,B$BZ,INIT3,B$EOF,INITX                      1110
    003373  000000 7200 11                    LXL     0,T$SRW1,T
    003374  000077 3600 03                    ANX     0,B$STMK,DU
```

JOB -- SET UP

```
    003375  003403 6000 00                    TZE      INIT4
    003376  000003 1000 03                    CMPX     0,B$RZ,DU
    003377  003366 6000 00                    TZE      INIT3
    003400  000016 1000 03                    CMPX     0,B$EOF,DU
    003401  003531 6000 00                    TZE      INITX
    003402  777777 7100 00                    TRA      $ERROR
                                 5669 *
                                 5670 *       INFORMATION READ
                                 5671 *
                      003403     5672 INIT4   BSS      0
    003403  000011 2220 16       5673         LDX      X,J$BUF,J          GET BACK BUFFER POINTER
    003404  000001 2200 12       5674         LDX      0,TNSZ,X           GET TREE-SIZE
    003405  003413 6010 00       5675         TNZ      INT4.2             GOT A TREE-SIZE
    003406  000001 2220 03       5676 INT4.1  LDX      X,1,DU             BUMP READ PTR TO NEXT
    003407  000001 0420 16       5677         ASX      X,J$QFLOC,J        IN JCB
    003410  000006 2220 11       5678         LDX      X,T$NCB,T          AND IN THE NCB
    003411  000031 0540 12       5679         AOS      C$QFLOC,X
    003412  003366 7100 00       5680         TRA      INIT3              AND READ AGAIN
                      003413     5681 INT4.2  BSS      0
    003413  000026 7400 11       5682         STX      0,T$TEMP2,T        SAVE IN TASK BLOCK
    003414  000002 2240 12       5683         LDX      Z,TYPE,X           GET TYPE
 END OF BINARY CARD IOS00109
    003415  000001 3640 03       5684         ANX      Z,TYPMK,DU         MASK TO TYPE ONLY
    003416  000003 7440 16       5685         STX      Z,J$TYPE,J         SAVE IN JBLOCK
    003417  000002 7240 12       5686         LXL      Z,DISP,X           GET DISPOSITION
    003420  000003 3640 03       5687         ANX      Z,DISMK,DU         MASK TO DISPOSITION
    003421  000003 4440 16       5688         SXL      Z,J$DISP,J         SAVE IN JBLOCK, ALSO
    003422  000003 2240 12       5689         LDX      Z,ACODE,X          GET ACODE FOR ACCOUNTING
    003423  000004 7440 16       5690         STX      Z,J$ACODE,J        SAVE IT
    003424  000004 6200 12       5691         EAX      0,TN,X             GET POINTER TO TREE-NAME
    003425  000027 7400 11       5692         STX      0,T$TEMP1,T        SAVE ONLY FOR OPEN
    003426  000003 2200 16       5693         LDX      0,J$TYPE,J         GET BACK TYPE
    003427  000001 3600 03       5694         ANX      0,TYPMK,DU         ISOLATE TYPE
    003430  004772 2200 10       5695         LDX      0,TABLE,0          GET CORRESPONDING ELEMENT SIZE
    003431  000025 7400 11       5696         STX      0,T$TEMP3,T        SAVE ONLY FOR OPEN
                                 5697 *
                                 5698 *
                      004772     5699         USE      CONST
                      004772     5700 TABLE   BSS      0
    004772  044000 000000        5701         ZERO     512*36,            512 ELEMENT SIZE
    004773  026400 000000        5702         ZERO     320*36,            320 ELEMENT SIZE
                      003432     5703         USE      PREVIOUS
                                 5704 *
                                 5705 *
                                 5706 *       OPEN FILE
                                 5707 *
                      003432     5708 INIT5   BSS      0
                      003432     5709         OPEN     (T$TEMP1,T),(T$TEMP2,T),(1,DU),(T$TEMP3,T),(0,DU)
    003432  000667 7000 00                    TSX      0,$OPEN
    003433  000027 0000 11                    ARG      T$TEMP1,T
```

JOB -- SET UP

```
 003434  000026 0000 11                       ARG     T$TEMP2,T
 003435  000001 0000 03                       ARG     1,DU
 003436  000025 0000 11                       ARG     T$TEMP3,T
END OF BINARY CARD IOS00110
 003437  000000 0000 03                       ARG     0,DU
                 003440    5710               CHECK   INIT6,B$BZ,INIT5,B$ITN,INT4,1                     1530
 003440  000000 7200 11                       LXL     0,T$SRW1,T
 003441  000077 3600 03                       ANX     0,B$STMK,DU
 003442  003450 6000 00                       TZE     INIT6
 003443  000003 1000 03                       CMPX    0,B$BZ,DU
 003444  003432 6000 00                       TZE     INIT5
 003445  000005 1000 03                       CMPX    0,B$ITN,DU
 003446  003406 6000 00                       TZE     INT4,1
 003447  777777 7100 00                       TRA     $ERROR
                           5711 *                                                                       1540
                           5712 *             FILE OPENED, FINISH UP                                    1550
                           5713 *                                                                       1560
                 003450    5714 INIT6         BSS     0                                                 1570
 003450  000000 2200 11    5715               LDX     0,T$SRW1,T       GET FRN OF FILE                  1580
 003451  000002 7400 16    5716               STX     0,J$FRN,J        SAVE IT                          1590
 003452  000001 2200 11    5717               LDX     0,T$SRW2,T       GET NUMBER OF UNITS              1600
 003453  000012 4500 16    5718               STZ     J$SIZE,J         CLEAR STORAGE AREA               1610
 003454  000012 4400 16    5719               SXL     0,J$SIZE,J       SAVE SIZE FOR LATER DECISIONS    1620
 003455  000006 2200 11    5720               LDX     0,T$NCB,T        GET BACK POINTER TO NCB          1630
 003456  000031 0540 10    5721               AOS     C$QFLOC,0        BUMP R/W PTR TO NEXT             1640
 003457  000032 0540 10    5722               AOS     C$BUSY,0         BUMP THE BUSY COUNT              1650
 003460  000011 2220 16    5723               LDX     X,J$BUF,J        GET PTR TO INPUT BUFFER          1660
 003461  000000 2350 07    5724               LDA     0,DL             BUILD MESSAGE WORD               1670
 003462  000001 7200 12    5725               LXL     0,BANR,X         GET BANNER BITS                  1680
 003463  003465 6010 00    5726               TNZ     *+2              NO BANNER                        1690
 003464  020000 2750 03    5727               ORA     B$HDRMK,DU       OR IN BANNER BIT                 1700
END OF BINARY CARD IOS00111
 003465  000002 2200 12    5728               LDX     0,TYPE,X         GET OUTPUT FORMAT                1710
 003466  003470 6000 00    5729               TZE     *+2              OK, 512 FORMAT                   1720
 003467  010000 2750 03    5730               ORA     B$OUTMK,DU       OR IN 320 FORMAT                 1730
 003470  000010 7550 16    5731               STA     J$MESS,J         MESSAGE BUILT EXCEPT FOR JOB NUMBER  1740
                 003471    5732               RELC    (J$BUF,J)        RELEASE THE BUFFER               1750
 003471  000011 2350 16                       LDA     J$BUF,J
 003472  001252 7070 00                       TSX     L,R$RELC
                           5733 *                                                                       1760
                           5734 *             FAKE THE RESTART ADDRESS                                  1770
                           5735 *                                                                       1780
 003473  003577 6200 00    5736               EAX     0,RUN            RESTART ADDRESS                  1790
 003474  000004 7400 11    5737               STX     0,T$TRA,T        PLACE IN TCB                     1800
                           5738 *                                                                       1810
                           5739 *             PLACE ON PROPER WAIT QUEUE                                1820
                           5740 *                                                                       1830
 003475  000004 6200 11    5741               EAX     0,Q$OFFST,T      GET OFFSET POINTER               1840
 003476  000006 2210 11    5742               LDX     T,T$NCB,T        GET BACK PTR TO NCB              1850
 003477  000034 2250 11    5743               LDX     Q,C$QUEUE,T      GET PTR TO QUEUE LIST            1860
```

JOB -- SET UP

```
    003500  005720 1010 03     5744          CMPX   T,C$LPE,DU       TEST FOR LINF PRINTER JOB          1870
    003501  003517 6010 00     5745          TNZ    INIT9            NO, NOT A LP JOB                   1880
    003502  000002 2230 12     5746          LDX    Y,TYPE,X         GET ITS TYPE (512/320 FORMAT)      1890
    003503  003507 6010 00     5747          TNZ    INIT7            SKIP IF 320                        1900
    003504  005670 2350 00     5748          LDA    R$S.512                                             1910
    003505  005672 2360 00     5749          LDQ    R$M.512                                             1920
    003506  003511 7100 00     5750          TRA    INIT8            DO COMPARE WITH THESE LIMITS       1930
    003507  005671 2350 00     5751 INIT7    LDA    R$S.320                                             1940
    003510  005673 2360 00     5752          LDQ    R$M.320                                             1950
    003511  000012 1110 16     5753 INIT8    CWL    J$SIZE,J         TEST WITH Q GETS HIM               1960
    003512  003517 6040 00     5754          TMI    INIT9            LARGE, ON Q$INP1                   1970
 END OF BINARY CARD IOS00112
    003513  003516 6010 00     5755          TNZ    *+3              SHORT?                             1980
    003514  005260 2250 03     5756          LDX    Q,Q$INP3,DU      MEDIUM                             1990
    003515  003517 7100 00     5757          TRA    INIT9            PLACE ON Q                         2000
    003516  005240 2250 03     5758          LDX    Q,Q$INP2,DU      SHORT                              2010
                    003517     5759 INIT9    BSS    0                                                   2020
    003517  000002 7170 15     5760          XED    Q$XADD,Q         PLACE ON QUEUE LIST                2030
                               5761 *                                                                   2040
                               5762 *        UNLOCK THE QUEUE FILE                                      2050
                               5763 *                                                                   2060
                    003520     5764 INT10    BSS    0                                                   2070
                    003520     5765          UNLCK  (C$QFRN,T)       UNLOCK THE FILE QUEUE              2080
    003520  000746 7000 00                   TSX    0,$UNLCK
    003521  000030 0000 11                   ARG    C$QFRN,T
                    003522     5766          CHECK  INT11,B$BZ,INT10                                    2090
    003522  000000 7200 11                   LXL    0,T$SRW1,T
    003523  000077 3600 03                   ANX    0,B$STMK,DU
    003524  003530 6000 00                   TZE    INT11
    003525  000003 1000 03                   CMPX   0,B$BZ,DU
    003526  003520 6000 00                   TZE    INT10
    003527  777777 7100 00                   TRA    $ERROR
                    003530     5767 INT11    BSS    0                                                   2100
    003530  003303 7100 00     5768          TRA    INIT0            SEE IF MORE                        2110
                               5769 *                                                                   2120
                               5770 *                                                                   2130
                               5771 *        EOF REACHED                                                2140
                               5772 *                                                                   2150
                    003531     5773 INITX    BSS    0                                                   2160
                    003531     5774          UNLCK  (J$QFRN,J)       UNLOCK THE QUEUE FILE              2170
    003531  000746 7000 00                   TSX    0,$UNLCK
    003532  000000 0000 16                   ARG    J$QFRN,J
                    003533     5775          CHECK  INTX1,B$BZ,INITX                                    2180
    003533  000000 7200 11                   LXL    0,T$SRW1,T
    003534  000077 3600 03                   ANX    0,B$STMK,DU
    003535  003541 6000 00                   TZE    INTX1
    003536  000003 1000 03                   CMPX   0,B$BZ,DU
    003537  003531 6000 00                   TZE    INITX
    003540  777777 7100 00                   TRA    $ERROR
                    003541     5776 INTX1    BSS    0                                                   2190
```

JOB -- SET UP

```
                    003541      5777          RELC     (J$BUF,J)          RELEASE THE BUFFER                    2200
  END OF BINARY CARD IOS00113
     003541  000011 2350 16                   LDA      J$BUF,J
     003542  001252 7070 00                   TSX      L,R$RELC
                                5778 *                                                                          2210
                                5779 *        WAKE UP THE SCHEDULER                                             2220
                                5780 *                                                                          2230
                    003543      5781          BRANCH   NOPASS,R$SKED      TRY TO START THE DUDES                2240
     003543  001467 7000 00                   TSX      0,T$GETT
     003544  000000 6220 11                   EAX      X,0,T
     003545  000005 2210 12                   LDX      T,T$LINK,X
     003546  000000 6210 12                   EAX      T,0,X
     003547  002565 6200 00                   EAX      0,R$SKED
     003550  000004 7400 11                   STX      0,T$TRA,T
     003551  000004 6200 11                   EAX      0,Q$OFFST,T
     003552  005162 7170 00                   XED      Q$XADD+Q$TASK
     003553  000005 2210 12                   LDX      T,T$LINK,X
                    003554                    BUGXR    (0,X)
                    525261           BUGBUG   SET      BUGBUG+1
     003554  525261 2200 03                   LDX      0,BUGBUG,DU
     003555  525261 2220 03                   LDX      X,BUGBUG,DU
                    003556      5782          RELJ                        RELEASE JOB                           2250
     003556  001511 7000 00                   TSX      0,J$RELJ
     003557  000000 6230 11     5783          EAX      Y,0,T              SAVE PTR TO CURRENT TCB               2260
     003560  000006 2210 11     5784          LDX      T,T$NCB,T          GET BACK NCB                          2270
                                5785 *                                                                          2280
                                5786 *        RE-ISSUE NOTIFY                                                   2290
                                5787 *                                                                          2300
                    003561      5788          BRANCH   PASS,C$NSRVX       SETUP TASK TO RE-ISSUE NOTIFY         2310
     003561  001467 7000 00                   TSX      0,T$GETT
     003562  000000 6220 11                   EAX      X,0,T
     003563  000005 2210 12                   LDX      T,T$LINK,X
     003564  003171 6200 00                   EAX      0,C$NSRVX
     003565  000004 7400 11                   STX      0,T$TRA,T
     003566  000004 6200 11                   EAX      0,Q$OFFST,T
  END OF BINARY CARD IOS00114
     003567  005162 7170 00                   XED      Q$XADD+Q$TASK
     003570  000000 6210 12                   EAX      T,0,X
                    003571                    BUGXR    (0,X)
                    525262           BUGBUG   SET      BUGBUG+1
     003571  525262 2200 03                   LDX      0,BUGBUG,DU
     003572  525262 2220 03                   LDX      X,BUGBUG,DU
                                5789 *                                                                          2320
                                5790 *        RELEASE REMAINING RESOURCE                                        2330
                                5791 *                                                                          2340
                    003573      5792          RELT                        RELEASE NEW TCB                       2350
     003573  001477 7000 00                   TSX      0,T$RELT
     003574  000000 6210 13     5793          EAX      T,0,Y              RESTORE TO OLD TCB                    2360
                    003575      5794          RELT                        RELEASE THE TRAP BLCOK                2370
     003575  001477 7000 00                   TSX      0,T$RELT
```

JOB -- SET UP

```
                         003576     5795        EXIT                          EVAPORATE                 2380
    003576  003074 7100 00                      TRA      $EXIT
                                    5796 *$*    DISK     RUN                                            2390
```

JOB -- RUN

```
                   003577    5798        USE     CODE                                          110
                             5799        HEAD                                                  120
                             5800 *                                                            130
                             5801 *                              RUN                           140
                             5802 *                                                            150
                             5803 *      THIS ROUTINE RUNS THE NEXT JOB                        160
                             5804 *      IT IS ENTERED ONLY AFTER A JOB HAS HAD ALL ITS RESOURCE 170
                             5805 *      REUQESTS SATISFIED (I.E. ALL RESOURCES ALLOCATED TO IT). 180
                             5806 *      ALL THAT NEED BE DONE IS PASS THE INFORMATION TO THE SUBMODULE. 190
                             5807 *      THAT MEANS ISSUING A CAUSE PASSING THE FRN OF THE INPUT FILE 200
                             5808 *      AND A MESSAGE WORD DESCRIBING WHAT TO DO WITH IT.  THE  210
                             5809 *      STATE TELLS WHAT PERIPHERAL TYPE IS TO BE THE OUPUT DEVICE TYPE. 220
                             5810 *      AFTER A SUCCESSFUL CAUSE, WE PUT OUT A NOTIFY ON THE JOB. 230
                             5811 *      STATE CORRESPONDS TO THE JOB NUMBER WHICH ARE UNIQUE.  WHEN 240
                             5812 *      CAUSED, WE GO TO JOB TERMINATION.                       250
                             5813 *                                                            260
                             5814 *      CALL WITH                                             270
                             5815 *              C(XT) = JOB-CONTROL-BLOCK ADDRESS             280
                             5816 *              ALL RESOURCES ALLOCATED                       290
                             5817 *                                                            300
                   003577    5818 RUN    BSS     0                                             310
                             5819 *                                                            320
                             5820 *      ISSUE CAUSE TO START JOB                              330
                             5821 *                                                            340
                   003577    5822        CAUSE   C$FRN2,(1,DU),(J$STATI,J),(J$MESS,J)          350
                   003577    5823        ETC     (J$FRN,J),(B$RD+B$LK,DU)                      360
 003577  000767 7000 00                  TSX     0,$CAUSE
 003600  006221 0000 00                  ARG     C$FRN2
 003601  000001 0000 03                  ARG     1,DU
 003602  000006 0000 16                  ARG     J$STATI,J
 003603  000010 0000 16                  ARG     J$MESS,J
 003604  000002 0000 16                  ARG     J$FRN,J
 003605  000021 0000 03                  ARG     B$RD+B$LK,DU
                   003606    5824        CHECK   RUN1,B$BZ,RUN                                 370
 003606  000000 7200 11                  LXL     0,T$SRW1,T
 003607  000077 3600 03                  ANX     0,B$STMK,DU
 003610  003614 6000 00                  TZE     RUN1
 003611  000003 1000 03                  CMPX    0,B$BZ,DU
 003612  003577 6000 00                  TZE     RUN
 003613  777777 7100 00                  TRA     $ERROR
                   003614    5825 RUN1   BSS     0                                             380
END OF BINARY CARD IOS00115
 003614  000000 2220 11      5826        LDX     X,T$SRW1,T      GET NUMBER OF PEOPLE CAUSED   390
 003615  003577 6000 00      5827        TZE     RUN             NO ONE, TRY AGAIN             400
                             5828 *                                                            410
                             5829 *      PUT OUT A NOTIFY ON THIS JOB                          420
                             5830 *                                                            430
 003616  003640 6200 00      5831        EAX     0,TERM          GET RESTART ADDRESS           440
 003617  000005 4400 11      5832        SXL     0,C$RLINK,T     SAVE IN TCB                   450
 003620  000007 2360 16      5833        LDQ     J$STATT,J       GET THE TERMINATE STATE       460
```

JOB -- RUN

```
003621  000025 7560 11      5834          STQ     T$TEMP3,T        SAVE FOR COMMUNICATIONS              470
003622  006220 2360 00      5835          LDQ     C$FRN1           GET ERN FOR NOTIFY                   480
003623  000024 7560 11      5836          STQ     T$TEMP4,T        SAVE FOR COMMUNICATIONS              490
               003624       5837          BRANCH  PASS,C$NSRVX     PUT OUT A NOTIFY                     500
003624  001467 7000 00                    TSX     0,T$GETT
003625  000000 6220 11                    EAX     X,0,T
003626  000005 2210 12                    LDX     T,T$LINK,X
003627  003171 6200 00                    EAX     0,C$NSRVX
003630  000004 7400 11                    STX     0,T$TRA,T
003631  000004 6200 11                    EAX     0,Q$OFFST,T
003632  005162 7170 00                    XED     Q$XADD+Q$TASK
003633  000000 6210 12                    EAX     T,0,X
               003634                     BUGXR   (0,X)
               525263           BUGBUG    SET     BUGBUG+1
003634  525263 2200 03                    LDX     0,BUGBUG,DU
003635  525263 2220 03                    LDX     X,BUGBUG,DU
                            5838 *                                                                  510
                            5839 *        FINISH UP                                                 520
                            5840 *                                                                  530
               003636       5841          RELT                     RELEASE TRAP BLOCK                   540
003636  001477 7000 00                    TSX     0,T$RELT
               003637       5842          EXIT                                                          550
003637  003074 7100 00                    TRA     $EXIT
                            5843 *$*      DISK    TERM1                                                 560
```

JOB -- TERMINATION

```
                        003640   5845        USE    CODE                                        110
                                 5846        HEAD                                               120
                                 5847 *                                                         130
                                 5848 *                                                         140
                                 5849 *                             TERMINATION                 150
                                 5850 *                                                         160
                                 5851 *      JOB TERMINATION DOES THE FOLLOWING:                170
                                 5852 *             DOES ACCOUNTING REPORT                      180
                                 5853 *             ALLOCATES A WORKING BUFFER                  190
                                 5854 *             ACTS ON DISPOSITON CODE                     200
                                 5855 *             MARKS INPUT FILE DESCRIPTOR AS DONE         210
                                 5856 *             TRIES TO SCRATCH THE INPUT FILE QUEUE       220
                                 5857 *             RELEASES JOBS RESOURCES                     230
                                 5858 *             AWAKENS SCHEDULER                           240
                                 5859 *                                                         250
                                 5860 *                                                         260
                        003640   5861 TERM   BSS    0                                           270
 003640  000001 2360 11          5862        LDQ    T$SRW2,T           GET MESSAGE FROM SUBMODULE   280
END OF BINARY CARD IOS00116
 003641  000012 7720 00          5863        QRL    18-4-4             RIGHT JUSTIFY IN QU      290
 003642  000017 3760 03          5864        ANQ    =017,DU            MASK TO COMMAND          300
 003643  000010 1160 03          5865        CMPQ   CMAX,DU            TEST VALIDITY            310
 003644  777777 6030 00          5866        TRC    $ERROR             ***PROBLEM               320
 003645  003646 7100 22          5867        TRA    *+1,QU*            BRANCH ON COMMAND        330
 003646  777777 0000 00          5868 CMDTB  ARG    $ERROR             0 = ILLEGAL              340
 003647  777777 0000 00          5869        ARG    $ERROR             1 = GET (ILLEGAL)        350
 003650  003670 0000 00          5870        ARG    TERM1              2 = KILL                 360
 003651  777777 0000 00          5871        ARG    $ERROR             3 = RELEASE (ILLEGAL)    370
 003652  777777 0000 00          5872        ARG    $ERROR             4 = XXXX (ILLEGAL)       380
 003653  777777 0000 00          5873        ARG    $ERROR             5 = RESTART (ILLEGAL)    390
 003654  003670 0000 00          5874        ARG    TERM1              6 = DONE                 400
 003655  777777 0000 00          5875        ARG    $ERROR             7 = READY (ILLEGAL)      410
                        000010   5876 CMAX   EQU    *-CMDTB            NUMBER OF COMMANDS       420
                                 5877 *                                                         430
                                 5878 *      ACCOUNTING REPORT                                  440
                                 5879 *                                                         450
                        003656   5880 TERM0  BSS    0                                           460
                        003656   5881        ACCT   (J$ACODE,J),(J$SIZE,J),(1,DU)  WRITE BILLING INFO   470
 003656  001017 7000 00                      TSX    0,$ACCT
 003657  000004 0000 16                      ARG    J$ACODE,J
 003660  000012 0000 16                      ARG    J$SIZE,J
 003661  000001 0000 03                      ARG    1,DU
                        003662   5882        CHECK  TERM1,B$BZ,TERM0                            480
 003662  000000 7200 11                      LXL    0,T$SRW1,T
 003663  000077 3600 03                      ANX    0,B$STMK,DU
 003664  003670 6000 00                      TZE    TERM1
 003665  000003 1000 03                      CMPX   0,B$BZ,DU
 003666  003656 6000 00                      TZE    TERM0
END OF BINARY CARD IOS00117
 003667  777777 7100 00                      TRA    $ERROR
```

JOB -- TERMINATION

```
                         003670     5883 TERM1   BSS     0
                                    5884 *
                                    5885 *       ALLOCATE A WORKING BUFFER
                                    5886 *
                         003670     5887 TRM1    BSS     0
                         003670     5888         GETC    (QBFSZ,DL)          GET A WORKING BUFFER
         003670  000100 2350 07                  LDA     QBFSZ,DL
         003671  001152 7070 00                  TSX     L,R$GETC
         003672  000011 7550 16     5889         STA     J$BUF,J             SAVE BUFFER POINTER
         003673  000003 7200 16     5890         LXL     0,J$DISP,J          GET DISPOSITION RULE
         003674  000003 3600 11     5891         ANX     0,DISMK,T           ISOLATE DISPOSITION RULE
         003675  003676 7100 30     5892         TRA     *+1,0*              FOLLOW THE RULES
         003676  003701 0000 00     5893         ARG     TRMD                DESTROY
         003677  003733 0000 00     5894         ARG     TRMS                SCRATCH
         003700  003744 0000 00     5895         ARG     TRMC                CLOSE
                                    5896 *
                                    5897 *       DESTROY SOURCE FILE
                                    5898 *
                         003701     5899 TRMD    BSS     0
                         003701     5900         RRF     (J$QFRN,J),(J$QFLOC,J),(J$BUF,J),(1,DU)
         003701  000564 7000 00                  TSX     0,$RRF
         003702  000000 0000 16                  ARG     J$QFRN,J
         003703  000001 0000 16                  ARG     J$QFLOC,J
         003704  000011 0000 16                  ARG     J$BUF,J
         003705  000001 0000 03                  ARG     1,DU
                         003706     5901         CHECK   TRMD1,B$BZ,TRMD
         003706  000000 7200 11                  LXL     0,T$SRW1,T
         003707  000077 3600 03                  ANX     0,B$STMK,DU
         003710  003714 6000 00                  TZE     TRMD1
         003711  000003 1000 03                  CMPX    0,B$BZ,DU
         003712  003701 6000 00                  TZE     TRMD
         003713  777777 7100 00                  TRA     $ERROR
         003714  000011 2220 16     5902 TRMD1   LDX     2,J$BUF,J           GET POINTER TO BUFFER AREA
   END OF BINARY CARD IOS00118
         003715  000004 6350 12     5903         EAA     TN,2                GET PTR TO TREE NAME
         003716  000027 7550 11     5904         STA     T$TEMP1,T           SAVE FOR DESTROY
                         003717     5905         DESTRO  (T$TEMP1,T),(TNSZ,2),(1,DU)
         003717  000714 7000 00                  TSX     0,$DESTRO
         003720  000027 0000 11                  ARG     T$TEMP1,T
         003721  000001 0000 12                  ARG     TNSZ,2
         003722  000001 0000 03                  ARG     1,DU
                         003723     5906         CHECK   TRMS,B$BZ,TRMD1,B$ITN,TRMC
         003723  000000 7200 11                  LXL     0,T$SRW1,T
         003724  000077 3600 03                  ANX     0,B$STMK,DU
         003725  003733 6000 00                  TZE     TRMS
         003726  000003 1000 03                  CMPX    0,B$BZ,DU
         003727  003714 6000 00                  TZE     TRMD1
         003730  000005 1000 03                  CMPX    0,B$ITN,DU
         003731  003744 6000 00                  TZE     TRMC
         003732  777777 7100 00                  TRA     $ERROR
```

JOB -- TERMINATION

```
                                  5907 *                                                           730
                                  5908 *        SCRATCH THE SOURCE FILE                            740
                                  5909 *                                                           750
                        003733    5910 TRMS     BSS     0                                          760
                        003733    5911          SCR     (J$FRN,J),(0,DU) ,SCRATCH THE FILE         770
   003733  000612 7000 00                       TSX     0,$SCR
   003734  000002 0000 16                       ARG     J$FRN,J
   003735  000000 0000 03                       ARG     0,DU
                        003736    5912          CHECK   TRMC,B$BZ,TRMS                             780
   003736  000000 7200 11                       LXL     0,T$SRW1,T
   003737  000077 3600 03                       ANX     0,B$STMK,DU
   003740  003744 6000 00                       TZE     TRMC
   003741  000003 1000 03                       CMPX    0,B$BZ,DU
   003742  003733 6000 00                       TZE     TRMS
 END OF BINARY CARD IOS00119
   003743  777777 7100 00                       TRA     $ERROR
                                  5913 *                                                           790
                                  5914 *        CLOSE SOURCE FILE                                  800
                                  5915 *                                                           810
                        003744    5916 TRMC     BSS     0                                          820
                        003744    5917          CLOSE   (J$FRN,J)                                  830
   003744  000704 7000 00                       TSX     0,$CLOSE
   003745  000002 0000 16                       ARG     J$FRN,J
                        003746    5918          CHECK   TRM2,B$BZ,TRMC                             840
   003746  000000 7200 11                       LXL     0,T$SRW1,T
   003747  000077 3600 03                       ANX     0,B$STMK,DU
   003750  003754 6000 00                       TZE     TRM2
   003751  000003 1000 03                       CMPX    0,B$BZ,DU
   003752  003744 6000 00                       TZE     TRMC
   003753  777777 7100 00                       TRA     $ERROR
                                  5919 *                                                           850
                                  5920 *        MARK SOURCE DONE                                   860
                                  5921 *                                                           870
                        003754    5922 TRM2     BSS     0                                          880
   003754  000011 2220 16         5923          LDX     X,J$BUF,J          POINT TO BUFFER         890
   003755  000001 0220 03         5924          ADLX    X,1,DU             BUMP ONE                900
   003756  000001 3360 07         5925          LCQ     1,DL               MARK BUFFER DONE        910
   003757  777777 7560 12         5926          STQ     -1,X               WITH CHECKSUM = -1      920
   003760  176300 5202 01         5927          RPT     QBFSZ-1,1,TZE      AND WITH THE REST       930
   003761  000000 4500 12         5928          STZ     0,X                OF THE BLOCK ZERO       940
                        003762    5929 TRM3     BSS     0                                          950
                        003762    5930          WRF     (J$QFRN,J),(J$QFLOC,J),(J$BUF,J),(1,DU)    960
   003762  000577 7000 00                       TSX     0,$WRF
   003763  000000 0000 16                       ARG     J$QFRN,J
   003764  000001 0000 16                       ARG     J$QFLOC,J
   003765  000011 0000 16                       ARG     J$BUF,J
   003766  000001 0000 03                       ARG     1,DU
                        003767    5931          CHECK   TRM4,B$BZ,TRM3                             970
   003767  000000 7200 11                       LXL     0,T$SRW1,T
   003770  000077 3600 03                       ANX     0,B$STMK,DU
```

JOB -- TERMINATION

```
END OF BINARY CARD IOS00120
    003771  003775 6000 00                      TZE     TRM4
    003772  000003 1000 03                      CMPX    0.B$BZ.DU
    003773  003762 6000 00                      TZE     TRM3
    003774  777777 7100 00                      TRA     $ERROR
                    003775              TRM4    BSS     0
                                        *
                                        *       CLEAN UP
                                        *
    003775  000005 2220 16                      LDX     X.J$NCB.J         GET PTR TO NCB
                    003776                      DECRM   (C$BUSY.X)        DECREMENT THE BUSY COUNT
    003776  000001 3360 07                      LCQ     1.DL
    003777  000032 0560 12                      ASQ     C$BUSY.X
    004000  777777 6040 00                      TMI     $ERROR            ***PROBLEM
    004001  004024 6010 00                      TNZ     TRM5              SKIP IN NON-ZERO
                                        *
                                        *       SETUP TASK TO SCRATCH INPUT Q FILE
                                        *
    004002  000030 2200 12                      LDX     0.C$QFRN.X        GET INPUT QUEUE FRN
    004003  000027 7400 11                      STX     0.T$TEMP1.T       SAVE FOR PASS
    004004  000026 7420 11                      STX     X.T$TEMP2.T       PASS PTR TO NCB
                    004005                      BRANCH  NOPASS.R$SCR.(T$TEMP1.T).(T$TEMP2.T)  SCRATCH INPUT QUEUE FILE
    004005  001467 7000 00                      TSX     0.T$GETT
    004006  000000 6220 11                      EAX     X.0.T
    004007  000005 2210 12                      LDX     T.T$LINK.X
    004010  000027 2360 11                      LDQ     T$TEMP1.T
    004011  000027 7560 12                      STQ     T$TEMP1.X
    004012  000026 2360 11                      LDQ     T$TEMP2.T
    004013  000026 7560 12                      STQ     T$TEMP2.X
    004014  000000 6210 12                      EAX     T.0.X
    004015  002765 6200 00                      EAX     0.R$SCR
    004016  000004 7400 11                      STX     0.T$TRA.T
END OF BINARY CARD IOS00121
    004017  000004 6200 11                      EAX     0.Q$OFFST.T
    004020  005162 7170 00                      XED     Q$XADD+Q$TASK
    004021  000005 2210 12                      LDX     T.T$LINK.X
                    004022                      BUGXR   (0.X)
                    525264              BUGBUG  SET     BUGBUG+1
    004022  525264 2200 03                      LDX     0.BUGBUG.DU
    004023  525264 2220 03                      LDX     X.BUGBUG.DU
                                        *
                                        *       RELEASE RESOURCES
                                        *
                    004024              TRM5    BSS     0
    004024  000013 6200 16                      EAX     0.J$RES.J         GET PTR TO RESOURCES
    004025  000026 7400 11              TRM6    STX     0.T$TEMP2.T       SAVE PTR
    004026  000000 2350 10                      LDA     0.0               GET NEXT RESOURCE TO RELEASE
    004027  004050 6040 00                      TMI     TRM8              TEST FOR DONE
    004030  777777 3750 03                      ANA     -1.DU             MASK TO TYPE
    004031  005570 1150 03                      CMPA    R$LPTAB.DU        TEST FOR A LINE PRINTER
```

JOB -- TERMINATION

```
    004032  004043 6010 00      5957         TNZ     TRM7              IF NOT, SKIP                    1230
    004033  000012 2360 16      5958         LDQ     J$SIZE,J          GET SIZE                        1240
    004034  000003 2220 16      5959         LDX     X,J$TYPE,J        AND TYPE                        1250
    004035  004774 7160 12      5960         XEC     LPTB1,X           TEST FOR LONG JOB               1260
                   004774       5961         USE     CONST                                             1270
    004774  005672 1160 00      5962 LPTB1   CMPQ    R$M,512           LONG 512?                       1280
    004775  005673 1160 00      5963         CMPQ    R$M,320           LONG 320?                       1290
                   004036       5964         USE     PREVIOUS                                          1300
    004036  004043 6000 00      5965         TZE     TRM7              IF NOT, SKIP                    1310
    004037  004043 6020 00      5966         TNC     TRM7              IF NOT, SKIP                    1320
                   004040       5967         DECRM   R$LJOBS           YES, SO DECREMENT COUNTER       1330
    004040  000001 3360 07                   LCQ     1,DL
 END OF BINARY CARD IOS00122
    004041  005674 0560 00                   ASQ     R$LJOBS
    004042  777777 6040 00      5968         TMI     $ERROR            ***PROBLEM                      1340
                   004043       5969 TRM7    RELP    (0,0)             RELEASE THE PERIPERAL           1350
    004043  000000 2350 10                   LDA     0,0
    004044  002447 7070 00                   TSX     L,R$RELP
    004045  000001 2200 03      5970         LDX     0,1,DU            ADD ONE                         1360
    004046  000026 0200 11      5971         ADLX    0,T$TEMP2,T       TO BUMP TO NEXT RESOURCE        1370
    004047  004025 7100 00      5972         TRA     TRM6              LOOP                            1380
                   004050       5973 TRM8    BSS     0                                                 1390
                   004050       5974         RELC    (J$BUF,J)         RELEASE BUFFER                  1400
    004050  000011 2350 16                   LDA     J$BUF,J
    004051  001252 7070 00                   TSX     L,R$RELC
                                5975 *                                                                 1410
                                5976 *       AWAKEN THE SCHEDULER                                      1420
                                5977 *                                                                 1430
                   004052       5978         BRANCH  NOPASS,R$SKED     AWAKEN SCHEDULER                1440
    004052  001467 7000 00                   TSX     0,T$GETT
    004053  000000 6220 11                   EAX     X,0,T
    004054  000005 2210 12                   LDX     T,T$LINK,X
    004055  000000 6210 12                   EAX     T,0,X
    004056  002565 6200 00                   EAX     0,R$SKED
    004057  000004 7400 11                   STX     0,T$TRA,T
    004060  000004 6200 11                   EAX     0,Q$OFFST,T
    004061  005162 7170 00                   XED     Q$XADD+Q$TASK
    004062  000005 2210 12                   LDX     T,T$LINK,X
                   004063                    BUGXR   (0,X)
                   525265            BUGBUG  SET     BUGBUG+1
    004063  525265 2200 03                   LDX     0,BUGBUG,DU
    004064  525265 2220 03                   LDX     X,BUGBUG,DU
    004065  000007 7220 16      5979         LXL     X,J$JOB,J         RELEASE JOB NUMBER              1450
    004066  005700 2340 12      5980         SZN     J$JTAB,X          CHECK FOR GOOD BOOKKEEPING      1460
 END OF BINARY CARD IOS00123
    004067  777777 6000 00      5981         TZE     $ERROR            ***PROBLEM                      1470
    004070  005700 4500 12      5982         STZ     J$JTAB,X          DEALLOCATE NUMBER               1480
                   004071       5983         RELJ                      RELEASE JCB                     1490
    004071  001511 7000 00                   TSX     0,J$RELJ
                   004072       5984         RELT                      RELEASE TRAP BLOCK              1500
```

JOB -- TERMINATION

```
004072  001477 7000 00                    TSX     0,T$RELT
                      004073    5985      EXIT                       POOF1          1510
004073  003074 7100 00                    TRA     $EXIT
                                5986 *$*  DISK    CRASH                             1520
```

SUB-MODULE ABNORMAL TERMINATION -- CRASH

```
                 004074     5988       USE     CODE                                                  110
                            5989       HEAD                                                          120
                            5990 *                                                                   130
                            5991 *                                                                   140
                            5992 *                             CRASH                                 150
                            5993 *                                                                   160
                            5994 *     THIS ROUTINE IS ENTERED WHENEVER ANY SUB-MODULE TERMINATES.   170
                            5995 *     THAT IS TO SAY, THE NOTIFY ON ITS PROCESSS EVENT IS CAUSED.   180
                            5996 *     IT LOGS A MESSAGE TO THE OPERATOR POINTING THE FINGER AT      190
                            5997 *     THE CULPRIT AND THEN DOES A SOFT CRASH.  THE 'LISTENER' WILL  200
                            5998 *     DO AN INSTANT RESTART AND THEN WE'RE BACK IN THE OLE BALL GAME. 210
                            5999 *                                                                   220
                            6000 *     ENTER WITH                                                    230
                            6001 *             C(XT) = NCB                                           240
                            6002 *                                                                   250
                 004074     6003 CRASH BSS     0                                                     260
                            6004 *                                                                   270
                            6005 *     TELL OPER THE NEWS                                            280
                            6006 *                                                                   290
                 004074     6007       LOGS                    TELL THE OP THE BAD NEWS              300
 004074  001517 7070 00                TSX     L,O$LOGS
                 004075     6008       LOGC    CRMS            THAT A SUB-MODULE CRASHED             310
 004075  004776 2350 00                LDA     CRMS
 004076  001543 7070 00                TSX     L,O$LOGC
                 004776     6009       USE     CONST                                                 320
 004776  004777 0031 40     6010 CRMS  TALLYB  *+1,24+1,0      'PERIPHERAL CRASHED: '                330
 004777  177177015012       6011       OCT     177177015012                                          340
 005000  120105122111       6012       UASCI   5,PERIPHERAL CRASHED:                                 350
                 004077     6013       USE     PREVIOUS                                              360
 004077  000007 6350 11     6014       EAA     C$ABBR,T        AND WHICH ONE IT WAS                  370
 004100  000540 2750 07     6015       ORA     4*$TAL+$TAL+$TALYB,DL                                 380
                 004101     6016       LOGC    A                                                     390
 004101  001543 7070 00                TSX     L,O$LOGC
                 004102     6017       LOGC    BYEMS           SAY SO LONG,FARE-THEE-WELL,AU REVOIR  400
 004102  005005 2350 00                LDA     BYEMS
END OF BINARY CARD IOS00124
 004103  001543 7070 00                TSX     L,O$LOGC
                 005005     6018       USE     CONST                                                 410
 005005  005006 0024 40     6019 BYEMS TALLYB  *+1,19+1,0      'IOS*OVER & OUT!                      420
 005006  007007015012       6020       OCT     007007015012                                          430
 005007  111117123052       6021       UASCI   4,IOS*OVER & OUT!                                     440
                 004104     6022       USE     PREVIOUS                                              450
                 004104     6023       LOGX                    SEND IT                               460
 004104  001560 7070 00                TSX     L,O$LOGX
                            6024 *                                                                   470
                            6025 *     DO A SOFT-CRASH                                               480
                            6026 *                                                                   490
 004105  000470 7100 00     6027       TRA     X$TERM          BYE-BYE TOOTS                         500
                            6028 *$*   DISK    ANIT                                                  510
```

```
                      004106      6030        USE     CODE                                                     110
                                  6031        HEAD                                                             120
                                  6032 *                                                                       130
                                  6033 *                                                                       140
                                  6034 *                                  INITIALIZATION                       150
                                  6035 *                                                                       160
                                  6036 *      THIS ROUTINE INITIALIZES THE PERIPHERAL SCHEDULER                170
                                  6037 *      MONITOR BY PERFORMING THE FOLLOWING FUNCTIONS.                   180
                                  6038 *                                                                       190
                                  6039 *      FUNCTIONS                                                        200
                                  6040 *              INITIALIZE REGISTERS AND LOCATION ZERO                   210
                                  6041 *              SET FAULT VECTOR                                         220
                                  6042 *              OPEN VARIOUS SYSTEM FILES                                230
                                  6043 *              OPEN PERIPHERALS                                         240
                                  6044 *              OPEN COM FILES                                           250
                                  6045 *              SPAWN SUBMODULES                                         260
                                  6046 *              SET UP NOTIFIES                                          270
                                  6047 *                                                                       280
                                  6048 *      USE     STORE           PUT THIS CODE IN THE STORAGE AREA FUDGE $MTOPO
                      004106      6049 NIT    BSS     0               INITIALIZATION ENTRY                     300
                      004106      6050 UP     EQU     NIT                                                      310
                                  6051 *                                                                       320
                                  6052 *      CATCH WILD TRANSFERS TO ZERO                                     330
                                  6053 *                                                                       340
 004106  000000 4500 00           6054        STZ     0               BURN OUR BRIDGES BEHIND US               350
                      004107      6055        CKPT                    SAVE REGISTERS                           360
 004107  000474 7170 00                       XED     X$CKPT
 004110  006350 2210 03           6056        LDX     T,SPTCB,DU      INITIALIZE T                             370
 004111  006350 7260 07           6057        LXL     J,SPTCB,DL      AND J TO THE SAME                        380
 004112  000006 4460 11           6058        SXL     J,T$JCB,T       SAVE FOR $EXIT                           390
 004113  004400 6340 07           6059        LDI     B$OVM+B$PAM,DL  MASK OFF OVERFLOWAND PARITY ERRORS       400
 004114  000400 2350 03           6060        LDA     ZZ1,DU          INITIALIZE TO AVAILABLE MEMORY           410
 004115  005300 7550 00           6061        STA     $AVAIL                                                   420
 004116  005700 5540 00           6062        STC1    J$JTAB          MAKE JOB NUMBER 0 ILLEGAL                430
                                  6063 *                                                                       440
                                  6064 *      SET FAULT VECTOR                                                 450
                                  6065 *                                                                       460
                      004117      6066 NIT1   SETFV   (X$FV,DU)       SET THE FAULT VECTOR                     470
 004117  000526 7000 00                       TSX     0,$SETFV
END OF BINARY CARD IOS00125
 004120  000000 0000 03                       ARG     X$FV,DU
                      004121      6067        CHECK   NIT2,B$BZ,NIT1                                           480
 004121  000000 7200 11                       LXL     0,T$SRW1,T
 004122  000077 3600 03                       ANX     0,B$STMK,DU
 004123  004127 6000 00                       TZE     NIT2
 004124  000003 1000 03                       CMPX    0,B$BZ,DU
 004125  004117 6000 00                       TZE     NIT1
 004126  777777 7100 00                       TRA     $ERROR
                                  6068 *$*    DISK    ANIT2                                                    490
```

INITIALIZATION -- OPEN SYSTEM FILES

```
                                 6070 *                                                     110
                                 6071 *                                                     120
                                 6072 *                          OPEN SYSTEM FILES          130
                                 6073 *                                                     140
                                 6074 *                                                     150
                                 6075 *         OPEN 'PRINT-FILE-QUEUE' FILE                160
                                 6076 *                                                     170
                     004127      6077 NIT2      BSS     0                                   180
                     004127      6078           OPEN    (LPQ,DU),(LPQTS,DU),(1,DU),(LPQES,DU),(0,DU)   190
    004127  000667 7000 00                      TSX     0,$OPEN
    004130  005013 0000 03                      ARG     LPQ,DU
    004131  000014 0000 03                      ARG     LPQTS,DU
    004132  000001 0000 03                      ARG     1,DU
    004133  004400 0000 03                      ARG     LPQES,DU
    004134  000000 0000 03                      ARG     0,DU
                     004135      6079           CHECK   NIT3,B$BZ,NIT2                      200
    004135  000000 7200 11                      LXL     0,T$SRW1,T
    004136  000077 3600 03                      ANX     0,B$STMK,DU
    004137  004143 6000 00                      TZE     NIT3
    004140  000003 1000 03                      CMPX    0,B$BZ,DU
    004141  004127 6000 00                      TZE     NIT2
    004142  777777 7100 00                      TRA     $ERROR
                     004143      6080 NIT3      BSS     0                                   210
    004143  000000 2200 11       6081           LDX     0,T$SRW1,T      GET FRN             220
    004144  005750 7400 00       6082           STX     0,C$FRLPQ       SAVE IT TABLE       230
                                 6083 *                                                     240
                                 6084 *         OPEN 'PUNCH-FILE-QUEUE' FILE                250
                                 6085 *                                                     260
                     004145      6086 NIT4      BSS     0                                   270
                     004145      6087           OPEN    (CPQ,DU),(CPQTS,DU),(1,DU),(CPQES,DU),(0,DU)   280
    004145  000667 7000 00                      TSX     0,$OPEN
 END OF BINARY CARD IOS00126
    004146  005027 0000 03                      ARG     CPQ,DU
    004147  000014 0000 03                      ARG     CPQTS,DU
    004150  000001 0000 03                      ARG     1,DU
    004151  004400 0000 03                      ARG     CPQES,DU
    004152  000000 0000 03                      ARG     0,DU
                     004153      6088           CHECK   NIT4.1,B$BZ,NIT4                    290
    004153  000000 7200 11                      LXL     0,T$SRW1,T
    004154  000077 3600 03                      ANX     0,B$STMK,DU
    004155  004161 6000 00                      TZE     NIT4.1
    004156  000003 1000 03                      CMPX    0,B$BZ,DU
    004157  004145 6000 00                      TZE     NIT4
    004160  777777 7100 00                      TRA     $ERROR
                     004161      6089 NIT4.1    BSS     0                                   300
    004161  000000 2200 11       6090           LDX     0,T$SRW1,T      GET FRN             310
    004162  006010 7400 00       6091           STX     0,C$FRCPQ       SAVE IN TABLE       320
```

INITIALIZATION -- OPEN SYSTEM EVENTS

```
                              6093 *                                                                340
                              6094 *                                                                350
                              6095 *                                   OPEN SYSTEM EVENTS           360
                              6096 *                                                                370
                              6097 *                                                                380
                              6098 *        OPEN 'PRINT-FILE-QUEUE' EVENT                           390
                              6099 *                                                                400
                      004163  6100 NIT5     BSS     0                                               410
                      004163  6101          OPEN    (LPE.DU).(LPETS.DU).(1.DU).(LPEES.DU).(0.DU)    420
  004163  000667 7000 00                    TSX     0.$OPEN
  004164  005043 0000 03                    ARG     LPE.DU
  004165  000014 0000 03                    ARG     LPETS.DU
  004166  000001 0000 03                    ARG     1.DU
  004167  002200 0000 03                    ARG     LPEES.DU
  004170  000000 0000 03                    ARG     0.DU
                      004171  6102          CHECK   NIT6.B$BZ.NIT5                                  430
  004171  000000 7200 11                    LXL     0.T$SRW1.T
  004172  000077 3600 03                    ANX     0.B$STMK.DU
  004173  004177 6000 00                    TZE     NIT6
END OF BINARY CARD IOS00127
  004174  000003 1000 03                    CMPX    0.B$BZ.DU
  004175  004163 6000 00                    TZE     NIT5
  004176  777777 7100 00                    TRA     $ERROR
                      004177  6103 NIT6     BSS     0                                               440
  004177  000000 2200 11      6104          LDX     0.T$SRW1.T       GET FRN                        450
  004200  005744 7400 00      6105          STX     0.C$FRLPE        SAVE IN TABLE                  460
                              6106 *                                                                470
                              6107 *        OPEN 'PUNCH-FILE-QUEUE' EVENT                           480
                              6108 *                                                                490
                      004201  6109 NIT7     BSS     0                                               500
                      004201  6110          OPEN    (CPE.DU).(CPETS.DU).(1.DU).(CPEES.DU).(0.DU)    510
  004201  000667 7000 00                    TSX     0.$OPEN
  004202  005057 0000 03                    ARG     CPE.DU
  004203  000014 0000 03                    ARG     CPETS.DU
  004204  000001 0000 03                    ARG     1.DU
  004205  002200 0000 03                    ARG     CPEES.DU
  004206  000000 0000 03                    ARG     0.DU
                      004207  6111          CHECK   NIT8.B$BZ.NIT7                                  520
  004207  000000 7200 11                    LXL     0.T$SRW1.T
  004210  000077 3600 03                    ANX     0.B$STMK.DU
  004211  004215 6000 00                    TZE     NIT8
  004212  000003 1000 03                    CMPX    0.B$BZ.DU
  004213  004201 6000 00                    TZE     NIT7
  004214  777777 7100 00                    TRA     $ERROR
                      004215  6112 NIT8     BSS     0                                               530
  004215  000000 2200 11      6113          LDX     0.T$SRW1.T       GET FRN                        540
  004216  006004 7400 00      6114          STX     0.C$FRCPE        SAVE IN TABLE                  550
```

INITIALIZATION -- OPEN SYSTEM EVENTS

```
                          6116 *                                                        570
                          6117 *                                                        580
                          6118 *        OPEN ,JOB-STREAM-SCHEDULER, EVENT               590
                          6119 *                                                        600
                 004217   6120 NIT9     BSS     0                                       610
                 004217   6121          OPEN    (JSS,DU),(JSSTS,DU),(1,DU),(JSSES,DU),(0,DU)   620
004217  000667 7000 00                  TSX     0,$OPEN
004220  005073 0000 03                  ARG     JSS,DU
004221  000014 0000 03                  ARG     JSSTS,DU
END OF BINARY CARD IOS00128
004222  000001 0000 03                  ARG     1,DU
004223  002200 0000 03                  ARG     JSSES,DU
004224  000000 0000 03                  ARG     0,DU
                 004225   6122          CHECK   NIT10,B$BZ,NIT9                         630
004225  000000 7200 11                  LXL     0,T$SRW1,T
004226  000077 3600 03                  ANX     0,B$STMK,DU
004227  004233 6000 00                  TZE     NIT10
004230  000003 1000 03                  CMPX    0,B$BZ,DU
004231  004217 6000 00                  TZE     NIT9
004232  777777 7100 00                  TRA     $ERROR
                 004233   6123 NIT10    BSS     0                                       640
004233  000000 2200 11    6124          LDX     0,T$SRW1,T     GET FRN                  650
004234  006044 7400 00    6125          STX     0,C$FRJSS      SAVE IN TABLE            660
```

INITIALIZATION -- OPEN SYSTEM SUB-MODULES

```
                              6127 *                                                          680
                              6128 *                                                          690
                              6129 *                        OPEN SYSTEM SUB-MODULES           700
                              6130 *                                                          710
                              6131 *     OPEN *LINE PRINTER/CARD PUNCH* MODULE                720
                              6132 *                                                          730
                     004235   6133 NIT11  BSS    0                                            740
                     004235   6134        OPEN   (PP,DU),(PPTS,DU),(1,DU),(PPES,DU),(0,DU)    750
   004235  000667 7000 00                 TSX    0,$OPEN
   004236  005107 0000 03                 ARG    PP,DU
   004237  000014 0000 03                 ARG    PPTS,DU
   004240  000001 0000 03                 ARG    1,DU
   004241  002200 0000 03                 ARG    PPES,DU
   004242  000000 0000 03                 ARG    0,DU
                     004243   6135        CHECK  NIT12,B$BZ,NIT11                             760
   004243  000000 7200 11                 LXL    0,T$SRW1,T
   004244  000077 3600 03                 ANX    0,B$STMK,DU
   004245  004251 6000 00                 TZE    NIT12
   004246  000003 1000 03                 CMPX   0,B$BZ,DU
   004247  004235 6000 00                 TZE    NIT11
 END OF BINARY CARD IOS00129
   004250  777777 7100 00                 TRA    $ERROR
                     004251   6136 NIT12  BSS    0                                            770
   004251  000000 2200 11     6137        LDX    0,T$SRW1,T     GET FRN                       780
   004252  006260 7400 00     6138        STX    0,C$FRPP       SAVE IN TABLE                 790
   004253  004272 7100 00     6139        TRA    NIT15          **********                    800
```

INITIALIZATION -- OPEN SYSTEM SUB-MODULES

```
                              6141 *                                                              820
                              6142 *                                                              830
                              6143 *        OPEN ,CARD READER, MODULE                             840
                              6144 *                                                              850
                     004254   6145 NIT13  BSS     0                                               860
                     004254   6146        OPEN    (CR,DU),(CRTS,DU),(1,DU),(CRES,DU),(0,DU)       870
004254  000667 7000 00                    TSX     0,$OPEN
004255  005123 0000 03                    ARG     CR,DU
004256  000014 0000 03                    ARG     CRTS,DU
004257  000001 0000 03                    ARG     1,DU
004260  002200 0000 03                    ARG     CRES,DU
004261  000000 0000 03                    ARG     0,DU
                     004262   6147        CHECK   NIT14,BSBZ,NIT13                                880
004262  000000 7200 11                    LXL     0,T$SRW1,T
004263  000077 3600 03                    ANX     0,B$STMK,DU
004264  004270 6000 00                    TZE     NIT14
004265  000003 1000 03                    CMPX    0,B$BZ,DU
004266  004254 6000 00                    TZE     NIT13
004267  777777 7100 00                    TRA     $ERROR
                     004270   6148 NIT14  BSS     0                                               890
004270  000000 2200 11        6149        LDX     0,T$SRW1,T      GET FRN                         900
004271  006320 7400 00        6150        STX     0,C$FRCR        SAVE IN TABLE                   910
```

INITIALIZATION -- OPEN COMMUNICATIONS NETWORK EVENTS

```
                              6152 *
                              6153 *
                              6154 *                 OPEN COMMUNICATIONS NETWORK EVENTS
                              6155 *
                              6156 *       OPEN •C$FRN0• EVENT
                              6157 *
                      004272  6158 NIT15   BSS      0
                      004272  6159         OPSCE    (0,DU),(B$TRANS,DU),(0,DU)
   004272  001005 7000 00                  TSX      0,$OPSCE
   004273  000000 0000 03                  ARG      0,DU
   004274  000001 0000 03                  ARG      B$TRANS,DU
   004275  000000 0000 03                  ARG      0,DU
                      004276  6160         CHECK    NIT16,B$BZ,NIT15
 END OF BINARY CARD IOS00130
   004276  000000 7200 11                  LXL      0,T$SRW1,T
   004277  000077 3600 03                  ANX      0,B$STMK,DU
   004300  004304 6000 00                  TZE      NIT16
   004301  000003 1000 03                  CMPX     0,B$BZ,DU
   004302  004272 6000 00                  TZE      NIT15
   004303  777777 7100 00                  TRA      $ERROR
                      004304  6161 NIT16   BSS      0
   004304  000000 2200 11     6162         LDX      0,T$SRW1,T     GET FRN
   004305  006217 7400 00     6163         STX      0,C$FRN0       SAVE IN TABLE
   004306  006345 7400 00     6164         STX      0,PFIL0        SAVE IN PASS LIST
                              6165 *
                              6166 *       OPEN •C$FRN1• EVENT
                              6167 *
                      004307  6168 NIT17   BSS      0
                      004307  6169         OPSCE    (0,DU),(B$TRANS,DU),(0,DU)
   004307  001005 7000 00                  TSX      0,$OPSCE
   004310  000000 0000 03                  ARG      0,DU
   004311  000001 0000 03                  ARG      B$TRANS,DU
   004312  000000 0000 03                  ARG      0,DU
                      004313  6170         CHECK    NIT18,B$BZ,NIT17
   004313  000000 7200 11                  LXL      0,T$SRW1,T
   004314  000077 3600 03                  ANX      0,B$STMK,DU
   004315  004321 6000 00                  TZE      NIT18
   004316  000003 1000 03                  CMPX     0,B$BZ,DU
   004317  004307 6000 00                  TZE      NIT17
   004320  777777 7100 00                  TRA      $ERROR
                      004321  6171 NIT18   BSS      0
   004321  000000 2200 11     6172         LDX      0,T$SRW1,T     GET FRN
   004322  006220 7400 00     6173         STX      0,C$FRN1       SAVE IN TABLE
   004323  006346 7400 00     6174         STX      0,PFIL1        SAVE IN PASS LIST
 END OF BINARY CARD IOS00131
   004324  006104 7400 00     6175         STX      0,C$FRLP1      SAVE FOR LP INPUT
   004325  006144 7400 00     6176         STX      0,C$FRCP1      SAVE FOR CP INPUT
   004326  006204 7400 00     6177         STX      0,C$FRCR1      SAVE FOR CR INPUT
                              6178 *
                              6179 *       OPEN •C$FRN2• EVENT
```

INITIALIZATION -- OPEN COMMUNICATIONS NETWORK EVENTS

```
                                 6180 *                                                           1210
                       004327    6181 NIT19   BSS     0                                           1220
                       004327    6182         OPSCE   (0,DU),(B$TRANS+B$PASS,DU),(0,DU)           1230
004327  001005 7000 00                        TSX     0,$OPSCE
004330  000000 0000 03                        ARG     0,DU
004331  000003 0000 03                        ARG     B$TRANS+B$PASS,DU
004332  000000 0000 03                        ARG     0,DU
                       004333    6183         CHECK   NIT20,B$BZ,NIT19                            1240
004333  000000 7200 11                        LXL     0,TSSRW1,T
004334  000077 3600 03                        ANX     0,B$STMK,DU
004335  004341 6000 00                        TZE     NIT20
004336  000003 1000 03                        CMPX    0,B$BZ,DU
004337  004327 6000 00                        TZE     NIT19
004340  777777 7100 00                        TRA     $ERROR
                       004341    6184 NIT20   BSS     0                                           1250
004341  000000 2200 11           6185         LDX     0,TSSRW1,T        GET FRN                   1260
004342  006221 7400 00           6186         STX     0,C$FRN2          SAVE IN TABLE             1270
004343  006347 7400 00           6187         STX     0,PFIL2           SAVE IN PASS LIST         1280
```

INITIALIZATION -- SPAWN ALL SUB-MODULES

```
                                6189 *                                                          1300
                                6190 *                                                          1310
                                6191 *                                                          1320
                                6192 *                SPAWN ALL SUB-MODULES                     1330
                                6193 *                                                          1340
                                6194 *        SPAWN 'LPCP' MODULE                               1350
                                6195 *                                                          1360
   004344  006260 2200 00       6196          LDX     0,C$FRPP          GET LP/CP FRN           1370
   004345  006341 7400 00       6197          STX     0,PSEG0           STICK IT IN PARAMETER LIST 1380
                      004346    6198 NIT21    BSS     0                                         1390
                      004346    6199          SPAWN   (PLOC,DU),(PLEN,DU),(0,DU)                1400
   004346  000644 7000 00                     TSX     0,$SPAWN
   004347  006327 0000 03                     ARG     PLOC,DU
   004350  000021 0000 03                     ARG     PLEN,DU
   004351  000000 0000 03                     ARG     0,DU
                      004352    6200          CHECK   NIT22,B$BZ,NIT21,B$RNA,NIT21              1410
 END OF BINARY CARD IOS00132
   004352  000000 7200 11                     LXL     0,T$SRW1,T
   004353  000077 3600 03                     ANX     0,B$STMK,DU
   004354  004362 6000 00                     TZE     NIT22
   004355  000003 1000 03                     CMPX    0,B$BZ,DU
   004356  004346 6000 00                     TZE     NIT21
   004357  000011 1000 03                     CMPX    0,B$RNA,DU
   004360  004346 6000 00                     TZE     NIT21
   004361  777777 7100 00                     TRA     $ERROR
                      004362    6201 NIT22    BSS     0                                         1420
   004362  000000 2200 11       6202          LDX     0,T$SRW1,T        GET PROCESS EVENT       1430
   004363  006254 7400 00       6203          STX     0,C$FRPR1         SAVE IN TABLE           1440
   004364  004405 7100 00       6204          TRA     NIT29             **********              1450
                                6205 *                                                          1460
                                6206 *        SPAWN 'CR' MODULE                                 1470
                                6207 *                                                          1480
   004365  006320 2200 00       6208          LDX     0,C$FRCR          GET CR FRN              1490
   004366  006341 7400 00       6209          STX     0,PSEG0           STICK IT IN PARAMETER LIST 1500
                      004367    6210 NIT25    BSS     0                                         1510
                      004367    6211          SPAWN   (PLOC,DU),(PLEN,DU),(0,DU)                1520
   004367  000644 7000 00                     TSX     0,$SPAWN
   004370  006327 0000 03                     ARG     PLOC,DU
   004371  000021 0000 03                     ARG     PLEN,DU
   004372  000000 0000 03                     ARG     0,DU
                      004373    6212          CHECK   NIT26,B$BZ,NIT25,B$RNA,NIT25              1530
   004373  000000 7200 11                     LXL     0,T$SRW1,T
   004374  000077 3600 03                     ANX     0,B$STMK,DU
   004375  004403 6000 00                     TZE     NIT26
   004376  000003 1000 03                     CMPX    0,B$BZ,DU
   004377  004367 6000 00                     TZE     NIT25
 END OF BINARY CARD IOS00133
   004400  000011 1000 03                     CMPX    0,B$RNA,DU
   004401  004367 6000 00                     TZE     NIT25
   004402  777777 7100 00                     TRA     $ERROR
```

INITIALIZATION -- SPAWN ALL SUB-MODULES

```
                  004403   6213 NIT26  BSS   0                                            1540
004403 000000 2200 11      6214        LDX   0,T$SRW1,T     GET PROCESS EVENT             1550
004404 006314 7400 00      6215        STX   0,C$FRPR2      SAVE IN TABLE                 1560
                           6216 *                                                         1570
                           6217 *            SPAWN PARAMETER LIST                         1580
                           6218 *                                                         1590
                  006327   6219        USE   STORE                                        1600
                  006327   6220 PLOC   BSS   0              PARAMETER LIST                1610
                  006327   6221        DUP   1,0            REGISTERS                     1620
006327 000000000000        6222        DEC   0                                            1630
006337 400000000000        6223        OCT   -0             TIME LIMIT                    1640
006340 000000 000010       6224        ZERO  0,B$CEND       OPTION BITS                   1650
006341 000000 000002       6225 PSEG0  ZERO  *-*,B$PF       SEGMENT ZERO                  1660
006342 000000 000000       6226 PSEG1  ZERO  0,B$VOID       SEGMENT ONE                   1670
006343 000000 000000       6227 PSEG2  ZERO  0,B$VOID       SEGMENT TWO                   1680
006344 000000 000000       6228 PSEG3  ZERO  0,B$VOID       SEGMENT THREE                 1690
006345 000000 000004       6229 PFIL0  ZERO  *-*,B$NO       SUB-SYSTEM INPUT EVENT        1700
006346 000000 000002       6230 PFIL1  ZERO  *-*,B$CA       SUB-SYSTEM OUTPUT EVENT       1710
END OF BINARY CARD IOS00134
006347 000000 000004       6231 PFIL2  ZERO  *-*,B$NO       SUB-SYSTEM INPUT FILE EVENT   1720
                  000021   6232 PLEN   EQU   *-PLOC                                       1730
                  004405   6233        USE   PREVIOUS                                     1740
```

INITIALIZATION -- OPEN PERIPHERALS

```
                             6235 *                                                               1760
                             6236 *                                                               1770
                             6237 *                 OPEN PERIPHERALS                              1780
                             6238 *                                                               1790
                             6239 *         OPEN ,OPERATOR'S CONSOLE,                             1800
                             6240 *                                                               1810
                     004405  6241 NIT29   BSS      0                                              1820
                     004405  6242         OPEN     (O$PTN,DU),(O$PTS,DU),(1,DU),(9,DU),(0,DU)     1830
 004405  000667 7000 00                   TSX      0,$OPEN
 004406  005527 0000 03                   ARG      O$PTN,DU
 004407  000014 0000 03                   ARG      O$PTS,DU
 004410  000001 0000 03                   ARG      1,DU
 004411  000011 0000 03                   ARG      9,DU
 004412  000000 0000 03                   ARG      0,DU
                     004413  6243         CHECK    NIT30,B$BZ,NIT29,B$LOCK,NIT29                  1840
 004413  000000 7200 11                   LXL      0,T$SRW1,T
 004414  000077 3600 03                   ANX      0,B$STMK,DU
 004415  004423 6000 00                   TZE      NIT30
 004416  000003 1000 03                   CMPX     0,B$BZ,DU
 004417  004405 6000 00                   TZE      NIT29
 004420  000013 1000 03                   CMPX     0,B$LOCK,DU
 004421  004405 6000 00                   TZE      NIT29
 004422  777777 7100 00                   TRA      $ERROR
                     004423  6244 NIT30   BSS      0                                              1850
 004423  000000 2200 11      6245         LDX      0,T$SRW1,T       GET FRN                       1860
 004424  005665 7400 00      6246         STX      0,R$OPFRN        SAVE IN TABLE                 1870
                             6247 *                                                               1880
                             6248 *         SEIZE OPERATOR'S CONSOLE TTY                          1890
                             6249 *                                                               1900
                     004425  6250 NIT31   BSS      0                                              1910
                     004425  6251         LOGS                      SEIZE OP CON/ LOCK IT/ SAY HELLO   1920
 004425  001517 7070 00                   TSX      L,O$LOGS
                             6252 *                                                               1930
                             6253 *         OPEN THE REST OF THE PERIPHERALS                      1940
                             6254 *                                                               1950
                     004426  6255         BRANCH   NOPASS,O$INP4    SETUP UP TASK TO READ DUMMY INPUT BUFFER 1960
 004426  001467 7000 00                   TSX      0,T$GETT
 004427  000000 6220 11                   EAX      X,0,T
 004430  000005 2210 12                   LDX      T,T$LINK,X
END OF BINARY CARD IOS00135
 004431  000000 6210 12                   EAX      T,0,X
 004432  002041 6200 00                   EAX      0,O$INP4
 004433  000004 7400 11                   STX      0,T$TRA,T
 004434  000004 6200 11                   EAX      0,Q$OFFST,T
 004435  005162 7170 00                   XED      Q$XADD+Q$TASK
 004436  000005 2210 12                   LDX      T,T$LINK,X
                     004437               BUGXR    (0,X)
                     525266       BUGBUG  SET      BUGBUG+1
 004437  525266 2200 03                   LDX      0,BUGBUG,DU
 004440  525266 2220 03                   LDX      X,BUGBUG,DU
```

INITIALIZATION -- OPEN PERIPHERALS

```
                 004441      6256        RELT                          RELEASE TCB                1970
004441  001477 7000 00                   TSX      0,T$RELT
```

INITIALIZATION -- SET UP NOTIFIES

```
                                   6258 *                                                                1990
                                   6259 *                                                                2000
                                   6260 *                                                                2010
                                   6261 *          SET UP NOTIFIES                                       2020
                                   6262 *                                                                2030
  004442  005720 2210 03           6263            LDX     T,C$LPE,DU        LINE PRINTER EVENT          2040
  004443  005720 7260 07           6264            LXL     J,C$LPE,DL                                    2050
                      004444       6265            BRANCH  PASS,C$NSRVX      RE-ISSUE NOTIFY             2060
  004444  001467 7000 00                           TSX     0,T$GETT
  004445  000000 6220 11                           EAX     X,0,T
  004446  000005 2210 12                           LDX     T,T$LINK,X
  004447  003171 6200 00                           EAX     0,C$NSRVX
  004450  000004 7400 11                           STX     0,T$TRA,T
  004451  000004 6200 11                           EAX     0,Q$OFFST,T
  004452  005162 7170 00                           XED     Q$XADD+Q$TASK
  004453  000000 6210 12                           EAX     T,0,X
                      004454                       BUGXR   (0,X)
                      525267             BUGBUG    SET     BUGBUG+1
  004454  525267 2200 03                           LDX     0,BUGBUG,DU
  004455  525267 2220 03                           LDX     X,BUGBUG,DU
                      004456       6266            RELT                      RELEASE TCB AND JCB         2070
  004456  001477 7000 00                           TSX     0,T$RELT
                                   6267 *                                                                2080
                                   6268 *                                                                2090
 END OF BINARY CARD IOS00136
  004457  005760 2210 03           6269            LDX     T,C$CPE,DU        CARD PUNCH FILE QUEUE       2100
  004460  005760 7260 07           6270            LXL     J,C$CPE,DL                                    2110
                      004461       6271            BRANCH  PASS,C$NSRVX      RE-ISSUE NOTIFY             2120
  004461  001467 7000 00                           TSX     0,T$GETT
  004462  000000 6220 11                           EAX     X,0,T
  004463  000005 2210 12                           LDX     T,T$LINK,X
  004464  003171 6200 00                           EAX     0,C$NSRVX
  004465  000004 7400 11                           STX     0,T$TRA,T
  004466  000004 6200 11                           EAX     0,Q$OFFST,T
  004467  005162 7170 00                           XED     Q$XADD+Q$TASK
  004470  000000 6210 12                           EAX     T,0,X
                      004471                       BUGXR   (0,X)
                      525270             BUGBUG    SET     BUGBUG+1
  004471  525270 2200 03                           LDX     0,BUGBUG,DU
  004472  525270 2220 03                           LDX     X,BUGBUG,DU
                      004473       6272            RELT                      RELEASE TCB                 2130
  004473  001477 7000 00                           TSX     0,T$RELT
                                   6273 *                                                                2140
                                   6274 *                                                                2150
  004474  006020 2210 03           6275            LDX     T,C$JSS,DU        JOB STREAM SCHEDULER EVENT  2160
  004475  006020 7260 07           6276            LXL     J,C$JSS,DL                                    2170
                      004476       6277            BRANCH  PASS,C$NSRVX      RE-ISSUE NOTIFY             2180
  004476  001467 7000 00                           TSX     0,T$GETT
  004477  000000 6220 11                           EAX     X,0,T
  004500  000005 2210 12                           LDX     T,T$LINK,X
```

INITIALIZATION -- SET UP NOTIFIES.

```
    004501  003171 6200 00                  EAX    0,C$NSRVX
    004502  000004 7400 11                  STX    0,T$TRA,T
    004503  000004 6200 11                  EAX    0,Q$OFFST,T
    004504  005162 7170 00                  XED    Q$XADD+Q$TASK
  END OF BINARY CARD IOS00137
    004505  000000 6210 12                  EAX    T,0,X
                        004506              BUGXR  (0,X)
                        525271       BUGBUG SET    BUGBUG+1
    004506  525271 2200 03                  LDX    0,BUGBUG,DU
    004507  525271 2220 03                  LDX    X,BUGBUG,DU
                        004510   6278       RELT                     RELEASE TCB                      2190
    004510  001477 7000 00                  TSX    0,T$RELT
                                 6279 *                                                               2200
                                 6280 *                                                               2210
    004511  006060 2210 03       6281       LDX    T,C$LP1,DU        COMMUNICATIONS INPUT FROM LP     2220
    004512  006060 7260 07       6282       LXL    J,C$LP1,DL                                         2230
                        004513   6283       BRANCH PASS,C$NSRVX                                       2240
    004513  001467 7000 00                  TSX    0,T$GETT
    004514  000000 6220 11                  EAX    X,0,T
    004515  000005 2210 12                  LDX    T,T$LINK,X
    004516  003171 6200 00                  EAX    0,C$NSRVX
    004517  000004 7400 11                  STX    0,T$TRA,T
    004520  000004 6200 11                  EAX    0,Q$OFFST,T
    004521  005162 7170 00                  XED    Q$XADD+Q$TASK
    004522  000000 6210 12                  EAX    T,0,X
                        004523              BUGXR  (0,X)
                        525272       BUGBUG SET    BUGBUG+1
    004523  525272 2200 03                  LDX    0,BUGBUG,DU
    004524  525272 2220 03                  LDX    X,BUGBUG,DU
                        004525   6284       RELT                                                      2250
    004525  001477 7000 00                  TSX    0,T$RELT
                                 6285 *                                                               2260
    004526  006120 2210 03       6286       LDX    T,C$CP1,DU        COMMUNICATIONS INPUT FROM CP     2270
    004527  006120 7260 07       6287       LXL    J,C$CP1,DL                                         2280
                        004530   6288       BRANCH PASS,C$NSRVX      RE-ISSUE NOTIFY                  2290
    004530  001467 7000 00                  TSX    0,T$GETT
    004531  000000 6220 11                  EAX    X,0,T
    004532  000005 2210 12                  LDX    T,T$LINK,X
  END OF BINARY CARD IOS00138
    004533  003171 6200 00                  EAX    0,C$NSRVX
    004534  000004 7400 11                  STX    0,T$TRA,T
    004535  000004 6200 11                  EAX    0,Q$OFFST,T
    004536  005162 7170 00                  XED    Q$XADD+Q$TASK
    004537  000000 6210 12                  EAX    T,0,X
                        004540              BUGXR  (0,X)
                        525273       BUGBUG SET    BUGBUG+1
    004540  525273 2200 03                  LDX    0,BUGBUG,DU
    004541  525273 2220 03                  LDX    X,BUGBUG,DU
    004542  004561 7100 00       6289       TRA    NITXX             **********                       2300
                        004543   6290       RELT                                                      2310
```

INITIALIZATION -- SET UP NOTIFIES

```
004543  001477 7000 00                    TSX     0,T$RELT
                             6291 *                                                          2320
                             6292 *                                                          2330
004544  006160 2210 03       6293         LDX     T,C$CR1,DU        COMMUNICATIONS INPUT FROM CR  2340
004545  006160 7260 07       6294         LXL     J,C$CR1,DL                                 2350
                  004546     6295         BRANCH  PASS,C$NSRVX      RE-ISSUE THE NOTIFY      2360
004546  001467 7000 00                    TSX     0,T$GETT
004547  000000 6220 11                    EAX     X,0,T
004550  000005 2210 12                    LDX     T,T$LINK,X
004551  003171 6200 00                    EAX     0,C$NSRVX
004552  000004 7400 11                    STX     0,T$TRA,T
004553  000004 6200 11                    EAX     0,Q$OFFST,T
004554  005162 7170 00                    XED     Q$XADD+Q$TASK
004555  000000 6210 12                    EAX     T,0,X
                  004556                  BUGXR   (0,X)
                  525274          BUGBUG  SET     BUGBUG+1
004556  525274 2200 03                    LDX     0,BUGBUG,DU
004557  525274 2220 03                    LDX     X,BUGBUG,DU
                  004560     6296         RELT                                               2370
004560  001477 7000 00                    TSX     0,T$RELT
                             6297 *                                                          2380
                             6298 *                                                          2390
                             6299 *                                                          2400
                             6300 *                                                          2410
                             6301 *****                                                      2420
                             6302 *                                                          2430
                             6303 *                                                          2440
                  004561     6304 NITXX   BSS     0                 ***********              2450
 END OF BINARY CARD IOS00139
004561  006230 2210 03       6305         LDX     T,C$PR1,DU        LPCP MODULE              2460
004562  006230 7260 07       6306         LXL     J,C$PR1,DL                                 2470
                  004563     6307         BRANCH  PASS,C$NSRVX      RE-ISSUE NOTIFY          2480
004563  001467 7000 00                    TSX     0,T$GETT
004564  000000 6220 11                    EAX     X,0,T
004565  000005 2210 12                    LDX     T,T$LINK,X
004566  003171 6200 00                    EAX     0,C$NSRVX
004567  000004 7400 11                    STX     0,T$TRA,T
004570  000004 6200 11                    EAX     0,Q$OFFST,T
004571  005162 7170 00                    XED     Q$XADD+Q$TASK
004572  000000 6210 12                    EAX     T,0,X
                  004573                  BUGXR   (0,X)
                  525275          BUGBUG  SET     BUGBUG+1
004573  525275 2200 03                    LDX     0,BUGBUG,DU
004574  525275 2220 03                    LDX     X,BUGBUG,DU
                  004575     6308         RELT                      RELEASE TCB              2490
004575  001477 7000 00                    TSX     0,T$RELT
                  004576     6309         EXIT                      ***********              2500
004576  003074 7100 00                    TRA     $EXIT
                             6310 *                                                          2510
                             6311 *                                                          2520
```

INITIALIZATION -- SET UP NOTIFIES

```
    004577  006270 2210 03        6312          LDX     T,C$PR2,DU       CR MODULE             2530
    004600  006270 7260 07        6313          LXL     J,C$PR2,DL                             2540
                       004601     6314          BRANCH  PASS,C$NSRVX                           2550
    004601  001467 7000 00                      TSX     0,T$GETT
    004602  000000 6220 11                      EAX     X,0,T
    004603  000005 2210 12                      LDX     T,T$LINK,X
    004604  003171 6200 00                      EAX     0,C$NSRVX
    004605  000004 7400 11                      STX     0,T$TRA,T
    004606  000004 6200 11                      EAX     0,G$OFFST,T
 END OF BINARY CARD IOS00140
    004607  005162 7170 00                      XED     Q$XADD+Q$TASK
    004610  000000 6210 12                      EAX     T,0,X
                       004611                   BUGXR   (0,X)
                       525276          BUGBUG SET     BUGBUG+1
    004611  525276 2200 03                      LDX     0,BUGBUG,DU
    004612  525276 2220 03                      LDX     X,BUGBUG,DU
                       004613     6315          RELT                     RELEASE TCB           2560
    004613  001477 7000 00                      TSX     0,T$RELT
                       004614     6316          EXIT                     AND WE'RE OFF'        2570
    004614  003074 7100 00                      TRA     $EXIT
                                  6317 *$*      DISK    TREENAME                               2580
```

INITIALIZATION -- TREE-NAMES & CONSTANTS

```
                  005013    6319        USE     CONST                               110
                            6320        HEAD                                        120
                            6321 *                                                  130
                            6322 *                                                  140
                            6323 *                          TREE-NAMES & CONSTANTS  150
                            6324 *                                                  160
                            6325 *              PRINT-FILE-QUEUE                    170
                            6326 *                                                  180
   005013  123131123117     6327 LPQ    UASCI   6,SYSOUT                            190
   005021  120122111116     6328        UASCI   6,PRINT-FILE-QUEUE                  200
                  000014    6329 LPQTS  EQU     *-LPQ           TREE-SIZE           210
                  004400    6330 LPQES  EQU     36*64           ELEMENT SIZE        220
                            6331 *                                                  230
                            6332 *                                                  240
                            6333 *              PUNCH-FILE-QUEUE                    250
                            6334 *                                                  260
                  005027    6335 CPQ    BSS     0               TREE-NAME           270
   005027  123131123117     6336        UASCI   6,SYSOUT                            280
 END OF BINARY CARD IOS00141
   005035  120125116103     6337        UASCI   6,PUNCH-FILE-QUEUE                  290
                  000014    6338 CPQTS  EQU     *-CPQ           TREE-SIZE           300
                  004400    6339 CPQES  EQU     36*64           ELEMENT SIZE        310
                            6340 *                                                  320
                            6341 *                                                  330
                            6342 *              PRINT-FILE-EVENT                    340
                            6343 *                                                  350
                  005043    6344 LPE    BSS     0               TREE-NAME           360
   005043  123131123117     6345        UASCI   6,SYSOUT                            370
   005051  120122111116     6346        UASCI   6,PRINT-FILE-EVENT                  380
                  000014    6347 LPETS  EQU     *-LPE           TREE-SIZE           390
                  002200    6348 LPEES  EQU     36*32           ELEMENT SIZE        400
                            6349 *                                                  410
                            6350 *                                                  420
                            6351 *              PUNCH-FILE-EVENT                    430
                            6352 *                                                  440
                  005057    6353 CPE    BSS     0               TREE-NAME           450
   005057  123131123117     6354        UASCI   6,SYSOUT                            460
 END OF BINARY CARD IOS00142
   005065  120125116103     6355        UASCI   6,PUNCH-FILE-EVENT                  470
                  000014    6356 CPETS  EQU     *-CPE           TREE-SIZE           480
                  002200    6357 CPEES  EQU     36*32           ELEMENT SIZE        490
                            6358 *                                                  500
                            6359 *                                                  510
                            6360 *              JOB STREAM SCHEDULER EVENT          520
                            6361 *                                                  530
                  005073    6362 JSS    BSS     0               TREE-NAME           540
   005073  122125102105     6363        UASCI   6,RUBENS                            550
   005101  105126105116     6364        UASCI   6,EVENT1                            560
 END OF BINARY CARD IOS00143
                  000014    6365 JSSTS  EQU     *-JSS           TREE-SIZE           570
```

INITIALIZATION -- TREE-NAMES & CONSTANTS

```
                    002200       JSSES  EQU     36*32          ELEMENT SIZE
                                 *
                                 *
                                 *              LINE PRINTER/CARD PUNCH
                                 *
                    005107       PP     BSS     0              TREE-NAME
         005107  122125102105           UASCI   6,RUBENS
         005115  130114120103           UASCI   6,XLPCPA
                    000014       PPTS   EQU     *-PP           TREE-SIZE
                    002200       PPES   EQU     36*32          ELEMENT SIZE
                                 *
                                 *
                                 *              CARD READ MODULE
                                 *
                    005123       CR     BSS     0              TREE-NAME
         005123  055055055055           UASCI   6,----
         005131  055055055055           UASCI   6,----
 END OF BINARY CARD IOS00144
                    000014       CRTS   EQU     *-CR           TREE-SIZE
                    002200       CRES   EQU     36*32          ELEMENT SIZE
                                 *$*    DISK    END
```

ASSEMBLY CONTROL CARDS

```
                                   HEAD
                          *
                          *
                          *                                   HOUSEKEEPING CARDS
                          *
                          *        HERE WE CLEAN ANY AND ALL ASSEMBLER BUGS.
                          *        THE JOB'S ARE PRE-ALLOCATED. THE LAST USED LOCATION OF
                          *        CORE IS CALCULATED AND WHAT REMAINS UP TO THE END OF THE
                          *        BAR IF ANY IS LINKED ON THE FREE MEMORY LIST.
                          *
               004615              USE     CODE
                                   HEAD
               004620              EIGHT
               004620     ZCODEL   EQU     *-ZCODE            CODE UNDER CODE
                          *
                          *
               005137              USE     CONST
                                   LIT                        FORCE LITERAL POOL HERE
               005140              EIGHT
               000320     ZCONSL   EQU     *-ZCONS            CODE UNDERCONST
                          *
                          *
               005300              USE     QSTOR
               005300              EIGHT
               000140     ZQSTRL   EQU     *-ZQSTR            CODE UNDER QSTOR
                          *
                          *
               006350              USE     STORE
               000037     TLEN     EQU     T$LEN+7            ROUND T$LEN TO
               000030     TLENR    EQU     TLEN/8*8           MULTIPLE OF EIGHT
               006350              EIGHT                      START OF DYNAMIC BUFFER AREA
               006350     SPTCB    BSS     TLENR-1            FIRST TCB (START OF DYNAMIC BUFFERS
006377 000000000000                DEC     0                  FOR THE CRUMMY LOADER
               006400              EIGHT
               001100     ZSTORL   EQU     *-ZSTOR            CODE UNDER STORE
                          *
                          *
               006400     LASTC    EQU     ZCODEL+ZCONSL+ZQSTRL+ZSTORL  LAST CARD
```

ASSEMBLY CONTROL CARDS

```
                        6426 *                                                                          500
                        6427 *                                                                          510
              006350    6428 ZTOP0   EQU     SPTCB            START OF DYNAMIC BUFFER AREA              520
              007377    6429 ZZ      EQU     LASTC+MQUAN-1    ROUND UP TO NEXT MULTIPLE                 530
              007000    6430 ZTOP    EQU     ZZ/MQUAN*MQUAN   OF CORE PAGE SIZE                         540
              000400    6431 ZZ1     EQU     ZTOP-LASTC       LENGTH OF LEFT OVER CORE                  550
                        6432 *                                                                          560
              006400    6437 NEXTF   EQU     LASTC            STICK EXTRA CORE ON FREE LIST             610
              006400    6438 NEXTB   EQU     NEXTF                                                      620
 006400  005346 000400  6439         ZERO    RSLAST,ZZ1       INITIALIZE HEADER DATA FOR LINKED LIST    630
 006401  005344 000000  6440         ZERO    RSFIRST,0                                                  640
                        6441 *                                                                          650
                        6442 *                                                                          660
 END OF BINARY CARD IOS00145
              004106    6443         TCD     $UP              MARK END OF BINARY DECK                   670
 END OF BINARY CARD IOS00146
                        6444 *                                                                          680
                        6445         DCARD   2,$                                                        690
                        6446 $       DKEND                                                              700
                        6447 $EOD                                                                       710
                        6448 *                                                                          720
                        6449 *                                                                          730
                        6450 *                                                                          740
              004106    6451 THIS    END     UP                                                         750
 END OF BINARY CARD IOS00149
 6402 IS THE NEXT AVAILABLE LOCATION.   GMAP VERSION  JMPA/062770 JMPB/062770 JMPC/062770
THERE WERE    NO  WARNING FLAGS IN THE ABOVE ASSEMBLY
```

OCTAL	SYMBOL	REFERENCES BY ALTER NO.					
2000	1K	771	771	772	2822		
24	A AD	442	442	443			
16	A CP	436	436	437	5634		
15	A CR	355	355	3532			
100060	A DO	365	365	3604	4023		
56	A DP	363	363				
3	A ID	430	430	431			
101	A LA	369	369	3750			
12	A LF	354	354	3534			
17	A LP	437	437	438	5631		
23	A MT	441	441	442			
77	A QM	368	368				
35	A RL	450	450	451			
200040	A SP	356	356	383	3547	3748	3792
52	A AST	361	361				
22	A CLM	440	440	441			
72	A COL	366	366				
32	A CPU	448	448	449			
177	A DEL	370	370				
200044	A DOL	359	359	387			
600004	A EOT	352	352	3745	3746	3961	
0	A FAC	427	427	428			
27	A MIP	445	445	446			
0	A NUL	351	351				
7	A BELL	353	353				
20	A CDRD	438	438	439			
33	A CORE	449	449	450			
13	A DISK	433	433	434			
14	A DRUM	434	434	435			
177	A MASK	373	373	3560	3743		
200073	A SCOL	367	367	410	4159	4211	4256	4301
7	A SHOP	431	431	432			
600043	A TERM	358	358	386			
600046	AAMPER	360	360	389	3958		
40	AASCLN	415	415				
4620	AASCTB	382	382	415	3752		
40	ABLANK	357	357				
1017	ACCT	2201	2201	2204	5881		
5343	ACCTT	2213	2201	2205	2206	2213	
3	ACODE	644	644	5689			
200054	ACOMMA	362	362	395			
2	ADATES	429	429	430			
15	AOPCON	435	435	436			
551	APEND	1360	1360	3629			
5322	APNDT	1373	1360	1363	1364	1365	1366	1373
240057	ASLASH	364	364	398			
37	ATIMEO	451	451				
1	ATIMES	428	428	429			
5300	AVAIL	775	775	2829	2964	3053	3096	6061
4	B AP	230	230	231	233	4683	4685	4686

```
OCTAL    SYMBOL    REFERENCES BY ALTER NO.

     3   B   BZ     266   266   906   927   952  3134  3433  3515  3630  3911  4106  4137  4555  5066  5074  5077
                         5084  5096  5269  5328  5615  5655  5668  5710  5766  5775  5824  5882  5901  5906  5912
                         5918  5931  6067  6079  6088  6102  6111  6122  6135  6147  6160  6170  6183  6200  6212
                         6243
     2   B   CA     235   235  6230
     2   B   EX     231   231   232   233
     4   B   IO     291   291  3134
     1   B   LK     232   232   233  5823
     4   B   NO     234   234   235  6229  6231
     0   B   OK     263   263
     2   B   PF     247   247  6225
     1   B   PS     246   246
    20   B   RD     228   228   229   233  4684  4686  5823
     0   B   SS     252   252
400000   B   WP     244   244
    10   B   WT     229   229   230   233
    37   B  ALL     233   233   763  4687
100000   B  CAR     215   215
    13   B  DBZ     273   273
    16   B  EOF     274   274  5074  5668
 20000   B  EOV     217   217
 10000   B  EUN     218   218
  2000   B  GET     327   327
     4   B  IOP     267   267
     5   B  ITN     281   281  5710  5906
   200   B  MOD     223   223
200000   B  NEG     214   214
 40000   B  OPR     306   306   394   396   411   413  3856  3863  4330
 40000   B  OVF     216   216
  4000   B  OVM     219   219  6059
   400   B  PAM     222   222  6059
  1000   B  PAR     221   221
 16000   B  RDY     333   333
  6000   B  REL     329   329
    11   B  RNA     296   296  6200  6212
  2000   B  RPT     198   198
  2000   B  TAL     220   220
    11   B  TLE     283   283  5296
    20   B  TMI     204   204
     2   B  TNC     207   207
    40   B  TNZ     203   203
     1   B  TOV     208   208
    10   B  TPL     205   205
     4   B  TRC     206   206
    35   B  TRO     276   276  3911  5339  5345  5352  5358  5364  5369  5383  5390
   100   B  TZE     202   202  4352
400000   B  ZER     213   213
  1000   B ABIT     199   199  2815  4352
   400   B BBIT     200   200  2815
     1   B BHDR     318   318
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO.

   200    B CBIT      201   201
    10    B CEND      242   242  6224
 14000    B DONE      332   332
    12    B HDWE      272   272
     2    B IACC      265   265
     7    B IELT      270   270
     1    B IFRN      264   264
    11    B IMOD      271   271
     5    B IPTR      268   268
    16    B IPWD      285   285
     6    B IREQ      269   269
  2000    B .GET      339   339  4130
  6000    B .REL      341   341  4568
  4000    B KILL      328   328
    13    B LOCK      284   284  3433  4106  5066  5615  6243
    26    B NSTR      275   275
     2    B PASS      255   255  6182
400000    B SIGN      302   302  4716
    77    B STMK      301   301  3134  3433  3515  3630  3911  4106  4137  4555  5066  5074  5077  5084  5096  5269
                           5296  5328  5615  5655  5668  5710  5766  5775  5824  5882  5901  5906  5912  5918  5931
                           6067  6079  6088  6102  6111  6122  6135  6147  6160  6170  6183  6200  6212  6243
400000    B TERM      303   303   304  3745  3746  3758  3830  3856  3863  3964  4330  4339
     7    B UERR      282   282
     0    B VOID      245   245  6226  6227  6228
 10000    B XXXX      330   330
     1    BANR        636   636  5725
    17    BBJBMK      316   316
     1    BBOTMK      320   320
  7777    BBSTMK      322   322
     4    BBSTRD      243   243
200000    BDELIM      304   304   305   352   356   358   359   360   362   364   367  3745  3746  3748  3758  3826
                           3832  3856  3863  4330  4339
100000    BDIGIT      305   305   306   365   399   400   401   402   403   404   405   406   407   408  3856  3863
 20000    BHDRMK      317   317  5727
     0    B.ABRT      338   338
  4000    B.KILL      340   340  4203
 12000    B.STRT      343   343  4293
 10000    B.XXXX      342   342
740000    BJOBMK      315   315
  1700    BMODMK      314   314
     0    BNPASS      254   254
 10000    BOUTMK      319   319  5730
 12000    BRSTRT      331   331
  7777    BSRTMK      321   321
     1    BTRANS      253   253  6159  6169  6182
     0    BUFSEG      769   769  3133
525200    BUG         742   742   743
525276    BUGBUG     6314   743  2348  2349  2350  2380  2522  2975  2978  3137  3218  3296  3440  3482  3483  3519
                           3537  3550  3609  3795  3836  3869  3891  3982  3983  4205  4295  4358  4477  4479  4481
                           4482  4529  4550  4570  4579  4580  4581  4582  4947  4949  5180  5181  5182  5183  5351
```

```
OCTAL    SYMBOL    REFERENCES BY ALTER NO.

                          5352  5357  5358  5363  5364  5368  5369  5382  5383  5389  5390  5781  5788  5837  5946
                          5978  6255  6265  6271  6277  6283  6288  6295  6307  6314
 5005    BYEMS      6019  6017  6019
    6    C    J
    7    C    L           5298  5500  5524  5525  5529  5530  5531
    5    C    Q
    1    C    T           5322  5325
    2    C    X           5270  5321  5501
    3    C    Y
    4    C    Z           5502  5526  5554  5562
 6120    C  CP1     5364  5364  6286  6287
 5760    C  CPE     5345  5345  5346  6269  6270
 6160    C  CR1     5369  5369  6293  6294
   24    C  ERN      568   568  5327
 3232    C  GET     5440  5413  5440
    6    C  JCB      550   550  5339  5345  5352  5358  5364  5369  5383  5390
 6020    C  JSS     5352  5352  6275  6276
 3250    C  LOG     5523  5503  5523
 6060    C  LP1     5358  5358  6281  6282
 5720    C  LPE     5339  5339  5340  5632  5744  6263  6264
    6    C  NCB      549   549  5339  5345  5352  5358  5364  5369  5383  5390
 6230    C  PR1     5383  5383  5384  6305  6306
 6270    C  PR2     5390  5390  5391  6312  6313
 3236    C  REL     5475  5415  5475
   35    C  RES      578   578  5636  5638
    2    C  RET      544   544
   33    C  RUN      574   574   577  5608
    4    C  TRA      546   546
    3    C  XED      545   545
    7    C ABBR      551   551  6014
   32    C BUSY      572   572   574  5069  5722  5937
 4761    C CLMS     5479  5476  5479
   10    C CMAX     5420  5409  5420
 3213    C COMD     5405  5358  5364  5369  5405
 6320    C FRCR     5391  5391  6150  6208
 6217    C FRN0     5375  5268  5375  6163
 6220    C FRN1     5376  5376  5835  6173
 6221    C FRN2     5377  4136  5377  5823  6186
 6260    C FRPP     5384  5384  6138  6196
 3233    C KILL     5455  5414  5455
 4753    C KLMS     5459  5456  5459
    5    C LINK      547   547   548
 3161    C NSRV     5295  5295  5323
 3177    C NSX1     5326  5326  5328
 3211    C NSX2     5329  5328  5329
 3262    C PERI     5550  5500  5550
   30    C QFRN      570   570   571  5340  5346  5384  5391  5614  5627  5765  5943
 3243    C RDY1     5499  5462  5482  5499
    0    C SRW1      542   542
    1    C SRW2      543   543
```

```
OCTAL   SYMBOL    REFERENCES BY ALTER NO.

  17    C TEM9     560   560
 767    CAUSE     2096  2096  4136  5268  5823
5741    CAUST     2112  2096  2099  2100  2101  2102  2103  2104  2112
3221    CCMDTB    5412  5412  5420
3231    CCOMDX    5428  5410  5412  5416  5428  5441  5515  5532  5565
6144    CFRCP1    5364  5364  6176
6004    CFRCPE    5345  5345  6114
6010    CFRCPQ    5346  5346  6091
6204    CFRCR1    5369  5369  6177
6044    CFRJSS    5352  5352  6125
6104    CFRLP1    5358  5358  6175
5744    CFRLPE    5339  5339  6105
5750    CFRLPQ    5340  5340  6082
6254    CFRPR1    5383  5383  6203
6314    CFRPR2    5390  5390  6215
 656    CHSEG     1710  1710  3133
5331    CHSGT     1721  1710  1713  1714  1721
 777    CKMK       744   744
   0    CKSM       634   634
 704    CLOSE     1812  1812  4554  5917
5333    CLOST     1822  1812  1815  1822
  10    CMAX      5876  5865  5876
3646    CMDTB     5868  5868  5876
3155    CMESS1    5270  5269  5270
3140    CMESSX    5267  4205  4295  4570  5267  5269  5271
3167    CNSRV1    5297  5296  5297
3171    CNSRVX    5319  3892  5296  5319  5429  5609  5788  5837  6265  6271  6277  6283  6288  6295  6307  6314
5057    CPE       6353  6110  6353  6356
2200    CPEES     6357  6110  6357
3273    CPERI1    5560  5560  5564
  14    CPETS     6356  6110  6356
5027    CPQ       6335  6087  6335  6338
4400    CPQES     6339  6087  6339
  14    CPQTS     6338  6087  6338
1000    CPST      4627  4627  4683  5364
  31    CQFLOC     571   571   572  5080  5629  5679  5721
  34    CQUEUE     577   577   578  5743
5123    CR        6380  6146  6380  6383
4074    CRASH     6003  5383  5390  6003
4767    CRDYMS    5496  5493  5496
3241    CREADY    5492  5419  5492
2200    CRES      6384  6146  6384
   5    CRLINK     548   548  5298  5832
4776    CRMS      6010  6008  6010
2000    CRST      4628  4628  4684  5369
3247    CRSTRT    5514  5417  5514
  14    CRTS      6383  6146  6383
  25    CSTATE     569   569  5327  5551
  16    CTEM10     561   561
  15    CTEM11     562   562
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO.

    14    CTEM12     563   563
    13    CTEM13     564   564
    12    CTEM14     565   565
    11    CTEM15     566   566
    10    CTEM16     567   567
    27    CTEMP1     552   552
    26    CTEMP2     553   553
    25    CTEMP3     554   554   569
    24    CTEMP4     555   555   568
    23    CTEMP5     556   556
    22    CTEMP6     557   557
    21    CTEMP7     558   558
    20    CTEMP8     559   559
    37    DACC       763   763   894
     1    DBG         40    40    43   850  1259  1313  1368  1423  1479  1525  1573  1618  1668  1716  1773  1817
                            1867  1911  1955  1999  2047  2107  2157  2208  2348  2349  2350  2380  2522  2975  2978
                            3137  3218  3296  3440  3482  3483  3519  3537  3550  3609  3795  3836  3869  3891  4205
                            4295  4358  4477  4479  4481  4482  4529  4550  4570  4579  4580  4581  4582  4680  4947
                            4949  5156  5180  5181  5182  5183  5781  5788  5837  5946  5978  6055  6255  6265  6271
                            6277  6283  6288  6295  6307  6314
   714    DESTRC    1860  1860  5905
  5334    DESTT     1872  1860  1863  1864  1865  1872
 44000    DESZ       762   762   893
     3    DISMK      649   649  5687  5891
     2    DISP       640   640  5686
  4660    DTN        759   759   761   890
    14    DTSZ       761   761   891
777777    ERROR      741   741  2354  2355  2443  2449  2450  2585  2601  2612  2623  2634  2646  2772  2774  2798
                            2830  2902  2905  2909  3054  3134  3188  3189  3215  3293  3433  3515  3630  3911  4106
                            4119  4137  4450  4456  4464  4475  4524  4528  4545  4555  4837  4849  4854  4856  4861
                            4862  4863  4895  4900  4905  4906  4907  4991  5066  5074  5077  5084  5096  5113  5163
                            5169  5170  5175  5176  5178  5179  5269  5296  5328  5418  5615  5655  5668  5710  5766
                            5775  5824  5866  5868  5869  5871  5872  5873  5875  5882  5901  5906  5912  5918  5931
                            5938  5968  5981  6067  6079  6088  6102  6111  6122  6135  6147  6160  6170  6183  6200
                            6212  6243
  3075    EXIT1     5158  5158  5194
  3134    EXIT2     5184  5184
  3074    EXIT      5155  1261  1315  1370  1425  1480  1527  1575  1620  1670  1718  1775  1819  1869  1913  1957
                            2001  2049  2109  2159  2210  2357  2398  2406  3101  4374  4584  4787  5087  5155  5276
                            5331  5795  5842  5985  6309  6316
    10    GEBORT
    22    GECALL
    27    GECHEK
    16    GEENDC
     3    GEFADD
    12    GEFCON
    13    GEFILS
     7    GEFINI
    40    GEFRCE
    41    GEFSYE
```

OCTAL	SYMBOL	REFERENCES BY ALTER NO.				
35	GEIUSE					
45	GEINFO					
1	GEINOS					
6	GELAPS					
37	GELBAR					
33	GELOOP					
11	GEMORE					
25	GEMREL					
43	GENEWS					
42	GEPRIO					
17	GERELC					
4	GERELS					
15	GERETS					
2	GEROAD					
31	GEROLL					
30	GEROUT					
24	GERSTR					
23	GESAVE					
14	GESETS					
5	GESNAP					
44	GESNUM					
20	GESPEC					
26	GESYOT					
21	GETIME					
32	GEUSER					
34	GEWAKE					
3303	INIT0	5613	5613	5615	5768	
3353	INIT1	5654	5646	5654	5655	
3363	INIT2	5659	5655	5659		
3366	INIT3	5666	5666	5668	5680	
3403	INIT4	5672	5668	5672		
3432	INIT5	5708	5708	5710		
3450	INIT6	5714	5710	5714		
3507	INIT7	5751	5747	5751		
3511	INIT8	5753	5750	5753		
3517	INIT9	5759	5745	5754	5757	5759
3301	INIT	5607	5339	5345	5607	
3531	INITX	5773	5668	5773	5775	
3315	INT0.0	5616	5615	5616		
3343	INT0.1	5642	5642	5650		
3520	INT10	5764	5764	5766		
3530	INT11	5767	5766	5767		
3406	INT4.1	5676	5676	5710		
3413	INT4.2	5681	5675	5681		
3541	INTX1	5776	5775	5776		
5	.APEND	143	143	1367		
32	.CATDR	164	164			
26	.CATLG	160	160			
42	.CATLK	172	172			
47	.CAUSE	177	177	2106		

OCTAL	SYMBOL	REFERENCES BY ALTER NO.					
22	.CHSEG	156	156	1715			
25	.CLOSE	159	159	1816			
21	.CLSEG	155	155				
57	.CRSG	183	183				
50	.DELET	178	178				
27	.DESTR	161	161	1866			
36	.EMM						
23	.EXSEG	157	157				
44	.LOCK	174	174	1954			
55	.MSTA	181	181				
46	.NOTIF	176	176	2046			
24	.OPEN	158	158	888	1772		
30	.OPENS	162	162				
36	.OPENW	168	168				
52	.OPSCE	180	180	2156			
20	.OPSEG	154	154				
17	.PAUSE	153	153	899	921	946	5191
0	.PRIV	138	138				
34	.RDACL	166	166				
37	.RDBRN	169	169				
35	.RDDIR	167	167				
40	.RDLK	170	170				
56	.RDME	182	182	2207			
4	.READ	142	142	1312			
13	.RQDT	149	149	881			
14	.RQERT	150	150				
12	.RQST	148	148	1617			
63	.RQWD	186	186				
6	.RRF	144	144	1422			
10	.SCR	146	146	913	1524		
1	.SETFV	139	139	1258			
2	.SETSQ	140	140				
15	.SPAWN	151	151	1667			
11	.SPTR	147	147	1572			
3	.SQUEZ	141	141				
16	.TERM	152	152	955			
51	.UNCAU	179	179				
45	.UNLCK	175	175	1998			
31	.UPDAT	163	163	1910			
61	.WAMI	185	185				
43	.WBRAN	173	173				
33	.WRACL	165	165				
7	.WRF	145	145	934	1477		
41	.WSINF	171	171				
60	.WTME	184	184				
6	J J		3267	3296	5115		
7	J L		3266	3295			
5	J Q						
1	J T						
2	J X		5109	5114			

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO.

     3    J     Y
     4    J     Z
    20    J    TT       528   527   528   529
    11    J   BUF       524   524   525  5661  5667  5673  5723  5732  5777  5889  5900  5902  5923  5930  5974
     2    J   FRN       514   514   515  5716  5823  5911  5917
     7    J   JOB       521   521   522  4880  5979
    20    J   LEN       529   529  3266  3294
     5    J   NCB       518   518   519   520  5623  5936
    13    J   RES       526   526   527  4844  4887  5637  5951
     5    J   TCB       519   519  5620
     3    J DISP        516   516   517  5688  5890
  1505    J GETJ       3265  3265  5618
  5700    J JTAB       5133  5112  5114  5115  5133  5980  5982  6062
    10    J MESS        523   523   524  4883  4886  5731  5823
     0    J QFRN        512   512   513   527  5628  5654  5667  5774  5900  5930
  1511    J RELJ       3291  3291  5782  5983
    12    J SIZE        525   525   526  5018  5718  5719  5753  5881  5958
     3    J TYPE        515   515   516  5019  5685  5693  5959
     6    J             734   734  3292  4844  4880  4883  4886  4887  5018  5019  5174  5619  5620  5623  5628  5630
                              5635  5637  5641  5654  5661  5667  5673  5677  5685  5688  5690  5693  5716  5718  5719
                              5723  5731  5732  5753  5774  5777  5823  5833  5881  5889  5890  5900  5902  5911  5917
                              5923  5930  5936  5951  5958  5959  5974  5979  6057  6058  6264  6270  6276  6282  6287
                              6294  6306  6313
     4    JACODE        517   517   518  5635  5690  5881
  3064    JJNUMB       5108  4879  5108
    20    JMAXJB       5129  5111  5129  5134
     1    JQFLOC        513   513   514  5630  5677  5900  5930
  5073    JSS          6362  6121  6362  6365
  2200    JSSES        6366  6121  6366
    14    JSSTS        6365  6121  6365
     6    JSTATI        520   520   521  5641  5823
     7    JSTATT        522   522   523  5833
     7    L             735   735  2351  2381  2447  2457  2461  2523  2832  2979  2998  3010  3138  3441  3484  3520
                              3538  3551  3562  3610  3638  3754  3757  3796  3837  3870  4359  4483  4583  4950  5116
                              5177  5184  5299  5561  5660  5732  5777  5888  5969  5974  6007  6008  6016  6017  6023
                              6251
  6400    LASTC        6424  6424  6429  6431  6437
   736    LOCK         1950  1950  3432  5065  5614
  5336    LOCKT        1960  1950  1953  1960
  5043    LPE          6344  6101  6344  6347
  2200    LPEES        6348  6101  6348
    14    LPETS        6347  6101  6347
  5013    LPQ          6327  6078  6327  6329
  4400    LPQES        6330  6078  6330
    14    LPQTS        6329  6078  6329
  3000    LPST         4629  4629  4685  5358
  4774    LPTB1        5962  5960  5962
  5301    MEMRQ         776   776  2969  3056
  1000    MQUAN         770   770   776  2858  2863  2864  2967  3084  3086  3130  3131  6429  6430
  5302    MTOPO         777   777  2899  2966  3051
```

OCTAL	SYMBOL	REFERENCES BY ALTER NO.															
5303	MTOP	778	778	2857	2859	2862	2900	2965	3050	3095	3129	3132	3133				
4000	MTST	4630	4630	4686													
6400	NEXTB	6438	2679	6438													
6400	NEXTF	6437	2674	6437	6438												
4233	NIT10	6123	6122	6123													
4235	NIT11	6133	6133	6135													
4251	NIT12	6136	6135	6136													
4254	NIT13	6145	6145	6147													
4270	NIT14	6148	6147	6148													
4272	NIT15	6158	6139	6158	6160												
4304	NIT16	6161	6160	6161													
4307	NIT17	6168	6168	6170													
4321	NIT18	6171	6170	6171													
4327	NIT19	6181	6181	6183													
4117	NIT1	6066	6066	6067													
4341	NIT20	6184	6183	6184													
4346	NIT21	6198	6198	6200													
4362	NIT22	6201	6200	6201													
4367	NIT25	6210	6210	6212													
4403	NIT26	6213	6212	6213													
4405	NIT29	6241	6204	6241	6243												
4127	NIT2	6077	6067	6077	6079												
4423	NIT30	6244	6243	6244													
4425	NIT31	6250	6250														
4143	NIT3	6080	6079	6080													
4145	NIT4	6086	6086	6088													
4161	NIT4.1	6089	6088	6089													
4163	NIT5	6100	6100	6102													
4177	NIT6	6103	6102	6103													
4201	NIT7	6109	6109	6111													
4215	NIT8	6112	6111	6112													
4217	NIT9	6120	6120	6122													
4106	NIT	6049	6049	6050													
4561	NITXX	6304	6289	6304													
5340	NOTFT	2052	2041	2044	2045	2052											
756	NOTIF	2041	2041	5327													
6	0 J		4247	4248													
7	0 L		3430	3439	3469	3475	3481	3511	3512	3513	3518	3531	3533	3535	3536	3546	3548
			3549	3601	3605	3608	3790	3791	3794	3824	3829	3835	3850	3853	3855	3862	3868
			3903	3904	3963	3968	3998	3999	4000	4020	4024	4025	4042	4090	4145	4154	4158
			4185	4209	4228	4250	4255	4273	4300	4325	4327	4329	4338	4357	4371	4372	
5	0 Q																
1	0 T		3891	4205	4295												
2	0 X		3891	3970	3974	3975	4091	4111	4139	4186	4205	4229	4274	4295	4344	4346	4347
3	0 Y		3852	3860	4326	4334											
4	0 Z		4092	4112	4151	4187	4230	4275	4349	4356							
2110	0 GET	4089	4065	4089	4160												
4676	0 LM1	3647	3437	3647													
4677	0 LM2	3648	3648														
1645	0 LOG	3620	3513	3563	3620	3904	4025										

OCTAL	SYMBOL	REFERENCES BY ALTER NO.
1611	0 LSP	3545 3545
4740	0 MOK	4378 4371 4378
2372	0 OPX	4370 4063 4370
5527	0 PTN	4165 4102 4105 4165 4169 6242
14	0 PTS	4169 4105 4169 6242
2253	0 REL	4227 4069 4227 4257
4705	0 CERR	4029 4020 4029
4713	0 FERR	4046 4042 4046
2124	0 GET1	4105 4105 4106
2142	0 GET2	4110 4106 4110
2167	0 GET3	4134 4134 4137 4140
2204	0 GET4	4138 4106 4137 4138
2206	0 GET5	4141 4098 4101 4141
5464	0 IBUF	3935 3910 3921 3931 3935 3938
1674	0 ICHR	3739 3739 3749 3791 3855 3862 4329
1747	0 ICMD	3849 3849 3968
5	0 ILEN	3938 3931 3938 3939
2012	0 INP1	3904 3904 4001
2015	0 INP2	3910 3910 3911 3918
2032	0 INP3	3915 3911 3915
2041	0 INP4	3923 3923 6255
5462	0 ITAL	3931 3742 3922 3931
2220	0 KILL	4184 4067 4184 4212
1620	0 LCHR	3559 3475 3533 3535 3548 3559 3605 4024
1624	0 LOCF	3570 3570
1632	0 LOCT	3600 3572 3581 3600
1655	0 LOG1	3629 3629 3630
1670	0 LOG2	3634 3626 3630 3634
1543	0 LOGC	3469 3469 4000 4020 4042 4145 4154 4371 5525 5529 6008 6016 6017
1517	0 LOGS	3429 3429 3903 3999 5524 6007 6251
1560	0 LOGX	3510 3510 3998 4372 5531 6023
4731	0 OPMS	4147 4145 4147
2332	0 PERI	4324 4090 4185 4228 4273 4324
2265	0 REL1	4244 4244
2273	0 REL2	4254 4236 4240 4254
5460	0 RFLG	3763 3740 3756 3763 3828 3834 3865 3926 3956 3966 4337
2277	0 STRT	4272 4071 4272 4302
4700	OBLNKS	3874 3851 3874
2075	OCMDER	4019 3973 4019 4043
5	OCMDLN	4073 3971 4073
12	OCMDLX	4072 4072 4073
4717	OCMDTB	4061 3972 4061 4072
1775	OENTER	3890 3890 5352
5526	OERCMD	3984 3924 3969 3984 4021
0	OFF	37 37
2105	OFORER	4041 3827 4041 4340 4348
40	OIBFSZ	3930 3905 3930 3939
1713	OICHR1	3755 3741 3755
1716	OICHR2	3758 3747 3758
1711	OICHR3	3753 3753 3759

```
OCTAL    SYMBOL    REFERENCES BY ALTER NO.

 1754    OICMD1    3855  3855  3861
 1766    OICMD2    3865  3857  3865
 1730    OIDELM    3823  3823  4158  4209  4255  4300
 1720    OINBLK    3789  3789  3829  3853  3963  4327  4338
 5463    OINCHR    3932  3906  3910  3932
 2011    OINPUT    3902  3891  3902
 2250    OKILL1    4210  4193  4210
 5525    OLCOMD    3983  3959  3975  3983
 1600    OLCRLF    3530  3512  3530  5530
 5420    OLGBUF    3657  3438  3629  3644  3645  3657
 5401    OLGCTL    3652  3470  3472  3652
 5410    OLGREG    3655  3621  3637  3655
 5400    OLGTAL    3651  3436  3561  3623  3636  3651
 4674    OLGTL0    3644  3622  3635  3644
 4675    OLGTL1    3645  3435  3645
 5402    OLNCHR    3653  3628  3629  3653
 1633    OLOCT1    3602  3602  3607
 1630    OLOCTL    3588  3588
 1626    OLOCTU    3579  3579  3590
 1545    OLOGC1    3471  3471  3476
 1552    OLOGC2    3480  3474  3480
 1521    OLOGS1    3432  3432  3433
 1533    OLOGS2    3434  3433  3434
 1563    OLOGX1    3514  3514  3515
 1573    OLOGX2    3516  3515  3516
 2045    OLSCAN    3955  3927  3955  4161  4213  4258  4303
 2051    OLSCN1    3960  3960  3965
 2057    OLSCN3    3967  3967
 2070    OLSCNX    3997  3962  3997  4026
 5461    OLTCHR    3764  3753  3755  3764  3825  3925  3957
 4701    OMORMS    4004  4000  4004
    1    ON          38    38    40    43   850  1259  1313  1368  1423  1479  1525  1573  1618  1668  1716  1773
                         1817  1867  1911  1955  1999  2047  2107  2157  2208  2348  2349  2350  2380  2522  2975
                         2978  3137  3218  3296  3440  3482  3483  3519  3537  3550  3609  3795  3836  3869  3891
                         4205  4295  4358  4477  4479  4481  4482  4529  4550  4570  4579  4580  4581  4582  4947
                         4949  5156  5180  5181  5182  5183  5781  5788  5837  5946  5978  6055  6255  6265  6271
                         6277  6283  6288  6295  6307  6314
  667    OPEN      1763  1763  4105  5709  6078  6087  6101  6110  6121  6134  6146  6242
 5332    OPENT     1778  1763  1766  1767  1768  1769  1770  1778
 2336    OPERI1    4329  4329  4335
 2346    OPERI2    4337  4331  4337
 2353    OPERI3    4345  4345  4355
 1005    OPSCE     2150  2150  6159  6169  6182
 5342    OPSET     2162  2150  2153  2154  2155  2162
 5000    OPST      4631  4631  4687
 2326    OSTRT2    4296  4281  4296
 6345    PFIL0     6229  6164  6229
 6346    PFIL1     6230  6174  6230
 6347    PFIL2     6231  6187  6231
   21    PLEN      6232  6199  6211  6232
```

OCTAL	SYMBOL	REFERENCES BY ALTER NO.
6327	PLOC	6220 6199 6211 6220 6232
5107	PP	6371 6134 6371 6374
2200	PPES	6375 6134 6375
14	PPTS	6374 6134 6374
6341	PSEG0	6225 6197 6209 6225
6342	PSEG1	6226 6226
6343	PSEG2	6227 6227
6344	PSEG3	6228 6228
6	Q J	
7	Q L	2352 2386 2585 2601 2612 2623 2634 2646
5	Q Q	2350 2585 2601 2612 2623 2634 2646
1	Q T	2346 2379
2	Q X	2350 2380 2388 2389 2395 2397 2451 2453 2454 2497 2498 2503 2504 2513 2517 2522
3	Q Y	2350 2509 2512 2515 2522
4	Q Z	2350
5140	Q OP	2586 2586 3431 3517
1103	Q DEQ	2440 2440 2585 2601 2612 2623 2634 2646
1031	Q ENQ	2338 2338 2585 2601 2612 2623 2634 2646
1130	Q INV	2493 2493 2585 2601 2612 2623 2634 2646
20	Q LEN	678 678
13	Q MAX	673 673 674 2341 2343
17	Q ABBR	677 677 678 4100
12	Q BUSY	671 671 673 2344 2346 2377 2379 2442 2444 2452 2971
5200	Q CORE	2612 2612 2767 2807 2971 3049 3099
1053	Q ENQ1	2352 2345 2352
1061	Q ENQF	2376 2376
5220	Q INP1	2623 2623 4782 5339 5345
5240	Q INP2	2634 2634 4780 5758
5260	Q INP3	2646 2646 4784 5756
1146	Q INV1	2521 2499 2505 2521
1	Q LAST	666 666 667 1173 1175 2389 2405 2446 2456 2459 2498 2504 2515 2517 2585 2601 2612 2623 2634 2646 4831 4870 4932 4935 5160 5164 5167
3	Q LINK	682 682 2397 2453 2509 2511 2513 2514 4872 4921 4934 4941 5171
5140	Q OP00	2585 2585 2586 4100
5160	Q TASK	2601 1173 1175 2455 2601 2975 3891 4205 4295 4570 4927 5159 5160 5162 5164 5166 5167 5172 5781 5788 5837 5946 5978 6255 6265 6271 6277 6283 6288 6295 6307 6314
2	Q XADD	667 667 668 2356 2455 2975 3891 4205 4295 4570 4927 5760 5781 5788 5837 5946 5978 6255 6265 6271 6277 6283 6288 6295 6307 6314
6	Q XDEQ	669 669 670 2807 3099 3517
4	Q XENQ	668 668 669 2767 3049 3431
10	Q XINV	670 670 671
5	Q	733 733 2339 2340 2341 2343 2344 2346 2356 2377 2379 2388 2389 2395 2396 2404 2405 2441 2442 2444 2445 2446 2448 2452 2454 2456 2458 2459 2497 2498 2503 2504 2512 2515 2517 4830 4831 4833 4870 4932 4935 5181 5743 5756 5758 5760
14	QAVAIL	674 674 675 2339 2340 2441
100	QBFSZ	652 652 653 5071 5660 5888 5927
4400	QELSZ	653 653
1067	QENQF0	2385 2378 2385

```
OCTAL   SYMBOL    REFERENCES BY ALTER NO.

 1074   QENQF1    2394  2394
 1100   QENQF2    2403  2390  2403
    0   QFIRST     665   665   666  2388  2395  2396  2404  2445  2448  2454  2458  2497  2503  2512  2585  2601
                        2612  2623  2634  2646  4830  4833  5159  5162  5166  5172
    4   QOFFST     681   681   682  2353  2387  2451  2509  2511  2513  2514  2975  3891  4205  4295  4570  4839
                        4869  4921  4934  4941  5168  5741  5781  5788  5837  5946  5978  6255  6265  6271  6277
                        6283  6288  6295  6307  6314
   15   QSPAR1     675   675   676
   16   QSPAR2     676   676   677
    6   R    J          2974  4471  4843
    7   R    L          2860  2865  2953  2957  3097  3128  3136  4445  4478  4517  4578  4781  4783  4785  4825
                        4879  4948  5013  5071  5081
    5   R    Q          4477  4580  4780  4782  4784  4826  4931  4947
    1   R    T          2973  2975  4570
    2   R    X          2794  2801  2803  2804  2817  2828  2829  2857  2858  2859  2903  2910  2915  2917  2930
                        2939  2954  2963  2964  2965  2966  2967  2969  2975  3050  3051  3052  3053  3055  3077
                        3094  3095  3096  4472  4473  4477  4522  4561  4570  4576  4580  4848  4880  4921  4923
                        4947  5019  5020  5068  5079
    3   R    Y          2793  2806  2818  2926  2929  3076  3080  3083  3084  3086  3087  4477  4548  4549  4580
                        4844  4845  4865  4866  4869  4870  4887  4888  4909  4911  4912  4922  4934  4935  4947
    4   R    Z          2780  2786  2789  2799  2802  2813  2822  2925  2931  2932  2955  2956  3004  3065  3070
                        3078  3082  4447  4448  4449  4451  4477  4530  4564  4580  4846  4947
 5620   R   CP    4747  4683  4747
 5623   R   CR    4748  4684  4748
 5626   R   LP    4749  4685  4749
 5634   R   MT    4750  4686  4750
 5664   R   OP    4751  4687  4751  4755
   47   R   TT    4752  4752  4753
    2   R  ACC    4653  4132  4653
    2   R  ELT    4652  4103  4652  4653  4654
    1   R  FRN    4708  4114  4548  4550  4708  4709  4755
    0   R  LEN     695   695  2795  2800  2907  2995  3005  3007  3009  3076  3083  3088  3090
    3   R  MAX    4654  4350  4455  4654  4655  4852  5559
    0   R  PTR    4650  4349  4457  4650  4651  4659  5554
 2765   R  SCR    5064  5064  5066  5946
    0   R  SET    4751  4747  4748  4749  4750  4751
    0   R ABBR    4707  4099  4127  4152  4200  4290  4354  4356  4565  4707  4708  4711  5527  5558  5560
    2   R ALLC    4710  4247  4471  4526  4529  4710  4711
400000  R BUSY    4716  4120  4192  4239  4245  4280  4458  4472  4575  4716  4717  4747  4748  4749  4750  4751
    2   R FLAG    4709  4096  4121  4191  4234  4238  4245  4279  4459  4473  4523  4535  4547  4577  4709  4710
 1152   R GETC    2765  2765  3185  3266  5071  5660  5888
 2377   R GETP    4444  4444  5013
 5346   R LAST    2678  2678  2939  3077  6439
 1403   R MEM1    3066  3066  3071
 1412   R MEM2    3076  3068  3069  3076
 1422   R MEM3    3084  3081  3084
 1440   R MEMX    3098  3059  3060  3072  3085  3098
 1241   R MORE    2856  2788  2856
 5370   R MREG    3016  2896  2904  2963  2977  3016  3017  3018
 1443   R MREQ    3128  2860  3097  3128
```

OCTAL	SYMBOL	REFEFENCES BY ALTER NO.
4	R OMAX	4655 4116 4544 4655 4656 4992 5339 5345
6	R OPER	4657 4118 4541 4657 4658
5620	R PERT	4745 4745 4752
1252	R RELC	2895 2865 2895 3217 3295 5081 5732 5777 5974
2447	R RELP	4516 4250 4516 5969
100000	R RSVE	4718 4120 4237 4245 4458 4536 4546 4718
2777	R SCR1	5067 5066 5067
3005	R SCR2	5073 5073 5074
3022	R SCR3	5075 5074 5075 5077 5093
3033	R SCR4	5078 5077 5078 5096
3037	R SCR5	5082 5070 5082 5084
3047	R SCR6	5085 5084 5085
3051	R SCR7	5091 5074 5091
3053	R SCR8	5094 5094 5096
2565	R SKED	4779 4779 5781 5978
5376	R TEMP	3018 2898 2901 2903 2906 2908 3006 3008 3018
5350	R TRAP	3015 2973 2974 3015 3016
5	RAVAIL	4656 4117 4474 4531 4542 4543 4656 4657 4976
200000	RCLOSE	4717 4097 4120 4235 4458 4546 4717 4718 4747 4748 4749 4750 4751
1	RCPMAX	4747 4683 4747
5550	RCPTAB	4683 4607 4683 5345
1	RCRMAX	4748 4684 4748
5560	RCRTAB	4684 4608 4684
5550	RDEVHR	4682 4344 4682 4689
3	RDEVLN	4711 4353 4356 4461 4711 4753 5562
5620	RDEVTB	4746 4746
10	RDHLEN	4659 4344 4346 4659
50	RDHRLN	4689 4689
536	READ	1305 1305 3910 5073 5667
5321	READT	1318 1305 1308 1309 1310 1311 1318
5344	RFIRST	2674 2674 2780 2910 3065 6440
1154	RGETC1	2768 2768
1165	RGETC3	2778 2778 2866
1212	RGETC4	2804 2797 2804
1234	RGETC5	2827 2827
1167	RGETC6	2781 2781 2787
1176	RGETC7	2789 2784 2785 2789
1227	RGETC8	2819 2819 2823
2412	RGETP1	4459 4459 4463
2420	RGETP2	4469 4460 4469
5374	RINSRT	3017 2917 2925 2956 2996 3017
1	RLINKB	696 696 2793 2802 2805 2806 2926 2928 2929 2931 3004 3078 3082 3092 3093
0	RLINKF	694 694 695 696 2794 2801 2804 2915 2930 2932 2997 3002 3003
5674	RLJOBS	5039 4995 5024 5039 5967
2	RLPMAX	4749 4685 4749
5570	RLPTAB	4685 4609 4685 5339 5956
2763	RLPTB1	5028 5021 5028
1365	RMEMCK	3048 2975 3048
5673	RM.320	5037 5030 5037 5752 5963
5672	RM.512	5036 5029 5036 5749 5962

OCTAL	SYMBOL	REFERENCES BY ALTER NO.									
1245	RMORE1	2861	2861								
1450	RMREQ1	3133	3133	3134							
1463	RMREQ2	3135	3134	3135							
10	RMTMAX	4750	4686	4750							
5600	RMTTAB	4686	4610	4686							
5665	ROPFRN	4755	3432	3514	3629	3910	4755	6246			
1	ROPMAX	4751	4687	4751							
5610	ROPTAB	4687	4611	4687							
15	RPERTL	4753	4753								
4001	RQMAX	772	772	2773							
634	RQST	1613	1613								
5327	RQSTT	1623	1613	1616	1623						
1276	RRELC1	2924	2924	2940							
1306	RRELC2	2938	2916	2938							
1310	RRELC3	2952	2934	2952							
1350	RRELC4	2994	2953	2957	2994						
1344	RRELCX	2976	2968	2970	2972	2976					
2505	RRELP1	4554	4554	4555							
2515	RRELP2	4556	4555	4556							
2545	RRELP3	4571	4537	4571							
564	RRF	1415	1415	5900							
5323	RRFT	1428	1415	1418	1419	1420	1421	1428			
2751	RSCPAL	5010	4901	5010							
2735	RSCPCK	4973	4857	4973	4994	4997					
2751	RSCRAL	5011	4902	5011							
2735	RSCRCK	4974	4858	4974							
5671	RS.320	5035	5035	5751							
5670	RS.512	5034	5034	5748							
2603	RSKDR0	4834	4834	4874							
2612	RSKDR1	4845	4845	4867							
2635	RSKDR2	4864	4864	4977	4978						
2640	RSKDR3	4868	4868	4979	4996						
2645	RSKDR4	4878	4847	4878							
2657	RSKDR5	4890	4890	4915							
2666	RSKDR6	4898	4898	4914							
2700	RSKDR7	4908	4908	5014							
2707	RSKDR8	4919	4892	4919							
2722	RSKDR9	4940	4933	4940							
2724	RSKDRX	4946	4832	4871	4936	4946					
2575	RSKEDR	4824	4781	4783	4785	4824					
2753	RSLPAL	5017	4903	5017							
2741	RSLPCK	4989	4859	4989							
2751	RSMTAL	5012	4904	5012	5022	5023	5025				
2735	RSMTCK	4975	4860	4975							
7	RSPARE	4658	4658	4659							
1	RSTATE	4651	4125	4198	4288	4562	4651	4652			
4744	RTABCP	4607	4607	4617							
4745	RTABCR	4608	4608	4618							
4743	RTABLE	4605	4449	4605	4617	4618	4619	4620	4621	4848	5553
4746	RTABLP	4609	4609	4619							

```
OCTAL    SYMBOL     REFERENCES BY ALTER NO.

 4747    RTABMT      4610  4610  4620
 4750    RTABOP      4611  4611  4621
    5    RTMAX        748   748
    1    RTYPCP      4617  4617  4627  5345
    2    RTYPCR      4618  4618  4628
    3    RTYPLP      4619  4619  4629  5339
    4    RTYPMT      4620  4620  4630
    5    RTYPOP      4621  4621  4631
 3614    RUN1        5825  5824  5825
 3577    RUN         5818  5736  5818  5824  5827
 5620    RXDVHR      4688  4347  4688
  612    SCR         1519  1519  5076  5911
 5325    SCRT        1530  1519  1522  1523  1530
 5320    SETFT       1264  1254  1257  1264
  526    SETFV       1254  1254  6066
  510    SETUP       1158  1158  1256  1307  1362  1417  1472  1521  1569  1615  1663  1712  1765  1814  1862  1908
                           1952  1996  2043  2098  2152  2203  5324
 2606    SKR0.1      4838  4838  4942
  644    SPAWN       1661  1661  6199  6211
 6350    SPTCB       6418  6056  6057  6418  6428
  623    SPTR        1567  1567  5095
 5326    SPTRT       1578  1567  1570  1571  1578
 5330    SPWNT       1673  1661  1664  1665  1666  1673
    6    T    J            3190
    7    T    L            3185  3217
    5    T    Q
    1    T    T            3186  3187  3218
    2    T    X
    3    T    Y
    4    T    Z
    6    T  JCB       476   476   550  3190  4248  4527  4843  5174  5619  6058
   30    T  LEN       478   478   481   570  3015  3016  3185  3216  6415
    6    T  NCB       475   475   476   549  5622  5678  5720  5742  5784
    2    T  RET       465   465   544
    4    T  TRA       470   470   546  1159  2352  2386  2975  3891  4205  4295  4445  4478  4479  4517  4570  4578
                           4579  5177  5180  5737  5781  5788  5837  5946  5978  6255  6265  6271  6277  6283  6288
                           6295  6307  6314
    3    T  XED       466   466   545  1164
 1467    T GETT      3184  2975  3184  3891  4205  4295  4570  5320  5617  5781  5788  5837  5946  5978  6255  6265
                           6271  6277  6283  6288  6295  6307  6314
    5    T LINK       474   474   547  2975  3186  3891  4205  4295  4570  5322  5326  5621  5781  5788  5837  5946
                           5978  6255  6265  6271  6277  6283  6288  6295  6307  6314
 1477    T RELT      3213  3100  3213  4373  4786  5086  5275  5330  5792  5794  5841  5984  6256  6266  6272  6278
                           6284  6290  6296  6308  6315
    0    T SRW1       463   463   542  1162  3134  3433  3515  3630  3911  3916  4106  4113  4137  4139  4555  5066
                           5074  5077  5084  5092  5096  5269  5270  5296  5328  5615  5655  5668  5710  5715  5766
                           5775  5824  5826  5882  5901  5906  5912  5918  5931  6067  6079  6081  6088  6090  6102
                           6104  6111  6113  6122  6124  6135  6137  6147  6149  6160  6162  6170  6172  6183  6185
                           6200  6202  6212  6214  6243  6245
    1    T SRW2       464   464   543  5406  5555  5717  5862
```

```
OCTAL    SYMBOL     REFERENCES BY ALTER NO.

    17   T TEM9      492   492   493   560  3128  3136  3137
     1   T           729   729  1159  1162  1164  2352  2353  2386  2387  2397  2766  2769  2775  2777  2779  2789
                           2796  2799  2811  2817  2828  2831  2975  3052  3055  3057  3058  3080  3087  3089  3094
                           3128  3134  3136  3137  3190  3214  3430  3433  3439  3440  3469  3481  3482  3511  3515
                           3518  3519  3531  3536  3537  3546  3549  3550  3601  3608  3609  3630  3790  3794  3795
                           3824  3835  3836  3850  3868  3869  3891  3911  3916  4091  4092  4104  4105  4106  4111
                           4112  4113  4115  4126  4131  4133  4136  4137  4139  4151  4186  4187  4199  4204  4205
                           4229  4230  4248  4249  4274  4275  4289  4294  4295  4325  4357  4358  4445  4446  4447
                           4451  4470  4476  4478  4479  4480  4481  4482  4517  4521  4522  4527  4530  4549  4554
                           4555  4561  4563  4564  4569  4570  4576  4578  4579  4825  4826  4835  4838  4845  4866
                           4888  4893  4897  4898  4909  4912  4913  4920  4922  4931  4948  4949  5065  5066  5068
                           5072  5073  5074  5076  5077  5079  5081  5083  5084  5092  5095  5096  5162  5164  5168
                           5171  5174  5177  5180  5268  5269  5270  5296  5298  5321  5322  5326  5328  5406  5457
                           5477  5494  5501  5502  5525  5526  5551  5555  5608  5614  5615  5619  5620  5621  5622
                           5643  5648  5655  5668  5678  5682  5692  5696  5709  5710  5715  5717  5720  5737  5741
                           5742  5743  5744  5765  5766  5775  5781  5783  5784  5788  5793  5824  5826  5832  5834
                           5836  5837  5862  5882  5891  5901  5904  5905  5906  5912  5918  5931  5944  5945  5946
                           5952  5971  5978  6014  6056  6058  6067  6079  6081  6088  6090  6102  6104  6111  6113
                           6122  6124  6135  6137  6147  6149  6160  6162  6170  6172  6183  6185  6200  6202  6212
                           6214  6243  6245  6255  6263  6265  6269  6271  6275  6277  6281  6283  6286  6288  6293
                           6295  6305  6307  6312  6314
  4772   TABLE      5700  5695  5700
   100   TAL         747   747  3920  4153  5528  6015
777700   TALMK       746   746  3625
    40   TALYB       745   745  3920  4153  5528  6015
  3656   TERM0      5880  5880  5882
  3670   TERM1      5883  5870  5874  5882  5883
  3640   TERM       5861  5831  5861
  6402   THIS       6451  6451
    37   TLEN       6415  6415  6416
    30   TLENR      6416  6416  6418
     4   TN          645   645  5691  5903
     1   TNSZ        635   635  5674  5905
   521   TRAP       1172  1163  1172
  3670   TRM1       5887  5887
  3754   TRM2       5922  5918  5922
  3762   TRM3       5929  5929  5931
  3775   TRM4       5932  5931  5932
  4024   TRM5       5950  5939  5950
  4025   TRM6       5952  5952  5972
  4043   TRM7       5969  5957  5965  5966  5969
  4050   TRM8       5973  5954  5973
  3744   TRMC       5916  5895  5906  5912  5916  5918
  3714   TRMD1      5902  5901  5902  5906
  3701   TRMD       5899  5893  5899  5901
  3733   TRMS       5910  5894  5906  5910  5912
     7   TSPARE      477   477   551
    16   TTEM10      493   493   494   561  2777  2796  2799  2828
    15   TTEM11      494   494   495   562
    14   TTEM12      495   495   496   563  2769  2775  2779  2789  2811  2817
```

```
OCTAL   SYMBOL     REFERENCES BY ALTER NO.

   13   TTEM13      496   496   497   564
   12   TTEM14      497   497   498   565
   11   TTEM15      498   498   499   566
   10   TTEM16      499   499   567  2766  2831
   27   TTEMP1      481   481   482   552  3052  4104  4105  4126  4136  4199  4289  4446  4447  4451  4470  4476
                         4480  4481  4521  4522  4530  4561  4564  4570  4576  5065  5073  5076  5083  5095  5268
                         5457  5477  5494  5525  5643  5648  5692  5709  5904  5905  5944  5946
   26   TTEMP2      482   482   483   553  3055  3057  4131  4136  4204  4294  4482  4570  5068  5079  5268  5501
                         5502  5526  5682  5709  5945  5946  5952  5971
   25   TTEMP3      483   483   484   554  3058  3080  4115  4136  4549  4554  4563  4570  4825  4826  4931  4948
                         4949  5072  5073  5081  5696  5709  5834
   24   TTEMP4      484   484   485   555  3087  3089  3094  4133  4136  4569  4570  4845  4866  4888  4909  4912
                         5836
   23   TTEMP5      485   485   486   556  3824  3835  3836  4091  4092  4111  4112  4151  4186  4187  4229  4230
                         4249  4274  4275  4893  4898
   22   TTEMP6      486   486   487   557  3531  3536  3537  3546  3549  3550  3601  3608  3609  3850  3868  3869
                         4325  4357  4358  4897  4913
   21   TTEMP7      487   487   488   558  3430  3439  3440  3469  3481  3482  3511  3518  3519  3790  3794  3795
   20   TTEMP8      488   488   492   559  4835  4838  4920  4922
    2   TYPE        637   637   640  5683  5728  5746
    1   TYPMK       648   648  5020  5684  5694
  746   UNLCK      1994  1994  3514  5083  5765  5774
 5337   UNLKT      2004  1994  1997  2004
   31   UNUSED      447   432   433   439   440   443   444   445   446   447   448
 4106   UP         6050   792  6050  6443  6451
  726   UPDATE     1906  1906  5654
 5335   UPDTT      1916  1906  1909  1916
 3135   WAIT       5191  5161  5191
  577   WRF        1470  1470  5930
 5324   WRFT       1483  1470  1473  1474  1475  1476  1483
    6   X      J
    7   X      L
    5   X      Q
    1   X      T
    2   X      X
    3   X      Y
    4   X      Z
    6   X     FT    798   798
    0   X     FV    791   791  6066
   51   X     IC    834   834   875   878
   24   X     OP    812   812
   34   X    DIV    820   820
   14   X    DRL    804   804
  401   X    FT1    887   887   907
  425   X    FT2    912   904   912   928
  445   X    FT3    933   926   933   953
   50   X    IC1    833   833   880
    2   X    MEM    794   794
    4   X    MME    796   796
   26   X    ONC    814   814
```

```
OCTAL   SYMBOL    REFERENCES BY ALTER NO.

 160    X  REG     852   852  1113  1117  1122
  36    X  XEC     822   822
5304    X CKIC    1128  1110  1124  1128
 474    X CKPT    1109  1109  1259  1313  1368  1423  1479  1525  1573  1618  1668  1716  1773  1817  1867  1911
                        1955  1999  2047  2107  2157  2208  5156  6055
  52    X DATE     838   838   883
 170    X DBGQ     855   852   855  1119  1121
  16    X LOCK     806   806
  40    X REGS     832   832   877
  55    X SBAR     841   841   939   940
  30    X STRT     816   816
 470    X TERM     954   908   929   951   954  6027
  53    X TIME     839   839
   2    X          730   730  2453  2509  2514  2802  2805  2806  2818  2820  2907  2915  2926  2931  2975  3002
                        3005  3078  3082  3093  3891  3972  3974  3976  4103  4116  4117  4118  4125  4132  4198
                        4205  4288  4295  4349  4350  4523  4526  4529  4535  4547  4548  4550  4562  4570  4577
                        4852  4881  4976  4992  5021  5069  5080  5112  5115  5171  5172  5181  5325  5621  5622
                        5623  5627  5629  5632  5636  5638  5673  5674  5676  5677  5678  5679  5683  5686  5689
                        5691  5723  5725  5728  5746  5781  5788  5826  5837  5923  5924  5926  5928  5936  5937
                        5943  5945  5946  5959  5960  5978  5979  5980  5982  6255  6265  6271  6277  6283  6288
                        6295  6307  6314
5310    XCKREG    1130  1112  1123  1130
  12    XCOMND     802   802
  20    XCONCT     808   808
  20    XDBGQN     853   853   855  1119
 370    XFAULT     874   793   795   797   799   801   803   805   807   809   811   813   815   817   819   821
                         823   874
  32    XOFLOW     818   818
  22    XPARTY     810   810
  56    XPATCH     846   846
  10    XTIMER     800   800
 414    XTRAP1     902   889   902   903   905   915   936
 435    XTRAP2     924   914   924   925
 461    XTRAP3     949   935   949   950
   3    Y          731   731  2511  2513  2804  2821  2932  2955  3866  4846  4850  4891  4910  4923  4934  5181
                        5636  5644  5649  5746  5783  5793
   4    Z          732   732  2783  2786  2793  2794  2795  2800  2801  2803  2928  2929  2930  2954  2995  2997
                        3003  3007  3009  3067  3070  3076  3083  3088  3090  3092  4096  4099  4114  4121  4127
                        4152  4191  4200  4234  4238  4246  4247  4279  4290  4354  4449  4455  4457  4474  4531
                        4541  4542  4543  4544  4565  4848  4855  5181  5527  5558  5560  5637  5643  5645  5647
                        5648  5683  5684  5685  5686  5687  5688  5689  5690
   0    ZCODE      108   108  6400
4620    ZCODEL    6400  6400  6424
4620    ZCONS      112   112  6406
 320    ZCONSL    6406  6406  6424
5140    ZQSTR      116   116  6411
 140    ZQSTRL    6411  6411  6424
5300    ZSTOR      120   120  6421
1100    ZSTORL    6421  6421  6424
6350    ZTOPO     6428   777  6428
```

OCTAL	SYMBOL	REFERENCES BY ALTER NO.					
7000	ZTOP	6430	778	6430	6431		
400	ZZ1	6431	775	6060	6431	6433	6439
7377	ZZ	6429	6429	6430			

** 22862 WORDS OF MEMORY WERE USED BY GMAP FOR THIS ASSEMBLY.

APPENDIX III

Listing of peripheral driver prototype - Line Printer and Card Punch module are implemented.

PAGE 139 THERE WERE NO WARNING FLAGS IN THE ABOVE ASSEMBLY

ASSEMBLY CONTROL CARDS

```
                           1        TTLS    ASSEMBLY CONTROL CARDS
                           2        TTL     LINE PRINTER/ CARD PUNCH MODULE
                           3        HEAD
                           4 *
                           5 *
                           6 *                              HOUSEKEEPING CARDS
                           7 *
                           8 *
                           9        PCC     ON              PRINT CONTROL CARDS
                          10        PMC     ON              EXPAND MACROS
                          11        DETAIL  ON              EXPAND OCT.DEC.BCI.ASCII.DUP.ETC.
                          12        DETAIL  OFF
                          13        PCC     OFF
          000000          19        ORG     0               SET ORIGIN
                          20        HEAD                    AND UNHEADED
                          21        LBL     LPCP.LPCP       L-INE P-RINTER/ C-ARD P-UNCH MODULE
                          22 *
                          23 *
                          24 *
                          25 *      SQUASH ASSEMBLER BUG
                          26 *
    001000 400013         27 MME    OPD     012/0010.6/.02/2.6/.6/.02/2.02/3
                          28 *
                          29 *
                          30 *
                          31 *              DEBUGGING AIDS
                          32 *
                          33 *      IF THE DEBUG SWITCH IS SET ON. THEN THE DEBUG MACROS
                          34 *      WILL BE EXPANDED. OTHERWISE NOT.
                          35 *
          000000          36 OFF    EQU     0               OFF SWITCH
          000001          37 ON     EQU     1               ON SWITCH
                          38 *
          000001          39 DBG    SET     ON              INITIALLY ON
                          40 *
                          41 *
                          43 *
                          44 TTL    OPSYN   TTLS            FOR SYSPROG PPRNT
```

```
        HEAD
 *
 *
 *                                            ASSEMBLY DECK SETUP
 *
 *
 *      GMAP ASSEMBLY DECK SETUP -- THE DECK SETUP FOR ASSEMBLING THE
 *      THIS PROGRAM (HEREAFTER LPCP) IS AS FOLLOWS.  ALL CONTROL
 *      CARDS BEGIN WITH A '$'.  THE FOLLOWING DECK WILL PRODUCE
 *      A NEW K* TAPE, A P* TAPE FOR THE LISTING AND IS ALSO USED
 *      TO PRODUCE A SYMBOL TABLE FOR DDT.  THE SECOND ACTIVITY,
 *      SYSPROG PPRNT, CREATES THE SYMBOL TABLE AND STORES THE
 *      RESULTS IN THE FILE 'RUBENS/LPCP-SYM-TAB.  TO DO AN ASSEMBLY
 *      SIMPLY SUBMIT THE FOLLOWING PROGRAM TO TECOS:
 *              SOURCE RUBENS/<NAME-OF-DECK>$RUN$EXIT
 *      ALTER CARDS SHOULD BE SAVED PREVIOUSLY IN FILE 'RUBENS/ALTERS'
 *
 *
 *$     SNUMB   MBR
 *$     IDENT   RUBENS,MICHAEL
 *$     GMAP    COMDK,DECK
 *$     TAPE    G*,X1D,,TAPE#A,,LPCPA
 *$     TAPE    *1,X2R
 *$     TAPE    K*,X3D,,,,LPCPB
 *$     TAPE    P*,X4S,,,,LPCPPRINT
 *$     ASCII   A*,BCDT,RUBENS/ALTERS
 *$     SYSPROG PPRNT
 *$     TAPE    IN,X4D,,,,LPCPPRINT
 *$     PRMFL   A1,S,R,RUBENS/LPCP-SYM-TAB-A
 *$     COMMENT PLEASE PRINT P* 'LPCPPRINT' FOR MBR.  THANX.
 *$     ENDJOB
 *$EOD
```

```
         HEAD
*
*
*                                        BINDER DECK SETUP
*
*
*        THE BINDER DECK SETUP -- THE DECK SETUP FOR LOADING THE
*        ASSEMBLED PROGRAM INTO THE R & DC SYSTEM IS AS FOLLOWS.  NOTE
*        THAT THE DECK IS LOADED VIA 'MULTIBINDER' AND NOT THE USUAL
*        'BINDER'.  THIS IS NECESSARY IF BOTH MULTI-SEGMENTS AND USE
*        COUNTERS ARE TO BE USED.
*
         DCARD   3,$
$BUILD   SPAWNSYS,RUBENS,MULTIBINDER
$        FNAME   */XLPCPA
$        OBJECT
```

LOCATION COUNTERS

```
                  HEAD
           *
           *
           *                                LOCATION COUNTERS
           *
           *
           *      THE FOLLOWING PREDEFINES THE ORDER IN WHICH THE LOCATION
           *      COUNTERS WILL OCCUR. INDEPENDENT OF THE ORDER IN WHICH
           *      THEY ARE USED WITHIN THE PROGRAM.
           *
000000            USE     CODE              MAIN PROGRAM SEGMENT
000000      ZCODE BSS     0
           *
           *
003060            USE     CONST             STORAGE FOR CONSTANTS. TABLES
003060      ZCONS BSS     0
           *
           *
003260            USE     QSTOR             FOR ALL QUEUES
003260      ZQSTR BSS     0
           *
           *
003320            USE     STORE             FOR ALL DYNAMIC STORAGE
003320      ZSTOR BSS     0
           *
           *
000000            USE     CODE              CODE LOCATION COUNTER INITIALLY
```

```
                HEAD
         *
         *
         *                                 EXECUTIVE CONSTANTS
         *
         *
         *      THE DEFINITIONS OF SYSTEM PARAMETERS SUCH AS MME NUMBERS
         *      ARE INDICATED BY SYMBOLS WHICH START WITH A DECIMAL POINT.
         *      THEY ARE NOT HEADED SO A TYPICAL REFERENCE IS LDX X0.S.OPEN.DU.
         *
         *                 DEFINITION OF MME NUMBERS
         *
000000   .PRIV  EQU   0              MICRO-CODED PRIVILEGED PRIMITIVE
000001   .SETFV EQU   1              SET UP FAULT VECTOR
000002   .SETSQ EQU   2              SET UP SQUEEZE MODE
000003   .SQUEZ EQU   3              ENTER SQUEEZE MODE
000004   .READ  EQU   4              READ
000005   .APEND EQU   5              APPEND
000006   .RRF   EQU   6              READ RANDOM FILE
000007   .WRF   EQU   7              WRITE RANDOM FILE
000010   .SCR   EQU   8              SCRATCH
000011   .SPTR  EQU   9              SET POINTER
000012   .RQST  EQU   10             REQUEST STATUS
000013   .RQDT  EQU   11             REQUEST DATE AND TIME
000014   .RQERT EQU   12             REQUEST ELAPSED RUN TIME
000015   .SPAWN EQU   13             SPAWN
000016   .TERM  EQU   14             TERMINATE
000017   .PAUSE EQU   15             PAUSE
000020   .OPSEG EQU   16             OPEN SEGMENT
000021   .CLSEG EQU   17             CLOSE SEGMENT
000022   .CHSEG EQU   18             CHANGE SEGMENT LENGTH
000023   .EXSEG EQU   19             EXCHANGE TWO SEGMENTS
000024   .OPEN  EQU   20             OPEN
000025   .CLOSE EQU   21             CLOSE
000026   .CATLG EQU   22             CATALOGUE
000027   .DESTR EQU   23             DESTROY
000030   .OPENS EQU   24             OPEN SCRATCH
000031   .UPDAT EQU   25             UPDATE
000032   .CATDR EQU   26             CATALOGUE DIRECTORY
000033   .WRACL EQU   27             WRITE ACCESS CONTROL LIST
000034   .RDACL EQU   28             READ ACCESS CONTROL LIST
000035   .RDDIR EQU   29             READ DIRECTORY
000036   .OPENW EQU   30             OPEN WORKING DIRECTORY
000037   .RDBRN EQU   31             READ BRANCH
000040   .RDLK  EQU   32             READ LINK
000041   .WSINF EQU   33             WRITE SYSTEM INFORMATION
000042   .CATLK EQU   34             CATALOGUE LINK
000043   .WBRAN EQU   35             WRITE BRANCH
000044   .LOCK  EQU   36             LOCK
000045   .UNLCK EQU   37             UNLOCK
```

DEFINITIONS OF EXECUTIVE SYSTEM CONSTANTS

```
000046    174 .NOTIF EQU    38        NOTIFY
000047    175 .CAUSE EQU    39        CAUSE
000050    176 .DELET EQU    40        DELETE EVENT
000051    177 .UNCAU EQU    41        UNCAUSE EVENT
000052    178 .OPSCE EQU    42        OPEN SCRATCH EVENT
000055    179 .MSTA  EQU    45        SYSTEM STATUS MEASUREMENTS
000056    180 .RDME  EQU    46        MEASURE READ ME
000057    181 .CRSG  EQU    47        CREATE SEGMENT
000060    182 .WTME  EQU    48        WRITE ME
000061    183 .WAMI  EQU    49        WHO AM I
000063    184 .RQWD  EQU    51        REQUEST WORKING DIRECTORY
```

BIT AND STATUS DEFINITIONS

```
                 HEAD    B
           *
           *
           *                         BIT DEFINITIONS
           *
           *
           *
           *             DEFINITIONS FOR REPEAT INSTRUCTIONS
           *
002000     RPT   BOOL    002000      COUNT FIELD FOR RPT INSTRUCTIONS
001000     ABIT  BOOL    001000      INCREMENT FIRST INDEX REGISTER
000400     BBIT  BOOL    000400      INCREMENT SECOND INDEX REGISTER
000200     CBIT  BOOL    000200      LOAD C(XO) FROM REPEAT INSTRUCTION
000100     TZE   BOOL    000100      ZERO
000040     TNZ   BOOL    000040      NON ZERO
000020     TMI   BOOL    000020      NEGATIVE
000010     TPL   BOOL    000010      POSITIVE
000004     TRC   BOOL    000004      CARRY
000002     TNC   BOOL    000002      NO CARRY
000001     TOV   BOOL    000001      OVERFLOW
           *
           *
           *             DEFINITIONS FOR INDICATOR REGISTER
           *
400000     ZER   BOOL    400000      ZERO INDICATOR BIT
200000     NEG   BOOL    200000      NEGATIVE
100000     CAR   BOOL    100000      CARRY
040000     OVF   BOOL    040000      OVERFLOW
020000     EOV   BOOL    020000      EXPONENT OVERFLOW
010000     EUN   BOOL    010000      EXPONENT UNDERFLOW
004000     OVM   BOOL    004000      OVERFLOW MASK -- ON PREVENTS OVERFLOW FAULTS
002000     TAL   BOOL    002000      TALLY RUNOUT
001000     PAR   BOOL    001000      PARITY ERROR
000400     PAM   BOOL    000400      PARITY ERROR MASK
000200     MOD   BOOL    000200      MASTER MODE
           *
           *
           *             DEFINITION FOR ACCESS BITS
           *
000020     RD    BOOL    20          READ ACCESS
000010     WT    EQU     RD/2        WRITE
000004     AP    EQU     WT/2        APPEND
000002     EX    EQU     AP/2        EXECUTE
000001     LK    EQU     EX/2        LOCK
000037     ALL   EQU     RD+WT+AP+EX+LK ALL
000004     NO    BOOL    4           NOTIFY
000002     CA    EQU     NO/2        CAUSE
           *
```

```
                   *
                   *
                   *                  DEFINITIONS FOR EXEC STATUS RETURNS
                   *
                   *                  LOGICAL STATUS CODE FOR I/O PRIMITIVES
                   *
000000             OK      BOOL       00                 OK
000001             IFRN    BOOL       01                 INVALID FRN
000002             IACC    BOOL       02                 INVALID ACCESS SPECIFIED
000003             BZ      BOOL       03                 EXECUTIVE TOO BUSY
000004             IOP     BOOL       04                 INVALID OPERATION FOR THIS DEVICE
000005             IPTR    BOOL       5                  COPY POINTER IS OUT OF BOUNDS
000006             IREQ    BOOL       6                  AMOUNT REQUESTED GREATER THAN FILE LENGTH
000007             IELT    BOOL       07                 ELEMENT SIZE IS NOT A MULTIPLE OF UNIT SIZE
000011             IMOD    BOOL       11                 INVALID MODE
000012             HDWE    BOOL       12                 HARDWARE ERROR -- OPERATION NOT COMPLETE
000013             DBZ     BOOL       13                 DEVICE UNAVAILABLE (HARDWARE)
000016             EOF     BOOL       16                 END-OF-FILE ENCOUNTERED
000026             NSTR    BOOL       26                 NO FILE STORAGE AVAILABLE
000035             TRO     BOOL       35                 TIMER RUN OUT ON OPERATOR'S CONSOLE
                   *
                   *
                   *                  LOGICAL STAUS CODE FOR FILE AND EVENT PRIMITIVES
                   *
000005             ITN     BOOL       5                  INVALID NAME
000007             UERR    BOOL       7                  UNRECOVERABLE ERROR
000011             TLE     BOOL       11                 TIME LIMIT EXCEEDED
000013             LOCK    BOOL       13                 ITEM LOCKED
000016             IPWD    BOOL       16                 INVALID PASSWORD
                   *
                   *
                   *                  LOGICAL STATUS CODES FOR CONTROL PRIMITIVES
                   *
                   *
000004             IO      BOOL       4                  I/O ACTIVITY IN PROGRESS
                   *
                   *
                   *                  MISCELLANEOUS BITS
                   *
000077             STMK    BOOL       77                 STATUS BIT MASK
400000             SIGN    BOOL       400000             SIGN BIT
400000             TERM    BOOL       400000             TERMINATOR BIT
200000             DELIM   EQU        TERM/2             DELIMINATOR
100000             DIGIT   EQU        DELIM/2            DIGIT
040000             OPR     EQU        DIGIT/2            OPERATOR
                   *
                   *
```

```
               *
               *
               *                  DEFINITIONS FOR J$FLAG
               *
               *                                  J$FLAGS (LOWER)
               *
      400000     KILL    BOOL     B$SIGN          KILL THIS JOB
      200000     RHDR    EQU      KILL/2          FAKE A READ OF A HEADER BUFFER
      100000     WHDR    EQU      RHDR/2          WRITE A HEADER BUFFER
      040000     ENDR    EQU      WHDR/2          END OF FILE REACHED ON READING
      020000     ENDW    EQU      ENDR/2          END OF FILE REACHED ON WRITING
      010000     HALT    EQU      ENDW/2          HALT READING TO DO SPTR (RESTART)
               *
      000400     LP      BOOL     400             LINE PRINTER
      000200     CP      BOOL     200             CARD PUNCH
               *
               *                                  J$FLAGS (UPPER)
               *
      001700     MODMK   BOOL     1700            MEDIA MASK FOR RCW (BITS 26 - 29)
      740000     JOBMK   BOOL     740000          JOB NUMBER MASK (TOP 4 BITS)
      000017     BJBMK   BOOL     17              4 BIT MASK
      020000     HDRMK   BOOL     020000          HEADER MASK (ON = OUTPUT BANNER)
      000001     BHDR    BOOL     1               HEADER BIT MASK
      010000     OUTMK   BOOL     010000          OUTPUT TYPE MASK (OFF = 512; ON = 320)
      000001     BOTMK   BOOL     1               OUTPUT BIT MASK
      007777     SRTMK   BOOL     007777          START ADDRESS IN ELEMENTS OF J$BUFSZ
      007777     BSTMK   BOOL     7777            BIT MASK
               *
               *
               *                  BITS RETURNED TO MONITOR
               *
      002000     GET     BOOL     002000          PERIPHERAL GOTTEN (NOT SENT)
      004000     ABORT   BOOL     004000          PERIPHERAL ABORTED
      006000     REL     BOOL     006000          PERIPHERAL RELEASED
      010000     XXXX    BOOL     010000          NOT USED
      012000     RSTRT   BOOL     012000          PERIPHERAL RESTARTED
      014000     DONE    BOOL     014000          JOB FINISHED SUCCESSFULLY
      016000     RDY     BOOL     016000          READY PERIPHERAL
               *
               *
               *                  CARD PUNCH MODES
               *
      000000     CP0     BOOL     0               STATUS
      000001     CP1     BOOL     1               ATTENTION
      000002     CP2     BOOL     2               WRITE CARD BINARY
      000003     CP3     BOOL     3               WRITE CARD HOLLERITH
      000004     CP4     BOOL     4               WRITE CARD HOLLERITH EDITED
```

```
                    *
                    *
                    *                     LINE PRINTER MODES
                    *
000000              LP0      BOOL     0                   STATUS
000001              LP1      BOOL     1                   ATTENTION
000002              LP2      BOOL     2                   WRITE EDITED CONTINOUS, NO SLEW
000003              LP3      BOOL     3                   EDITED, NO SLEW
000004              LP4      BOOL     4                   EDITED, SLEW 1 LINE
000005              LP5      BOOL     5                   EDITED, SLEW 2 LINES
000006              LP6      BOOL     6                   EDITED, SLEW TOP OF FORM
000007              LP7      BOOL     7                   NON EDITED, NO SLEW
000010              LP10     BOOL     10                  NON EDITED, SLEW 1 LINE
000011              LP11     BOOL     11                  NON EDITED, SLEW 2 LINES
000012              LP12     BOOL     12                  NON EDITED, SLEW TOP OF FORM
000013              LP13     BOOL     13                  SLEW 1 LINE
000014              LP14     BOOL     14                  SLEW 2 LINES
000015              LP15     BOOL     15                  SLEW TOP OF FORM
```

B TRAP BLOCK DESCRIPTION

```
                 HEAD    T
       *
       *
       *                                 TCB
       *
       *
       *         THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
       *         IN THE TRAP BLOCK (TBLOCK).
       *
000000   SRW1    EQU     0               FIRST STATUS RETURN WORD FROM EXEC
000001   SRW2    EQU     1               SECOND STATUS RETURN WORD
000002   RET     EQU     2               SAVED IC/IR WHEN EXEC SPRINGS TRAP
000003   XED     EQU     3               CONTROL IS TRANSFERRED HERE WHEN EXEC
       *                                 SPRINGS THE TRAP.  IT CONTAINS AN XED
       *                                 OF A CHAIN WHICH LINKS THE TRAP TO THE
       *                                 MASTER TASK QUEUE.
000004   TRA     EQU     4               (UPPER) RESTART ADDRESS FOR TASKS ON
       *                                 ON A QUEUE (SUCH AS THE QSTASK)
       *                                 (LOWER) MAY BE USED TO SAVE RETURN
       *                                 FROM A REENTRANT ROUTINE
000005   LINK    EQU     5               (UPPER) LINK TO PREVIOUS TCB
000006   NCB     EQU     6               (UPPER) POINTER TO NCB
000006   JCB     EQU     NCB             (LOWER) POINTER TO JCB
000007   SPARE   EQU     7               SPARE
000030   LEN     EQU     24              LENGTH OF TCB (NICE IF MULTIPLE OF 8)
       *
       *
000027   TEMP1   EQU     LEN-1           TEMPORARY STORAGE AT END OF BLOCK
000026   TEMP2   EQU     TEMP1-1         MORE TEMPORARY STORAGE
000025   TEMP3   EQU     TEMP2-1
000024   TEMP4   EQU     TEMP3-1
000023   TEMP5   EQU     TEMP4-1
000022   TEMP6   EQU     TEMP5-1
000021   TEMP7   EQU     TEMP6-1
000020   TEMP8   EQU     TEMP7-1
       *
       *         NO ONE EXCEPT RSGETC SHOULD USE TEMP9 - TEM16
       *
000017   TEM9    EQU     TEMP8-1
000016   TEM10   EQU     TEM9-1
000015   TEM11   EQU     TEM10-1
000014   TEM12   EQU     TEM11-1
000013   TEM13   EQU     TEM12-1
000012   TEM14   EQU     TEM13-1
000011   TEM15   EQU     TEM14-1
000010   TEM16   EQU     TEM15-1
```

```
        HEAD    J
*
*
*                               JCB
*
*
*
*       THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
*       IN THE JOB CONTROL BLOCK (HERAFTER CALL JCB).
*
*       PLEASE NOTE THAT THE NUMBER OF JCB'S IS (AND MUST BE) EQUAL
*       TO THE NUMBER OF DEVICES AT MAX.  THEREFORE IN ORDER TO
*       MAKE MORE EFFICIENT USAGE OF THE DYNAMIC BUFFER AREA
*       THE JCB'S ARE ALL PRE-ALLOCATED.  THIS IS DONE FOR THE FOL-
*       REASONS:
*               (1)  SINCE A JCB IS ALLOCATED AT JOB INITIALIZATION AND IS
*                   NOT RELEASED UNTIL JOB COMPLETION. WE DON'T WANT MEMCORY
*                    TIED DOWN THAT LONG.
*               (2)  ALSO WE DON'T WANT TO CREATE LARGE HOLES IN MEMORY (E.G.
*                    A JCB GRABBING A RELEASED 320 WORD BLOCK
*               (3)  THERE ARE ONLY AS MANY JCB'S AS DEVICE (CURRENTLY 3)
*
*       THE FOLLOWING MACROS CAN BE USED TO MANAGE JCB'S:
*
*               GETJ            GETS A JCB     ( C(XJ) = JCB-ADDRESS)
*               RELJ            RELEASES A JCB ( C(XJ) = DESTROYED)
```

J JOB CONTROL BLOCK DESCRIPTION

```
                *
                *
                *
        777777  ALLC    EQU     -1                ALLOCATED FLAG WORD (0 = FREE)
        000000  FLAGS   EQU     0                 FLAG BITS
        000001  FLAG    EQU     1                 TSSRW2 WHEN CAUSED
        000002  IFRN    EQU     2                 (UPPER) INPUT FRN
        000003  OFRN    EQU     3                 (UPPER) OUTPUT FRN
                                                  (LOWER) PERIPHERAL UNIT NUMBER
        000004  CARD1   EQU     4                 FIRST PUNCH CARD (0 = MESSAGE TO OPER)
        000005  BUFSZ   EQU     5                 (LOWER) WORKING BUFFER SIZE
        000006  RSTRT   EQU     6                 (LOWER) RESTART ELEMENT ADDRESS
        000007  RCW     EQU     7                 (UPPER) PTR TO NEXT RCW
                                                  (LOWER) MUST BE ZERO
        000010  XDATA   EQU     8                 (UPPER) PTR TO END OF INPUT DATA
        000011  EMPTY   EQU     9                 (UPPER) RESTART ADDRESS FOR ENQ
                                                  (LOWER) SEMAPHORE -- NUMBER OF BUFFERS TO FILL
        000012  FULL    EQU     10                (UPPER) RESTART ADDRESS FOR END
                                                  (LOWER) SEMAPHORE -- NUMBER OF BUFFERS TO EMPTY
        000013  RPTR    EQU     11                (UPPER) PTR TO NEXT INPUT BUFFER INDIRECT
                                                  (LOWER) MUST BE ZERO
        000014  WPTR    EQU     12                (UPPER) PTR TO NEXT OUTPUT BUFFER INDIRECT
                                                  (LOWER) MUST BE ZERO
        000015  RTASK   EQU     13                (UPPER) POINTER TO READ TASK
        000015  WTASK   EQU     RTASK             (LOWER) POINTER TO WRITE TASK
        000016  RES     EQU     WTASK+1           TYPE OF RESOURCE REQUIRED
        000017  RTRY    EQU     RES+1             RETRY COUNTER
        000020  RESET   EQU     RTRY+1            (UPPER) PTR TO END OF BUFFER QUEUE
                                                  (LOWER) PTR TO START OF BUFFER QUEUE
                                                  RESET MUST BE JUST BEFORE BUFFER QUEUE.
                *
        000002  N       EQU     2                 NUMBER OF WORKING BUFFERS
        000024  LEN     EQU     RESET+1+N-ALLC    JCB LENGTH
                                                  NOT NECESSARY TO BE EVEN
```

J NOTIFY BLOCK DESCRIPTION

```
                 HEAD    C
           *
           *
           *                             NCB
           *
           *
           *     THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
           *     IN THE NOTIFY BLOCK (ALIAS NCB).
           *
           *             A NCB IS A TCB WITH EXTRAS.
           *
000000     SRW1  EQU     T$SRW1
000001     SRW2  EQU     T$SRW2
000002     RET   EQU     T$RET
000003     XED   EQU     T$XED
000004     TRA   EQU     T$TRA
000005     LINK  EQU     T$LINK
000005     RLINK EQU     LINK             (LOWER) RESTART AFTER NOTIFY
000006     NCB   EQU     T$NCB
000006     JCB   EQU     T$JCB
000007     SPARE EQU     T$SPARE
000027     TEMP1 EQU     T$TEMP1
000026     TEMP2 EQU     T$TEMP2
000025     TEMP3 EQU     T$TEMP3
000024     TEMP4 EQU     T$TEMP4
000023     TEMP5 EQU     T$TEMP5
000022     TEMP6 EQU     T$TEMP6
000021     TEMP7 EQU     T$TEMP7
000020     TEMP8 EQU     T$TEMP8
000017     TEM9  EQU     T$TEM9
000016     TEM10 EQU     T$TEM10
000015     TEM11 EQU     T$TEM11
000014     TEM12 EQU     T$TEM12
000013     TEM13 EQU     T$TEM13
000012     TEM14 EQU     T$TEM14
000011     TEM15 EQU     T$TEM15
000010     TEM16 EQU     T$TEM16
000030     ERN   EQU     T$LEN            (UPPER) EVENT FRN
000031     STATE EQU     ERN+1            STATE
000032     MESS  EQU     STATE+1          36 BITS MESSAGE (OPTIONAL)
000033     RES   EQU     MESS+1           TYPE OF RESOURCE REQUIRED (LP/ CP)
```

C NOTIFY BLOCK DESCRIPTION

```
*
*
*       THIS MACRO IS USED TO GENERATE THE NECESSARY NOTIFY CONTROL
*       BLOCKS FOR THE COMMUNICATIONS NETWORK.
*
*
*
NCB     MACRO   LABEL,RESTART-ADD,ERN,STATE,MESSAGE,RESOURC-TYPE,PER-BITS
        USE     STORE
        EIGHT
#1      BSS     0                   LABEL
        ZERO    0,B$TRO             SRW1; SIMULATE A TIMER RUNOUT
        ZERO    0,0                 SRW2
        ZERO    0,0                 RET
        ZERO    0,0                 XED
        ZERO    **,0                TRA
        ZERO    0,#2                LINK/ RESTART
        ZERO    *-C$NCB,*-C$JCB     *NCB/ JCB POINT TO THEMSELVES
        DEC     0
        DUP     1,16                TEMP1 THRU TEM16
        DEC     0
        ZERO    #3,                 ERN
        ZERO    #4,0                STATE
        VFD     36/#5               MESSAGE
        ZERO    #6,#7               RESOURCE TYPE / PERIPHERAL BIT
        USE     PREVIOUS
        ENDM    NCB
```

```
                 533        HEAD    Q
                 534 *
                 535 *
                 536 *                              QCB
                 537 *
                 538 *
                 539 *        THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS
                 540 *        IN A QBLOCK GENERATED BY THE QUEUE MACRO.
                 541 *
000000           542 FIRST  EQU     0               POINTER TO FIRST BLOCK OF QUEUE
000001           543 LAST   EQU     FIRST+1         POINTER TO LAST BLOCK OF QUEUE
000002           544 XADD   EQU     LAST+1          INSTRUCTION PAIR FOR ADDINGA BLOCK
000004           545 XENQ   EQU     XADD+2          INSTRUCTION PAIR FOR ENQUEUEING
000006           546 XDEQ   EQU     XENQ+2          INSTRUCTION PAIR FOR DEQUEUEING
000010           547 BUSY   EQU     XDEQ+2          RESPONSIBLE BLOCK IF QUEUE IS BUSY
                 548                                ZERO OTHERWISE
000011           549 MAX    EQU     BUSY+1          MAXIMUM NUMBER OF ITEMS ASSOCIATED WITH Q
000012           550 AVAIL  EQU     MAX+1           NUMBER OF ITEMS CURRENTLY AVAILABLE
000013           551 SPAR1  EQU     AVAIL+1         SPAR1
000014           552 SPAR2  EQU     SPAR1+1         SPAR2
000015           553 ABBR   EQU     SPAR2+1         ASCII ABBREVIATION OF QUEUE
000016           554 LEN    EQU     ABBR+1          LENGTH OF QUEUE (WISE TO KEEP EVEN)
                 555 *
                 556 *
000004           557 OFFST  EQU     4               OFFSET FOR QUEUE POINTER
000003           558 LINK   EQU     OFFST-1         FORWARD LINK POINTER
```

Q CORE MANAGEMENT DEFINITIONS

```
                HEAD    R
         *
         *
         *                             CCB
         *
         *
         *      THESE SYMBOLS ARE USED IN REFERENCING SPECIFIED WORDS IN
         *      A BLOCK ON THE FREE MEMORY LIST.
         *
000000     LINKF  EQU     0                POINTER TO SUCCESSOR (UPPER)
000000     LEN    EQU     LINKF            TOTAL LENGTH (IN WORDS) OF BLOCK (LOWER)
000001     LINKB  EQU     LINKF+1          POINTER TO PREDECCESSOR
```

```
                 HEAD
         *
         *
         *                                  GLOBALS
         *
         *                  HEAD SYMBOL USAGE
         *
                 HEAD                       GLOBAL AND EXEC CONSTANTS
                 HEAD       B               GENERAL PURPOSE BITS
                 HEAD       C               COMMUNICATIONS ROUTINES
                 HEAD       J               JCB SYMBOLS AND ROUTINES
                 HEAD       Q               QUEUE SYMBOLS AND ROUTINES
                 HEAD       R               RESOURCE ALLOCATION
                 HEAD       S               STATISTICS COUNTERS
                 HEAD       T               TRAP SYMBOLS AND ROUTINES
                 HEAD       X               DIAGNOSTIC ROUTINES
                 HEAD
         *
         *
         *                  INDEX REGISTER DEFINITIONS
         *
         *       THE SYMBOLIC INDEX REGISTERS USED IN THIS PROGRAM ARE
         *       ONE CHARACTER SYMBOLS, DEFINED UNDER EACH HEAD SYMBOL
         *       IN USE IN THE PROGRAM.  INDEX REGISTER 0 IS SPECIAL,
         *       SINCE IT IS USED FOR REPEAT INSTRUCTIONS, SO IT IS NOT
         *       SYMBOLIC.
         *
                 HEAD       0,C,J,Q,R,T,X
         *
000001   T       EQU        1               TRAP BLOCK POINTER
000002   X       EQU        2               TEMP
000003   Y       EQU        3               TEMP
000004   Z       EQU        4               TEMP
000005   Q       EQU        5               QUEUES AND GENERAL USE
000006   J       EQU        6               JOB NUMBER
000007   L       EQU        7               LINK REGISTER FOR SUBROUTINE CALLS
         *
         *
         *                  OTHER GLOBAL SYMBOLS
         *
                 HEAD
777777   ERROR   EQU        -1              USED TO GENERATE MEMORY FAULTS
525200   BUG     BOOL       525200          BUGGING QUANTITY
525200   BUGBUG  SET        BUG
001000   MQUAN   EQU        512             QUANTUM INCREMENT FOR MEMORY REQUEST
000777   CKMK    BOOL       777             STATUS MASK (9 BITS)
000040   TALYB   BOOL       40              TALLYB BIT
777700   TALMK   BOOL       777700          MASK FOR TALLY COUNT FIELD
000100   TAL     BOOL       100             TALLY DISPLACEMENT
000005   RTMAX   EQU        5               RETRY ERROR ONLY 5 TIMES
```

GLOBAL CONSTANTS

```
                                  624          HEAD
                                  625 *
                                  626 *
                                  627 *                                  GLOBALS
                                  628 *
                                  629 *
                                  630 *                     COMMUNICATIONS FRN'S
                                  631 *
                       000000     632 FRN0     EQU     0                 INPUT EVENT FRN
                       000001     633 FRN1     EQU     1                 OUTPUT EVENT FRN
                       000002     634 FRN2     EQU     2                 INPUT PASS EVENT FRN
                                  635 *
                                  636 *                 COMMUNICATIONS STATES
                                  637 *
                       001000     638 CPST     BOOL    1000              CP STATE C(AU)
                       003000     639 LPST     BOOL    3000              LP STATE C(AU)
                                  640 *
                                  641 *
                                  642 *
                                  643 *                DUMPFILE PARAMETERS
                                  644 *
                       003060     645          USE     CONST
003060  122125102105              646 DTN      UASCI   6,RUBENS
003066  114120103120              647          UASCI   6,LPCPDUMP
                       000014     648 DTSZ     EQU     *-DTN             TREE-SIZE
                       044000     649 DESZ     EQU     36*512            ELEMENT SIZE (ONE PAGE OF CORE)
                       000037     650 DACC     BOOL    BSALL
                       000000     651          USE     PREVIOUS
                                  652 *
                                  653 *
                                  654 *                MEMORY MANAGEMENT PARAMETERS
                                  655 *
                       000000     656 BUFSEG   EQU     0                 SEGMENT WHERE DYNAMIC BUFFER IS LOCATED
                       002000     657 1K       EQU     1024              ONE K DECIMAL
                       004001     658 RQMAX    EQU     2*1K+1            MAX MEMORY REQUEST/SHOT
                                  659 *
                       003320     660          USE     STORE
003320  000010 000000             661 AVAIL    ZERO    ZZ1,0             MEMORY AVAILABLE
003321  001000 000000             662 MEMRQ    ZERO    MQUAN,0           MEMORY REQUIREMENT/ NEED ONE UNIT ALWAYS
003322  003740 0000 00            663 MTOPO    ARG     ZTOPO             END OF PROGRAM
003323  004000 0000 00            664 MTOP     ARG     ZTOP              TOP OF MEMORY
                       000000     665          USE     PREVIOUS
```

LOW CORE ALLOCATION -- FAULT VECTOR

```
                          000000          USE     CODE
                                          HEAD    X
                                    *
                                    *
                                    *                              FAULT VECTOR
                                    *
                                    *     CONSIDER ALL FAULTS FATAL
                                    *
                          000000          ORG     0                IN CASE OF PRECEEDING ERRORS. FORCE ZERO
      000000  002750 7100 00        FV    TRA     SANIT            ANITIALIZATION; 0 = SHUTDOWN
      000001  000370 7170 00              XED     FAULT
      000002  000000 000000         MEM   ZERO                     1 = MEMORY
      000003  000370 7170 00              XED     FAULT
  END OF BINARY CARD LPCP0003
      000004  000000 000000         MME   ZERO                     2 = MASTER MODE ENTRY
      000005  000370 7170 00              XED     FAULT
      000006  000000 000000         FT    ZERO                     3 = FAULT TAG
      000007  000370 7170 00              XED     FAULT
      000010  000000 000000         TIMER ZERO                     4 = TIMER RUNOUT
      000011  000370 7170 00              XED     FAULT
      000012  000000 000000         COMND ZERO                     5 = COMMAND
      000013  000370 7170 00              XED     FAULT
      000014  000000 000000         DRL   ZERO                     6 = DERAIL
      000015  000370 7170 00              XED     FAULT
      000016  000000 000000         LOCK  ZERO                     7 = LOCKUP
      000017  000370 7170 00              XED     FAULT
      000020  000000 000000         CONCT ZERO                     8 = CONNECT
      000021  000370 7170 00              XED     FAULT
      000022  000000 000000         PARTY ZERO                     9 = PARITY
      000023  000370 7170 00              XED     FAULT
      000024  000000 000000         OP    ZERO                     10 = ILLEGAL OP CODE
      000025  000370 7170 00              XED     FAULT
      000026  000000 000000         ONC   ZERO                     11 = OPERATION NOT COMPLETE
      000027  000370 7170 00              XED     FAULT
      000030  000000 000000         STRT  ZERO                     12 = STARTUP
      000031  000370 7170 00              XED     FAULT
  END OF BINARY CARD LPCP0004
      000032  000000 000000         OFLOW ZERO                     13 = OVERFLOW
      000033  000370 7170 00              XED     FAULT
      000034  000000 000000         DIV   ZERO                     14 = DIVIDE CHECK
      000035  000370 7170 00              XED     FAULT
      000036  000000 000000         XEC   ZERO                     15 = EXECUTE
      000037  000370 7170 00              XED     FAULT
```

```
          X                    LOW CORE ALLOCATION -- DEBUG STORAGE

                            709 *
                            710 *
                            711 *                                DEBUG STORAGE
                            712 *
                            713 *        STORAGE FOR REGISTERS AND IC ON CRASH
                            714 *
                   000040   715          EIGHT
                   000040   716 REGS     BSS      8              STORAGE FOR CRASH REGISTERS
000050  000000 000000       717 IC1      ZERO                    IC BEFORE FAULT
000051  000000 000000       718 IC       ZERO                    IC AT FAULT
                            719 *
                            720 *        DATE AND TIME OF CRASH
                            721 *
                   000052   722 DATE     BSS      1              DATE OF CRASH
                   000053   723 TIME     BSS      1              TIME OF CRASH
000054  001101070701        724 DATEA    DATE                    ASSEMBLY DATE
000055  000000 0000 00      725 SBAR     ARG      0              BAR SETTING WHEN CRASHED
                            726 *
                            727 *        PATCH AREA
                            728 *
                   000056   729          EVEN
                   000056   730 PATCH    BSS      64             LEAVE LOTS OF ROOM
                            731 *
                            732 *        STORAGE FOR DEBUGGING QUEUE
                            733 *
                   000160   735          EIGHT                   QUASH STUPID ASSEMBLER BUG.
000160  000170 0000 00      736 REG      ARG      DBGQ           POINTER TO NEXT ENTRY
                   000020   737 DBGQN    EQU      16             NUMBER OF ENTRIES
                   000170   738          EIGHT
                   000170   739 DBGQ     BSS      8*DBGQN        RESERVE SPACE
```

```
                      000370          USE     CODE
                                      HEAD    X
                               *
                               *
                               *                              DIAGNOSTICS
                               *
                               *      THIS SECTION IS ENTERED FROM THE FAULT VECTOR IN THE
                               *      EVENT OF A PROGRAMMING ERROR.  THEN REGISTERS ARE
                               *      PRESERVED.  THE ENTIRE PROGRAM IS WRITTEN OUT INTO
                               *      THE FILE RUBENS/LPCPDUMP.
                               *
                               *      CONSIDER ALL FAULTS FATAL.
                               *
                               *      ENTER FROM THE FAULT VECTOR BY
                               *              XED X$FAULT
                               *
                      000370          EVEN
                      000370   FAULT  BSS     0               ENTRY POINT
   000370  000051 5540 00             STC1    IC              SAVE IC AND IR
   000371  000372 7100 00             TRA     *+1             BREAK XED
   000372  000040 7530 00             SREG    REGS            SAVE REGISTERS
   000373  000051 2200 00             LDX     0,IC            FIND IC+1 AT FAULT
 END OF BINARY CARD LPCP0005
   000374  777776 2350 10             LDA     -2,0            GET SAVED IC
   000375  000050 7550 00             STA     IC1             FOR SPECIAL LOCATION
   000376  000013 2200 03             LDX     0,$,RQDT,DU     REQUEST DATE AND TIME
   000377  000000 0010 00             MME
   000400  000052 7570 00             STAQ    DATE            SAVE DATE AND TIME
                               *
                               *      OPEN DUMP FILE
                               *
                      000401   FT1    BSS     0               OPEN THE DUMP FILE
   000401  000024 2200 03             LDX     0,$,OPEN,DU     MME NUMBER
   000402  000414 2210 03             LDX     1,TRAP1,DU      TRAP
   000403  003060 2240 03             LDX     4,$DTN,DU       TREE-NAME
   000404  000014 2250 03             LDX     5,$DTSZ,DU      TREE-SIZE
   000405  000001 2260 03             LDX     6,1,DU          BEHALF
   000406  044000 2270 03             LDX     7,$DESZ,DU      ELEMENT-SIZE
   000407  000037 2360 07             LDQ     $DACC,DL        ACCESSES
   000410  000000 0010 00             MME                     TRY OPENING
                               *
                               *      PAUSE TILL OPENED
                               *
   000411  000017 2200 03             LDX     0,$,PAUSE,DU    PAUSE INDEFINITELY
   000412  000000 0010 00             MME                     WAIT FOR TRAP
   000413  000411 7100 00             TRA     *-2             KEEP WAITING
                      000414   TRAP1  BSS     3               TRAP ON DUMPFILE OPEN
   000417  000414 2220 00             LDX     2,TRAP1         GET FRN
   000420  000425 6010 00             TNZ     FT2             DID WE REALLY GET IT OPEN?
   000421  000414 7200 00             LXL     0,TRAP1         NO, SEE WHY NOT
```

```
          X                         DIAGNOSTICS

   000422  000003 1000 03                  CMPX    0,BSBZ,DU      WAS THE EXEC TOO BUSY?
   000423  000401 6000 00                  TZE     FT1            YES, SO RETRY
END OF BINARY CARD LPCP0006
   000424  000470 7100 00                  TRA     TERM           NOPE, WELL THAT'S IT
                                   *
                                   *       SCRATCH DUMP FILE
                                   *
                        000425     FT2     BSS     0
   000425  000010 2200 03                  LDX     0,$.SCR,DU     MME NUMBER
   000426  000435 2210 03                  LDX     1,TRAP2,DU     TRAP ADDRESS
   000427  000414 2220 00                  LDX     2,TRAP1        FRN OF FILE
   000430  000000 2230 03                  LDX     3,0,DU         SCRATCH IT
   000431  000000 0010 00                  MME
                                   *
                                   *       PAUSE TILL SCRATCHED
                                   *
   000432  000017 2200 03                  LDX     0,$.PAUSE,DU   ALWAYS PAUSE
   000433  000000 0010 00                  MME                    WAIT
   000434  000432 7100 00                  TRA     *-2            FOREVER
                        000435     TRAP2   BSS     3
   000440  000435 7200 00                  LXL     0,TRAP2        CHECK EXEC STATUS RETURN
   000441  000445 6000 00                  TZE     FT3            OK, CONTINUE
   000442  000003 1000 03                  CMPX    0,BSBZ,DU      NO, WELL WAS THE EXEC TOO BUSY?
   000443  000425 6000 00                  TZE     FT2            YES, JUST RETRY
   000444  000470 7100 00                  TRA     TERM           NOPE, JUST BLEWIT
                                   *
                                   *       START CORE TO FILE DUMP
                                   *
                        000445     FT3     BSS     0
   000445  000007 2200 03                  LDX     0,$.WRF,DU     WRITE RANDON FILE
   000446  000461 2210 03                  LDX     1,TRAP3,DU     TRAP ADDRESS
   000447  000414 2220 00                  LDX     2,TRAP1        FRN OF DUMPFILE
   000450  000000 2230 03                  LDX     3,0,DU         TO: FILE LOCATION
   000451  000000 2240 03                  LDX     4,0,DU         FROM: CORE LOCATION
   000452  000055 5500 00                  SBAR    SBAR           GET CORE-SIZE
   000453  000055 2250 00                  LDX     5,SBAR         LOAD NUMBER OF ELEMENTS
END OF BINARY CARD LPCP0007
   000454  000377 3650 03                  ANX5    =0377,DU       MASK TO NUMBER OF ELEMENTS
   000455  000000 0010 00                  MME                    INITIATE THE COPY
                                   *
                                   *       PAUSE TILL DUMP IS DONE
                                   *
   000456  000017 2200 03                  LDX     0,$.PAUSE,DU   ALWAYS PAUSE
   000457  000000 0010 00                  MME                    WAIT
   000460  000456 7100 00                  TRA     *-2            FOREVER
                        000461     TRAP3   BSS     3              COPY TO DUMP FILE
   000464  000461 7200 00                  LXL     0,TRAP3        GET STATUS
   000465  000470 6000 00                  TZE     TERM           NOW IT IS TIME TO TERMINATE
   000466  000003 1000 03                  CMPX    0,BSBZ,DU      WAS THE EXEC TOO BUSY
   000467  000445 6000 00                  TZE     FT3            YES, SO JUST RETRY
```

X DIAGNOSTICS

```
                          000470      838 TERM   BSS      0                    TIME TO SAY SO LONG
000470  000016 2200 03                839        LDX      0,$,TERM,DU          TERMINATE
000471  000000 0010 00                840        MME                           BYE-BYE
000472  000470 7100 00                841        TRA      *-2                  TAKE NO CHANCES
```

X EXIT MACRO

```
000473          USE     CODE
                HEAD
       *
       *
       *                                EXIT
       *
       *        EXIT TERMINATES A THREAD OF CONTROL BY RETURNING TO THE
       *        TASK DISTRIBUTOR.
       *
       EXIT     MACRO                   NO ARGUMENTS
                TRA     $EXIT
                ENDM    EXIT
```

BUGGING MACROS

```
000473          USE     CODE
                HEAD
      *
      *
      *         BUGGING MACROS PLANT ADDRESSES IN INVALID DATA AREAS SO THAT
      *         ANY UNAUTHORIZED USE OF SUCH DATA WILL RESULT IN A MEMORY FAULT
      *         OR EXECUTIVE CALL REJECT.
      *
      *
      *                                         BUG
      *
      *         BUG FILLS BOTH UPPER AND LOWER HALVES OF A STORAGE WORD WITH
      *         THE BUG PATTERN $BUGBUG.
      *
      BUG       MACRO   STORAGE-ADDRESS
                IFE     $DBG,$ON,4
      BUGBUG    SET     BUGBUG+1
                LDX     0,BUGBUG,DU
                STX     0,#1
                SXL     0,#1
                ENDM    BUG
      *
      *
      *                                         BUGU
      *
      *         BUGU FILLS THE UPPER HALF OF A STORAGE WORD WITH THE BUG
      *         PATTERN $BUGBUG
      *
      BUGU      MACRO   STORAGE-ADDRESS
                IFE     $DBG,$ON,3
      BUGBUG    SET     BUGBUG+1
                LDX     0,BUGBUG,DU
                STX     0,#1
                ENDM    BUGU
      *
      *
      *                                         BUGL
      *
      *         BUGL FILL THE LOWER HALF OF A STORAGE WORD WITH THE BUG
      *         PATTERN $BUGBUG.
      *
      BUGL      MACRO   STORAGE-ADDRESS
                IFE     $DBG,$ON,3
      BUGBUG    SET     BUGBUG+1
                LDX     0,BUGBUG,DU
                SXL     0,#1
                ENDM    BUGL
```

BUGGING MACROS

```
*
*
*                                       BUGXR
*
*       BUGXR LOADS THE SPECIFIED INDEX REGISTER(S) WITH THE
*
BUGXR   MACRO   INDEX-REGISTER(S)
        IFE     $DBG,$ON,4
BUGBUG  SET     BUGBUG+1
        IDRP    #1
        LDX     #1,BUGBUG,DU
        IDRP
        ENDM    BUGXR
*
*
*                                       BUGA
*
*       BUGA LOADS THE CONTENTS OF THE THE A REGISTER WITH THE BUG
*       PATTERN $BUGBUG.
*
BUGA    MACRO   <NO ARGUMENTS>
        IFE     $DBG,$ON,3
BUGBUG  SET     BUGBUG+1
        LDA     BUGBUG,DU
        ORA     BUGBUG,DL
        ENDM    BUGA
*
*
*                                       BUGQ
*
*       BUGQ LOADS THE CONTENTS OF THE Q REGISTER WITH THE BUG
*       PATTERN $BUGBUG.
*
BUGQ    MACRO   <NO ARGUMENTS>
        IFE     $DBG,$ON,3
BUGBUG  SET     BUGBUG+1
        LDQ     BUGBUG,DU
        ORQ     BUGBUG,DL
        ENDM    BUGQ
*
*
*                                       DECRM MACRO
*
*       DECREMENT A COUNTER
*
DECRM   MACRO   COUNTER-ADDRESS
        LCQ     1,DL
        ASQ     #1
        ENDM    DECRM
```

CHECKPOINT MACRO

```
                   000473          USE     CODE
                                   HEAD    X
                             *
                             *
                             *                               CHECKPOINTS
                             *
                             *     THIS MARCRO CAUSES THE REGISTERS TO BE STORED IN 8-WORD
                             *     BLOCKS IN A CIRCULAR QUEUE FOR DEBUGGING USE.  INFORMATION
                             *     IS STORED IN THE FOLLOWING FORMAT:
                             *
                             *
                             *     C(0) =  C(X0) (UPPER)
                             *             C(X1) (LOWER)
                             *     C(1) =  C(X2) (UPPER)
                             *             C(X3) (LOWER)
                             *     C(2) =  C(X4) (UPPER)
                             *             C(X5) (LOWER)
                             *     C(3) =  C(X6) (UPPER)
                             *             C(X7) (LOWER)
                             *     C(4) =  C(A)
                             *     C(5) =  C(Q)
                             *     C(6) =  C(E) (0-7 BITS)
                             *     C(7) =  C(TR) (0-23 BITS)
                             *
                             *
                             *                               CKPT
                             *
                             CKPT  MACRO   <NO ARGUMENTS>
                                   IFE     $DBG,$ON,1
                                   XED     X$CKPT
                                   ENDM    CKPT
                             *
                             *
                             *             CKPT -- SUBROUTINE
                             *
                   000474          EVEN
                   000474    CKPT  BSS     0               ENTRY POINT
 000474  003324 5540 00            STC1    CKIC            SAVE IC
 000475  000476 7100 00            TRA     *+1             BREAK XED
 000476  003330 7530 00            SREG    CKREG           SAVE REGISTERS FOR A RELOAD
 000477  000160 7530 51            SREG    REG,I           SAVE IN 8-WORD BLOCK
                             *
                             *     UPDATE POINTER FOR CIRCULAR QUEUE
                             *
 000500  000160 2200 00            LDX     0,REG           GET CURRENT ADDRESS
 000501  000010 0200 03            ADLX    0,8,DU          BUMP TO NEXT ENTRY
 000502  000370 1000 03            CMPX    0,DBGQ+8*DBGQN,DU  OVER THE END?
END OF BINARY CARD LPCP0008
 000503  000505 6020 00            TNC     *+2             NO, THIS IS VALID
 000504  000170 2200 03            LDX     0,DBGQ,DU       YES, RESET TO BEGINNING
```

```
         X                          CHECKPOINT MACRO

000505  000160 7400 00        1003          STX     0,REG        SAVE FOR NEXT TIME
000506  003330 0730 00        1004          LREG    CKREG        RESTORE REGISTERS
000507  003324 6300 00        1005          RET     CKIC         RESTORE IC
                              1006 *
                              1007 *
                    003324    1008          USE     STORE
                    003324    1009 CKIC     BSS     1            TEMP STORAGE FOR IC/IR
                    003330    1010          EIGHT
                    003330    1011 CKREG    BSS     8            TEMP STORAGE FOR REGISTERS
                    000510    1012          USE     PREVIOUS
```

```
        000510            USE     CODE
                          HEAD
                   *
                   *
                   *                        SETUP MACRO
                   *
                   SETUP  MACRO             NO ARGUMENTS
                          XED     $SETUP
                          ENDM    SETUP
                   *
                   *
                   *              SETUP -- SUBROUTINE TO SET UP A TRAP
                   *
                   *      CALL WITH
                   *              C(XT) = TBLOCK-ADDRESS
                   *              C(XJ) = JBLOCK ADDRESS
                   *              C(X0) = TRANSFER ADDRESS FOR J$TRA
                   *      ENTER BY
                   *              XED T$SETUP
                   *      DESTROYS C(A), C(Q), C(X0)
                   *      USES NO TEMPORARIES
                   *
                   *
        000510            EVEN
        000510     SETUP  BSS     0
000510  000004 7400 11    STX     0,T$TRA,T         SET T$TRA = RESTART ADDRESS
000511  000512 7000 00    TSX     0,*+1             BREAK XED
000512  000000 4310 03    FLD     0,DU              ZERO OUT A AND Q
000513  000000 7570 11    STAQ    T$SRW1,T          ZERO STATUS WORDS
000514  000520 2370 00    LDAQ    TRAP-1            GET ZERO, XED WORDS
000515  000002 7570 11    STAQ    T$XED-1,T         SAVE ZERO, XED
000516  000000 7100 10    TRA     0,0               RETURN
                   *
                   *
                   *              TRAP -- XED SEQUENCE TO PUT BLOCK ON Q$TASK
                   *
        000520            EVEN
000520  000000 000000     ZERO                      CAN BE USED FOR CLEARING RET WORDS
000521  000522 7170 00     TRAP   XED     *+1       THIS IS EXECUTED FROM THE TBLOCK
000522  003261 5540 54    STC1    Q$LAST+Q$TASK,DI  *UPDATE PREVIOUS LAST POINTER
000523  000524 7170 00    XED     *+1               CONTINUE WITHOUT AFFECTING IC
000524  003261 5540 00    STC1    Q$LAST+Q$TASK     UPDATE POINTER TO LAST
000525  777777 6300 04    RET     -1,IC             RETURN TO POINT OF INTERRUPTION
```

```
000526        USE     CODE
              HEAD
        *
        *
        *
        *     ALL SYSTEM CALLS ARE DONE THRU PRE-DEFINED MACROS.  THOSE
        *     MACROS ARE LISTED IN THE FOLLOWING PAGES.  EACH OF THESE
        *     MACROS IS CODED TO SPECIFIC CONVENTIONS:
        *     ENTER BY
        *             TSX  0,$<MACRO-NAME>
        *     ENTERED WITH
        *             C(XT) = TBLOCK-ADDRESS
        *             C(XJ) = JBLOCK-ADDRESS
        *     CALLS
        *             SETUP
        *     ISSUES MME AND THEN 'EXITS'
        *     RETURNS TO FIRST LOCATION AFTER MACRO
        *     RETURNS WITH
        *             C(XT) = TBLOCK-ADDRESS
        *             C(XJ) = JBLOCK-ADDRESS
        *             C(XL) = RESTART-ADDRESS
        *
        *     IN ACTUALITY, THE FOLLOWING HAPPENS:  THE MACRO DOES
        *     DO THE 'TSX0'.  HOWEVER, FOLLOWING THAT INSTRUCTION IS
        *     THE SET OF ARGUMENTS FOR THE MME.  THE SUBROUTINE ENTERED
        *     WHICH ALWAYS HAS THE SAME NAME AS THE MACRO KNOWS THE
        *     NUMBER OF ARGUMENTS IT IS PASSED.  THEREFORE IT SIMPLY CAL-
        *     CULATES THE RESTART ADDRESS.  IT THEN CALLS 'SETUP' WHICH
        *     RESETS THE TRAP BLOCK (C(XT) = X1) WITH THE FIRST THREE WORDS
        *     BEING ZEROED AND THE LINK WORD FOR THE EXEC SET TO THE
        *     XED INSTRUCTION TO PLACE THE BLOCK ON THE QSTASK QUEUE.  HENCE
        *     WHEN TRAPPED, THE BLOCK WILL BE LINKED ON THE QSTASK QUEUE
        *     AND THE INTERRUPT WILL BE TRANSPARENT TO THE CURRENTLY RUN-
        *     NING TASK UNLESS THE CURRENT TASK IS 'PAUSE'.  THE PARAMETERS
        *     ARE THEN FETCHED (SEE PROGRAMMER'S REFERENCE MANUAL IF YOU
        *     DON'T FULLY UNDERSTAND 'IDC' MODIFICATIONS; CAUSE IF YOU DON'T
        *     YOU WILL BE LOST) AND THE MME EXECUTED.  'EXIT' IS CALLED
        *     THUS PLACING THE CURRENTLY RUNNING TASK IN A BLOCKED STATE
        *     WHILE STARTING THE NEXT READY-TO-RUN TASK.  THIS IS HOW WE
        *     MULTI-PROGRAM.  WHEN THE EXEC TRAPS THIS OPERATION, AS
        *     WE HAVE SAID, THE XED INSTRUCTION PLACES THE TASK BACK
        *     ON THE QSTASK QUEUE.
```

SETUP FAULT VECTOR MACRO

```
                          000526   1101          USE     CODE
                                   1102          HEAD
                                   1103 *
                                   1104 *
                                   1105 *                           SETFV
                                   1106 *
                                   1107 SETFV    MACRO   CORELOC
                                   1108          TSX     0,$SETFV
                                   1109          ARG     #1              ADDRESS OF SLAVE FAULT VECTOR
                                   1110          ENDM    SETFV
                                   1111 *
                                   1112 *
                                   1113 *                SETFV -- SUBROUTINE
                                   1114 *
                                   1115 *        THIS SUBROUTINE IS CALLED BY THE SETFV MACRO.  IT ISSUES THE
                                   1116 *        COMMAND TO LOCATE THE SLAVE FAULT VECTOR.
                                   1117 *
                                   1118 *        CALL WITH
                                   1119 *                C(XT) = TBLOCK-ADDRESS
                                   1120 *                C(XJ) = JBLOCK ADDRESS
                                   1121 *        ENTER BY
                                   1122 *                TSX 0,$SETFV
                                   1123 *                ARG CORE LOCATION TO FAULT VECTOR
                                   1124 *        RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                                   1125 *        RETURNS WITH
                                   1126 *                C(XT) = TBLOCK-ADDRESS
                                   1127 *                C(XJ) = JBLOCK ADDRESS
                                   1128 *                C(XL) = RESTART ADDRESS
                                   1129 *        USES LOCAL TEMPORARY ONLY
                                   1130 *
                                   1131 *
END OF BINARY CARD LPCP0009
    000526  003340 7400 00         1132 SETFV    STX     0,SETFT         POINTER TO ARGUMENT LIST
    000527  000001 0200 03         1133          ADLX    0,1,DU          RESTART ADDRESS
                          000530   1134          SETUP
    000530  000510 7170 00                       XED     $SETUP
    000531  003340 2220 57         1135          LDX     2,SETFT,IDC     LOAD ADDRESS OF FAULT VECTOR
    000532  000001 2200 03         1136          LDX     0,.SETFV,DU     LOAD MME NUMBER
                          000533   1137          CKPT                    CHECKPOINT
    000533  000474 7170 00                       XED     X$CKPT
    000534  000000 0010 00         1138          MME                     SETUP FAULT VECTOR
                          000535   1139          EXIT
    000535  001547 7100 00                       TRA     $EXIT
                                   1140 *
                          003340   1141          USE     STORE
    003340  000000 0000 20         1142 SETFT    ARG     0,*             POINTER TO ARGUMENT LIST
                          000536   1143          USE     PREVIOUS
```

READ MACRO

```
                   000536     1145        USE     CODE
                              1146        HEAD
                              1147 *
                              1148 *
                              1149 *                               READ
                              1150 *
                              1151 READ    MACRO   FRN,CORELOC,N,MODE
                              1152        TSX     0,$READ
                              1153        ARG     #1              FRN ADDRESS
                              1154        ARG     #2              ADDRESS OF CORE LOC
                              1155        ARG     #3              NUMBER OF ELEMENTS
                              1156        ARG     #4              MODE
                              1157        ENDM    READ
                              1158 *
                              1159 *
                              1160 *              READ -- SUBROUTINE
                              1161 *
                              1162 *      THIS SUBROUTINE IS CALLED BY THE READ MACRO.  IT ISSUES THE
                              1163 *      COMMAND TO READ THE NEXT N ELEMENTS OF FRN IN A PARTICULAR MODE.
                              1164 *
                              1165 *      CALL WITH
                              1166 *              C(XT) = TBLOCK-ADDRESS
                              1167 *              C(XJ) = JBLOCK-ADDRESS
                              1168 *      ENTER BY
                              1169 *              TSX 0,$READ
                              1170 *              ARG ADDRESS OF FRN
                              1171 *              ARG ADDRESS OF CORELOC
                              1172 *              ARG N
                              1173 *              ARG MODE
                              1174 *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                              1175 *      RETURNS WITH
                              1176 *              C(XT) = TBLOCK-ADDRESS
                              1177 *              C(XJ) = JBLOCK-ADDRESS
                              1178 *              C(XL) = RESTART-ADDRESS
                              1179 *      USES LOCAL TEMPORARY ONLY
                              1180 *
                              1181 *
000536  003341 7400 00        1182 READ   STX     0,READT         POINTER TO ARGUMENT LIST
000537  000004 0200 03        1183        ADLX    0,4,DU          RESTART ADDRESS
                   000540     1184        SETUP
000540  000510 7170 00                    XED     $SETUP
000541  003341 2220 57        1185        LDX     2,READT,IDC     LOAD FRN
000542  003341 2240 57        1186        LDX     4,READT,IDC     LOAD CORE LOC
000543  003341 2250 57        1187        LDX     5,READT,IDC     LOAD N
000544  003341 2260 57        1188        LDX     6,READT,IDC     LOAD MODE
000545  000004 2200 03        1189        LDX     0,.READ,DU      LOAD MME NUMBER
                   000546     1190        CKPT                    CHECKPOINT
000546  000474 7170 00                    XED     X$CKPT
000547  000000 0010 00        1191        MME                     READ
                   000550     1192        EXIT
```

READ MACRO

```
     000550  001547 7100 00                   TRA      $EXIT
                                   1193 *
                          003341   1194       USE      STORE
  END OF BINARY CARD LPCP0010
     003341  000000 0000 20        1195 READT ARG      0,*            POINTER TO ARGUMENT LIST
                          000551   1196       USE      PREVIOUS
```

```
                     000551                 USE     CODE
                                            HEAD
                                     *
                                     *
                                     *                              APEND
                                     *
                                     APEND  MACRO   FRN,CORELOC,N,MODE
                                            TSX     0,$APEND
                                            ARG     #1              FRN ADDRESS
                                            ARG     #2              ADDRESS OF CORE LOC
                                            ARG     #3              NUMBER OF ELEMENTS
                                            ARG     #4              MODE
                                            ENDM    APEND
                                     *
                                     *
                                     *              APEND -- SUBROUTINE
                                     *
                                     *      THIS SUBROUTINE IS CALLED BY THE APEND MACRO.  IT ISSUES
                                     *      THE COMMAND TO APEND N ELEMENTS TO THE FRN SPECIFIED
                                     *      VIA THE SPECIFIED MODE.
                                     *
                                     *      CALL WITH
                                     *              C(XT) = TBLOCK-ADDRESS
                                     *              C(XJ) = JBLOCK-ADDRESS
                                     *      ENTER BY
                                     *              TSX 0,$APEND
                                     *              ARG FRN ADDRSS
                                     *              ARG ADDRESS OF CORE LOC
                                     *              ARG NUMBER OF ELEMENTS
                                     *              ARG MODE
                                     *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                                     *      RETURNS WITH
                                     *              C(XT) = TBLOCK-ADDRESS
                                     *              C(XJ) = JBLOCK ADDRESS
                                     *              C(XL) = RESTART ADDRESS
                                     *      USES LOCAL TEMPORARY ONLY
                                     *
                                     *
000551  003342 7400 00               APEND  STX     0,APNDT         POINTER TO ARGUMENT LIST
000552  000004 0200 03                      ADLX    0,4,DU          RESTART ADDRESS
                     000553                 SETUP
000553  000510 7170 00                      XED     $SETUP
000554  003342 2220 57                      LDX     2,APNDT,IDC     LOAD FRN
000555  003342 2240 57                      LDX     4,APNDT,IDC     LOAD CORE LOC
000556  003342 2250 57                      LDX     5,APNDT,IDC     LOAD NUMBER OF ELEMENTS
000557  003342 2260 57                      LDX     6,APNDT,IDC     LOAD MODE
000560  000005 2200 03                      LDX     0,.APEND,DU     LOAD MME NUMBER
                     000561                 CKPT                    CHECKPOINT
000561  000474 7170 00                      XED     X$CKPT
000562  000000 0010 00                      MME                     APEND
```

APEND MACRO

```
                      000563    1246         EXIT
000563  001547 7100 00                       TRA     $EXIT
                                1247 *
                      003342    1248         USE     STORE
003342  000000 0000 20          1249 APNDT   ARG     0,*            POINTER TO ARGUMENT LIST
                      000564    1250         USE     PREVIOUS
```

SET POINTER MACRO

```
                    000564     1252          USE      CODE
                               1253          HEAD
                               1254 *
                               1255 *
                               1256 *                            SET POINTER
                               1257 *
                               1258 SPTR     MACRO    FRN,N
                               1259          TSX      0,$SPTR
                               1260          ARG      #1              FRN ADDRESS
                               1261          ARG      #2              NUMBER OF ELEMENTS TO MOVE POINTER
                               1262          ENDM     SPTR
                               1263 *
                               1264 *
                               1265 *                 SET POINTER -- SUBROUTINE
                               1266 *
                               1267 *        THIS SUBROUTINE IS CALLED BY THE  SPTR  MACRO.  IT ISSUES
                               1268 *        THE COMMAND TO ADD (OR SUBTRACT) N ELEMENTS TO THE CUR-
                               1269 *        RENT SETTING OF THE READ POINTER.
                               1270 *
                               1271 *        CALL WITH
                               1272 *                 C(XT) = TBLOCK-ADDRESS
                               1273 *                 C(XJ) = JBLOCK-ADDRESS
                               1274 *        ENTER BY
                               1275 *                 TSX 0,$SPTR
                               1276 *                 ARG FRN
                               1277 *                 ARG NUMBER OF ELEMENTS
                               1278 *        RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                               1279 *        RETURNS WITH
                               1280 *                 C(XJ) = JCB
                               1281 *                 C(XT) = TCB
                               1282 *        USES LOCAL TEMPORARY ONLY
                               1283 *
  000564  003343 7400 00       1284 SPTR     STX      0,SPTRT         POINTER TO ARGUMENT LIST
  000565  000002 0200 03       1285          ADLX     0,2,DU          RESTART ADDRESS
                    000566     1286          SETUP
  000566  000510 7170 00                     XED      $SETUP
  000567  003343 2220 57       1287          LDX      2,SPTRT,IDC     LOAD FRN
  000570  003343 2230 57       1288          LDX      3,SPTRT,IDC     LOAD N
  000571  000011 2200 03       1289          LDX      0,,SPTR,DU      LOAD MME NUMBER
                    000572     1290          CKPT                     CHECKPOINT
END OF BINARY CARD LPCP0011
  000572  000474 7170 00                     XED      X$CKPT
  000573  000000 0010 00       1291          MME                      SET POINTER
                    000574     1292          EXIT
  000574  001547 7100 00                     TRA      $EXIT
                               1293 *
                    003343     1294          USE      STORE
  003343  000000 0000 20       1295 SPTRT    ARG      0,*             POINTER TO ARGUMENT LIST
                    000575     1296          USE      PREVIOUS
```

REQUEST STATUS MACRO

```
                     000575              USE      CODE
                                         HEAD
                                   *
                                   *
                                   *                                REQUEST STATUS
                                   *
                                   RQST  MACRO    FRN
                                         TSX      0,$RQST
                                         ARG      #1                FRN ADDRESS
                                         ENDM     RQST
                                   *
                                   *
                                   *              REQUEST STATUS -- SUBROUTINE
                                   *
                                   *     THIS SUBROUTINE IS CALLED BY THE  RQST  MACRO.  IT ISSUES
                                   *     THE COMMAND TO REQUEST STATUS ON THE FRN SPECIFIED.
                                   *
                                   *     CALL WITH
                                   *              C(XT) = TCB
                                   *              C(XJ) = JCB
                                   *     ENTER BY
                                   *              TSX 0,$RQST
                                   *              ARG FRN
                                   *     RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                                   *     RETURNS WITH
                                   *              C(XT) = TCB
                                   *              C(XJ) = JCB
                                   *     USES LOCAL TEMPORARY ONLY
                                   *
                                   *
000575  003344 7400 00             RQST  STX      0,RQSTT           POINTER TO ARGUMENT LIST
000576  000001 0200 03                   ADLX     0,1,DU            RESTART ADDRESS
                     000577              SETUP
000577  000510 7170 00                   XED      $SETUP
000600  003344 2220 57                   LDX      2,RQSTT,IDC       LOAD FRN
000601  000012 2200 03                   LDX      0,.RQST,DU        LOAD MME NUMBER
                     000602              CKPT                       CHECKPOINT
000602  000474 7170 00                   XED      X$CKPT
000603  000000 0010 00                   MME                        REQUEST STATUS
                     000604              EXIT
000604  001547 7100 00                   TRA      $EXIT
                                   *
                     003344              USE      STORE
003344  000000 0000 20             RQSTT ARG      0,*               POINTER TO ARGUMENT LIST
                     000605              USE      PREVIOUS
```

CHANGE SEGMENT MACRO

```
                   000605               USE     CODE
                                        HEAD
                                *
                                *
                                *                               CHSEG
                                *
                                CHSEG   MACRO   SEGMENT-NUMBER,LENGTH
                                        TSX     0,$CHSEG
                                        ARG     #1              NUMBER-OF-SEGMENT
                                        ARG     #2              NEW-LENGTH
                                        ENDM    CHSEG
                                *
                                *
                                *               CHSEG -- SUBROUTINE
                                *
                                *       THIS SUBROUTINE IS CALLED BY THE CHSEG MACRO.  IT ISSUES
                                *       THE COMMAND TO CHANGE THE LENGTH OF THE NAMED SEGMENT TO
                                *       THE NEW LENGTH SPECIFIED.
                                *
                                *       CALL WITH
                                *               C(XT) = TBLOCK-ADDRESS
                                *               C(XJ) = JBLOCK ADDRESS
                                *       ENTER BY
                                *               TSX 0,$CHSEG
                                *               ARG SEGMENT-NUMBER
                                *               ARG LENGTH
                                *       RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                                *       RETURNS WITH
                                *               C(XT) = TBLOCK-ADDRESS
                                *               C(XJ) = JBLOCK ADDRESS
                                *       USES LOCAL TEMPORARY ONLY
                                *
  000605  003345 7400 00        CHSEG   STX     0,CHSGT         POINTER TO ARGUMENT LIST
  000606  000002 0200 03                ADLX    0,2,DU          RESTART ADDRESS
                   000607               SETUP
  000607  000510 7170 00                XED     $SETUP
  000610  003345 2220 57                LDX     2,CHSGT,IDC     LOAD SEGMENT NUMBER
  000611  003345 2230 57                LDX     3,CHSGT,IDC     LOAD SEGMENT LENGTH
END OF BINARY CARD LPCP0012
  000612  000022 2200 03                LDX     0,.CHSEG,DU     LOAD MME NUMBER
                   000613               CKPT                    CHECKPOINT
  000613  000474 7170 00                XED     X$CKPT
  000614  000000 0010 00                MME                     CHANGE SEGMENT
                   000615               EXIT
  000615  001547 7100 00                TRA     $EXIT
                                *
                   003345               USE     STORE
  003345  000000 0000 20        CHSGT   ARG     0,*             POINTER TO ARGUMENT LIST
                   000616               USE     PREVIOUS
```

CLOSE MACRO

```
                    000616              USE     CODE
                                        HEAD
                                 *
                                 *
                                 *                              CLOSE
                                 *
                                 CLOSE  MACRO   FRN
                                        TSX     0,$CLOSE
                                        ARG     #1              FILE REFERENCE ADDRESS
                                        ENDM    CLOS
                                 *
                                 *
                                 *              CLOSE -- SUBROUTINE
                                 *
                                 *      THIS SUBROUTINE IS CALLED BY THE CLOS MACRO.  IT ISSUES THE
                                 *      MME TO CLOSE A FILE.
                                 *
                                 *      CALL WITH
                                 *              C(XT) = TBLOCK-ADDRESS
                                 *              C(XJ) = JBLOCK ADDRESS
                                 *      ENTER BY
                                 *              TSX 0,$CLOS
                                 *              ARG FILE-REFERENCE-ADDRESS
                                 *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                                 *      RETURNS WITH
                                 *              C(XT) = TBLOCK-ADDRESS
                                 *              C(XJ) = JBLOCK ADDRESS
                                 *              C(XL) = RESTART ADDRESS
                                 *      USES LOCAL TEMPORARY ONLY
                                 *
000616  003346 7400 00           CLOSE  STX     0,CLOST         POINTER TO ARGUMENT LIST
000617  000001 0200 03                  ADLX    0,1,DU          RESTART ADDRESS
                    000620              SETUP
000620  000510 7170 00                  XED     $SETUP
000621  003346 2220 57                  LDX     2,CLOST,IDC     LOAD FILE REFERENCE
000622  000025 2200 03                  LDX     0,.CLOSE,DU     LOAD MME NUMBER
                    000623              CKPT                    CHECKPOINT
000623  000474 7170 00                  XED     X$CKPT
000624  000000 0010 00                  MME                     CLOS
                    000625              EXIT
000625  001547 7100 00                  TRA     $EXIT
                                 *
                    003346              USE     STORE
003346  000000 0000 20           CLOST  ARG     0,*             POINTER TO ARGUMENT LIST
                    000626              USE     PREVIOUS
```

LOCK MACRO

```
                          000626   1430        USE     CODE
                                   1431        HEAD
                                   1432 *
                                   1433 *
                                   1434 *                           LOCK
                                   1435 *
                                   1436 LOCK   MACRO   FRN
                                   1437        TSX     0,$LOCK
                                   1438        ARG     #1            FILE-REFERENCE ADDRESS
                                   1439        ENDM    LOCK
                                   1440 *
                                   1441 *
                                   1442 *              LOCK -- SUBROUTINE
                                   1443 *
                                   1444 *      THIS SUBROUTINE IS CALLED BY THE LOCK MACRO.  IT ISSUES THE
                                   1445 *      MME TO LOCK A FILE.
                                   1446 *
                                   1447 *      CALL WITH
                                   1448 *              C(XT) = TBLOCK-ADDRESS
                                   1449 *              C(XJ) = JBLOCK ADDRESS
                                   1450 *      ENTER BY
                                   1451 *              TSX 0,$LOCK
                                   1452 *              ARG FILE-REFERENCE-ADDRESS
                                   1453 *      RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                                   1454 *      RETURNS WITH
                                   1455 *              C(XT) = TBLOCK-ADDRESS
                                   1456 *              C(XJ) = JBLOCK ADDRESS
                                   1457 *              C(XL) = RESTART ADDRESS
                                   1458 *      USES LOCAL TEMPORARY ONLY
                                   1459 *
     000626  003347 7400 00        1460 LOCK   STX     0,LOCKT       POINTER TO ARGUMENT LIST
     000627  000001 0200 03        1461        ADLX    0,1,DU        RESTART ADDRESS
                          000630   1462        SETUP
     000630  000510 7170 00                    XED     $SETUP
     000631  003347 2220 57        1463        LDX     2,LOCKT,IDC   LOAD FILE REFERENCE
END OF BINARY CARD LPCP0013
     000632  000044 2200 03        1464        LDX     0,.LOCK,DU    MME NUMBER
                          000633   1465        CKPT                  CHECKPOINT
     000633  000474 7170 00                    XED     X$CKPT
     000634  000000 0010 00        1466        MME                   LOCK
                          000635   1467        EXIT
     000635  001547 7100 00                    TRA     $EXIT
                                   1468 *
                          003347   1469        USE     STORE
     003347  000000 0000 20        1470 LOCKT  ARG     0,*           POINTER TO ARGUMENT LIST
                          000636   1471        USE     PREVIOUS
```

UNLOCK MACRO

```
                    000636     1473         USE     CODE
                               1474         HEAD
                               1475 *
                               1476 *
                               1477 *                          UNLOCK
                               1478 *
                               1479 UNLCK   MACRO   FRN
                               1480         TSX     0,$UNLCK
                               1481         ARG     #1              FILE REFERENCE ADDRESS
                               1482         ENDM    UNLK
                               1483 *
                               1484 *
                               1485 *               UNLOCK -- SUBROUTINE
                               1486 *
                               1487 *       THIS SUBROUTINE IS CALLED BY THE UNLK MACRO.  IT ISSUES THE
                               1488 *       MME TO UNLOCK A FILE.
                               1489 *
                               1490 *       CALL WITH
                               1491 *               C(XT) = TBLOCK-ADDRESS
                               1492 *               C(XJ) = JBLOCK ADDRESS
                               1493 *       ENTER BY
                               1494 *               TSX 0,$UNLK
                               1495 *               ARG FILE-REFERENCE-ADDRESS
                               1496 *       RETURNS TO FIRST LOC AFTER THE MACRO EXPANSION
                               1497 *       RETURNS WITH
                               1498 *               C(XT) = TBLOCK-ADDRESS
                               1499 *               C(XJ) = JBLOCK ADDRESS
                               1500 *               C(XL) = RESTART ADDRESS
                               1501 *       USES LOCAL TEMPORARY ONLY
                               1502 *
000636  003350 7400 00         1503 UNLCK   STX     0,UNLKT         POINTER TO ARGUMENT LIST
000637  000001 0200 03         1504         ADLX    0,1,DU          RESTART ADDRESS
                    000640     1505         SETUP
000640  000510 7170 00                      XED     $SETUP
000641  003350 2220 57         1506         LDX     2,UNLKT,IDC     LOAD FILE REFERENCE
000642  000045 2200 03         1507         LDX     0,.UNLCK,DU     MME NUMBER
                    000643     1508         CKPT                    CHECKPOINT
000643  000474 7170 00                      XED     X$CKPT
000644  000000 0010 00         1509         MME                     UNLOCK
                    000645     1510         EXIT
000645  001547 7100 00                      TRA     $EXIT
                               1511 *
                    003350     1512         USE     STORE
003350  000000 0000 20         1513 UNLKT   ARG     0,*             POINTER TO ARGUMENT LIST
                    000646     1514         USE     PREVIOUS
```

NOTIFY MACRO

```
                      000646     1516         USE     CODE
                                 1517         HEAD
                                 1518 *
                                 1519 *                                NOTIFY
                                 1520 *
                                 1521 NOTIF   MACRO   ERN,STATE
                                 1522         TSX     0,$NOTIF
                                 1523         ARG     #1               FRN ADDRESS
                                 1524         ARG     #2               STATE ADDRESS
                                 1525         ENDM    NOTIF
                                 1526 *
                                 1527 *
                                 1528 *               NOTIFY -- SUBROUTINE
                                 1529 *
                                 1530 *       THIS SUBROUTINE IS CALLED BY THE  NOTIF  MACRO.  IT ISSUES
                                 1531 *       THE NOTIFY ON THE SPECIFIED EVENT.  NOTE THAT C(X3) POINT TO
                                 1532 *       CTRAP.
                                 1533 *
                                 1534 *       CALL WITH
                                 1535 *               C(XT) = TBLOCK-ADDRESS
                                 1536 *               C(XJ) = JBLOCK ADDRESS
                                 1537 *               C(X3) = CTRAP ADDRESS
                                 1538 *       ENTER BY
                                 1539 *               TSX 0,NOTIF
                                 1540 *               ARG EVENT-FRN
                                 1541 *               ARG STATE ADDRESS
                                 1542 *       RETURNS TO FIRST LOC AFTER MACRO EXPANSION
                                 1543 *       RETURNS WITH
                                 1544 *               C(XT) = TBLOCK-ADDRESS
                                 1545 *               C(XJ) = JBLOCK ADDRESS
                                 1546 *               C(XL) = RESTART ADDRESS
                                 1547 *       USES LOCAL TEMPORARY ONLY
                                 1548 *
   000646  003351 7400 00        1549 NOTIF   STX     0,NOTFT          POINTER OF ARGUMENT LIST
   000647  000002 0200 03        1550         ADLX    0,2,DU           RESTART ADDRESS
                      000650     1551         SETUP
   000650  000510 7170 00                     XED     $SETUP
   000651  003351 2220 57        1552         LDX     2,NOTFT,IDC      LOAD EVENT FRN
 END OF BINARY CARD LPCP0014
   000652  003351 2350 57        1553         LDA     NOTFT,IDC        LOAD STATE
   000653  000046 2200 03        1554         LDX     0,,NOTIF,DU      LOAD MME NUMBER
                      000654     1555         CKPT                     CHECKPOINT
   000654  000474 7170 00                     XED     X$CKPT
   000655  000000 0010 00        1556         MME                      NOTIFY
                      000656     1557         EXIT
   000656  001547 7100 00                     TRA     $EXIT
                                 1558 *
                      003351     1559         USE     STORE
   003351  000000 0000 20        1560 NOTFT   ARG     0,*              POINTER TO ARGUMENT LIST
                      000657     1561         USE     PREVIOUS
```

CAUSE MACRO

```
                        000657            USE     CODE
                                          HEAD
                                  *
                                  *
                                  *                                CAUSE
                                  *
                                  CAUSE   MACRO   ERN,NUMBER,STATE,MESSAGE,ACCESSES,FRN (N.B. ORDERING)
                                          TSX     0,$CAUSE
                                          ARG     #1               FILE REFERENCE ADDRESS
                                          ARG     #2               NUMBER WHICH ARE TO NOTIFIED
                                          ARG     #3               STATE
                                          ARG     #4               MESSAGE
                                          ARG     #5               ACCESSES ON PASSED ITEM
                                          ARG     #6               FRN OF ITEM TO PASS
                                          ENDM    CAUS
                                  *
                                  *
                                  *               CAUSE -- SUBROUTINE
                                  *
                                  *       THIS SUBROUTINE IS CALLED BY THE CAUS MACRO.  IT ISSUES THE
                                  *       MME TO CAUSE THE SPECIFIED FILE.
                                  *
                                  *       CALL WITH
                                  *               C(XJ) = JBLOCK ADDRESS
                                  *               C(XT) = TBLOCK-ADDRESS
                                  *       ENTER BY
                                  *               TSX 0,$CAUS
                                  *               ARG FILE-REFERENCE-NUMBER OF EVENT
                                  *               ARG NUMBER
                                  *               ARG PASSED-FILE-REFERENCE
                                  *               ARG ACCESSES-ON-PASSED-ITEMS
                                  *               ARG STATE
                                  *               ARG MESSAGE
                                  *       RETURNS TO FIRST LOC AFTER THE MACRO EXPANSION
                                  *       RETURNS WITH
                                  *               C(XJ) = JBLOCK ADDRESS
                                  *               C(XT) = TBLOCK-ADDRESS
                                  *               C(XL) = RESTART ADDRESS
                                  *       USES LOCAL TEMPORARY ONLY
                                  *
000657  003352 7400 00            CAUSE   STX     0,CAUST          POINTER TO ARGUMENT LIST
000660  000006 0200 03                    ADLX    0,6,DU           RESTART ADDRESS
                000661                    SETUP
000661  000510 7170 00                    XED     $SETUP
000662  003352 2220 57                    LDX     2,CAUST,IDC      LOAD FILE REFERENCE
000663  003352 7230 57                    LXL     3,CAUST,IDC      NUMBER
000664  003352 2350 57                    LDA     CAUST,IDC        STATE
000665  003352 2360 57                    LDQ     CAUST,IDC        MESSAGE
000666  003352 7270 57                    LXL     7,CAUST,IDC      LOAD ACCESSES
000667  003352 2260 57                    LDX     6,CAUST,IDC      LOAD FRN -- CLOBBER XR-J NOW
```

CAUSE MACRO

```
     000670  000047 2200 03       1612         LDX      0,,CAUSE,DU       MME NUMBER
                    000671        1613         CKPT                       CHECKPOINT
     000671  000474 7170 00                    XED      X$CKPT
     000672  000000 0010 00       1614         MME                        CAUSE
                    000673        1615         EXIT
     000673  001547 7100 00                    TRA      $EXIT
                                  1616 *
                    003352        1617         USE      STORE
  END OF BINARY CARD LPCP0015
     003352  000000 0000 20       1618 CAUST   ARG      0,*               POINTER TO ARGUMENT LIST
                    000674        1619         USE      PREVIOUS
```

CHECK MACRO

```
000674                USE      CODE
                      HEAD
           *
           *
           *                                  CHECK
           *
           *          THIS MACRO IS USED TO CHECK THE RESULT OF A TRAPPING MME.
           *          THE FIRST ARGUMENT IS THE ADDRESS TO WHICH TO TRANSFER IF THE
           *          STATUS RETURN IS ZERO.  THIS ARGUMENT MAY BE OMITTED, AND NO
           *          TEST WILL BE ASSEMBLED.  THE REMAINING ARGUMENTS COME IN PAIRS.
           *          THE FIRST OF THE PAIR IS A BOOLEAN PATTERN AGAINST WHICH A
           *          COMPARISON WILL BE MADE.  THE SECOND IS THE TRANSFER ADDRESS IN
           *          CASE OF A MATCH.  CURRENTLY THERE MAY BE ONLY 8 SUCH PAIRS.
           *
           *
           CHECK      MACRO    ZEROSTATADD,BOOLPAT,XFERADD,BOOLPAT,XFERADD,ETC.
                      LXL      0,T$SRW1,T       PICK UP LOGICAL STATUS
                      ANX      0,B$STMK,DU      ISOLATE STATUS
                      INE      '#1','',
                      TZE      #1               ZERO STATUS TEST
                      INE      '#2','',23
                      CMPX     0,#2,DU
                      TZE      #3               FIRST PAIR OF TESTS
                      INE      '#4','',20
                      CMPX     0,#4,DU
                      TZE      #5               SECOND PAIR OF TESTS
                      INE      '#6','',17
                      CMPX     0,#6,DU
                      TZE      #7               THIRD PAIR OF TESTS
                      INE      '#8','',14
                      CMPX     0,#8,DU
                      TZE      #9               FOURTH PAIR OF TESTS
                      INE      '#10','',11
                      CMPX     0,#10,DU
                      TZE      #11              FIFTH PAIR OF TESTS
                      INE      '#12','',8
                      CMPX     0,#12,DU
                      TZE      #13              SIXTH PAIR OF TESTS
                      INE      '#14','',5
                      CMPX     0,#14,DU
                      TZE      #15              SEVENTH PAIR OF TESTS
                      INE      '#16','',2
                      CMPX     0,#16,DU
                      TZE      #17              EIGHTH PAIR OF TESTS
                      TRA      $ERROR           DIE ON UNEXPECTED RETURN
                      ENDM     CHECK
```

```
000674           USE     CODE
                 HEAD    Q
            *
            *            QUEUE MANAGEMENT -- GENERAL INTRODUCTION
            *
            *    EACH QUEUE IN THE PROGRAM HAS A SIMILAR STRUCTURE.  A QUEUE
            *    CONSISTS OF A (POSSIBLY EMPTY) LINKED LIST OF BLOCK.  THE
            *    POINTERS POINT TO WORD 4 (Q$OFFST) OF A BLOCK.  THE LINK
            *    POINTERS ARE STORED IN WORD 3 (Q$LINK) OF A BLOCK.  THE WORD AT
            *    LOCATION Q$FIRST POINTS TO Q$OFFST OF THE FIRST BLOCK OF THE
            *    QUEUE.  THE LOCATION Q$LAST POINTS TO Q$OFFST OF THE LAST BLOCK
            *    OF THE QUEUE.  THE EMPTY QUEUE IS DENOTED BY THE WORD AT Q$LAST
            *    POINTING TO Q$FIRST+1.
            *
            *
            *                                QUEUE
            *
            *    THIS GENERATES A QBLOCK.  THIS STRUCTURE MUST AGREE
            *    WITH THE STRUCTURE DEFINED FOR QUEUE MANAGEMENT.
            *
            QUEUE    MACRO   QBLOCK-LOCATION-SYMBOL,ASCII-NAME
                     USE     QSTOR               PUT ALL QUEUES CONTIQUOUS
                     EVEN                        FOR XED
            #1       BSS     0                   NAME OF QUEUE
                     ARG     $ERROR              FIRST
                     ARG     Q$#1+Q$FIRST+1      LAST
                     STX     0,Q$#1+Q$LAST,DI    *XADD
                     STX     0,Q$#1+Q$LAST
                     EAX     Q,Q$#1              ENQ
                     TSX     L,Q$ENQ
                     EAX     Q,Q$#1              DEQ
                     TSX     L,Q$DEQ
                     ARG     0                   BUSY
                     DEC     0                   MAX
                     DEC     0                   AVAIL
                     DEC     0                   SPARE1
                     DEC     0                   SPARE2
                     UASCI   1,#1                ABBREVIATION
                     ENDM    QUEUE
```

```
                        000674    1708          USE     CODE
                                  1709          HEAD    Q
                                  1710 *
                                  1711 *
                                  1712 *                                ENQ
                                  1713 *
                                  1714 *        ENQ SUSPENDS A TASK UNTIL THE SPECIFIED QUEUE CAN BE MADE
                                  1715 *        AVAILABLE.
                                  1716 *
                                  1717 ENQ      MACRO   QADDRESS
                                  1718          XED     Q$#1+Q$XENQ
                                  1719          ENDM    ENQ
                                  1720 *
                                  1721 *                ENQ -- SUBROUTINE TO SERIALIZE RESOURCE USE
                                  1722 *
                                  1723 *        THIS SUBPROGRAM RETURNS IMMEDIATELY IF THERE IS NO NEED
                                  1724 *        TO QUEUE.  IF NECESSARY TO DO SO, IT SUSPENDS EXECUTION OF
                                  1725 *        THE CURRENT TASK UNTIL A WAKE-UP IS GENERATED BY A DEQ FOR
                                  1726 *        THIS QUEUE AND THE RESOURCE CAN BE ALLOCATED TO THIS TASK.
                                  1727 *
                                  1728 *        CALL WITH
                                  1729 *                C(XT) = TBLOCK-ADDRESS
                                  1730 *                C(XJ) = JBLOCK-ADDRESS
                                  1731 *                C(XQ) = QBLOCK-ADDRESS
                                  1732 *        ENTER BY
                                  1733 *                TSX L,Q$ENQ
                                  1734 *        CLOBBERS ALL BUT C(XT), C(XJ), AND C(XL)
                                  1735 *        USES NO LOCAL TEMPORARIES
                                  1736 *
                                  1737          INHIB   SAVE,ON
                        000674    1738 ENQ      BSS     0
000674  000012 0542 15            1739          AOS     AVAIL,Q          ONE MORE BLOCK CURRENTLY ON QUEUE
000675  000012 2352 15            1740          LDA     AVAIL,Q          UPDATE MAX LENGTH OF QUEUE
000676  000011 1152 15            1741          CMPA    MAX,Q            .
000677  000701 6022 00            1742          TNC     *+2              .
000700  000011 7552 15            1743          STA     MAX,Q            .
000701  000010 2342 15            1744          SZN     BUSY,Q           IS THIS QUEUE BUSY?
000702  000716 6012 00            1745          TNZ     ENQ1             YES, WILL HAVE TO QUEUE FOR IT
000703  000010 7412 15            1746          STX     T,BUSY,Q         NO, MARK WHO IS RESPONSIBLE
                                  1747          INHIB   RESTORE
                        000704    1748          BUGA
                        525201         BUGBUG SET     BUGBUG+1
000704  525201 2350 03                          LDA     BUGBUG,DU
000705  525201 2750 07                          ORA     BUGBUG,DL
                        000706    1749          BUGQ
                        525202         BUGBUG SET     BUGBUG+1
000706  525202 2360 03                          LDQ     BUGBUG,DU
000707  525202 2760 07                          ORQ     BUGBUG,DL
                        000710    1750          BUGXR   (0,X,Y,Z,Q)
                        525203         BUGBUG SET     BUGBUG+1
```

```
          Q                                  QUEUE MANAGEMENT -- ENQ

    000710  525203 2200 03                  LDX     0,BUGBUG,DU
    000711  525203 2220 03                  LDX     X,BUGBUG,DU
    000712  525203 2230 03                  LDX     Y,BUGBUG,DU
    000713  525203 2240 03                  LDX     Z,BUGBUG,DU
    000714  525203 2250 03                  LDX     Q,BUGBUG,DU
    000715  000000 7100 17       1751       TRA     0,L              RETURN TO CALLER
    000716  000004 7470 11       1752 ENQ1  STX     L,TSTRA,T        SAVE RESTART ADDRESS
    000717  000004 6200 11       1753       EAX     0,OFFST,T        STORE POINTER WITH OFFSET
 END OF BINARY CARD LPCP0016
    000720  777777 6000 00       1754       TZE     $ERROR           ***DBG
    000721  777777 6040 00       1755       TMI     $ERROR           ***DBG
    000722  000002 7170 15       1756       XED     XADD,Q           IN QUEUE LINKED LIST OF BLOCKS
                      000723     1757       EXIT
    000723  001547 7100 00                  TRA     $EXIT
```

```
                 000724         USE     CODE
                                HEAD    Q
                        *
                        *                                DEQ
                        *
                        *       DEQ RELEASES A SERIALLY REUSABLE RESOURCE SO THAT OTHER
                        *       PROCESSES MAY USE IT.  OTHER TASKS ARE AUTOMATICALLY INITIATED
                        *       BY PUTTING THEM ON THE Q$TASK QUEUE.
                        *
                        DEQ     MACRO   QADDRESS
                                XED     Q$#1+Q$XDEQ
                                ENDM    DEQ
                        *
                        *               DEQ -- SUBROUTINE TO WAKE-UP WAITING TASKS
                        *
                        *       THIS SUBROUTINE CHECKS TO SEE IF ANY OTHER TASKS ARE ASLEEP
                        *       IN THE QUEUE.  IF THE QUEUE IS NOT EMPTY, A TBLOCK IS TAKEN
                        *       FROM IT AND PUT ON THE Q$TASK QUEUE TO WAKE IT UP.  THIS SUB-
                        *       PROGRAM ALWAYS RETURNS TO THE CALLER IMMEDIATELY.
                        *
                        *       CALL WITH
                        *               C(XQ) = QBLOCK-ADDRESS
                        *               C(XT) = TBLOCK-ADDRESS
                        *               C(XJ) = JBLOCK-ADDRESS
                        *       ENTER BY
                        *               TSX L,Q$DEQ
                        *       DESTROYS C(X0), C(XX)
                        *       USES NO LOCAL TEMPORARIES
                        *
                                INHIB   SAVE,ON
                 000724 DEQ     BSS     0
                 000724         DECRM   (AVAIL,Q)       ONE LESS BLOCK ON QUEUE
 000724  000001 3362 07         LCQ     1,DL
 000725  000012 0562 15         ASQ     AVAIL,Q
 000726  000010 2342 15         SZN     BUSY,Q          IF QUEUE IS NOT BUSY
 000727  777777 6002 00         TZE     $ERROR          WE ARE IN BAD TROUBLE
 000730  000010 4502 15         STZ     BUSY,Q          MARK QUEUE NOT BUSY
 000731  000001 6202 15         EAX     0,FIRST+1,Q     ADDRESS OF FIRST ELEMENT
 000732  000001 1002 15         CMPX    0,LAST,Q        DOES LAST POINT TO IT?
 000733  000000 6002 17         TZE     0,L             YES, QUEUE EMPTY -- RETURN
 000734  000000 2202 15         LDX     0,FIRST,Q       GET OFFSET POINTER TO BLOCK
 000735  777777 6002 00         TZE     $ERROR          ***BLEWIT
 000736  777777 6042 00         TMI     $ERROR          ***DBG
 000737  777774 6222 10         EAX     X,-OFFST,0      RELATE TO THE BEGINNING OF BLOCK
 000740  000010 7402 15         STX     0,BUSY,Q        REMEMBER WHO IS RESPONSIBLE
 000741  000003 2222 12         LDX     X,LINK,X        GET NEXT ELEMENT ON QUEUE
 000742  000000 7422 15         STX     X,FIRST,Q       NOW MAKE IT FIRST
 000743  003262 7172 00         XED     XADD+TASK       ADD TO TASK QUEUE
 000744  000001 1002 15         CMPX    0,LAST,Q        LAST BLOCK ON QUEUE?
END OF BINARY CARD LPCP0017
```

```
000745  000000 6012 17     1806     TNZ     0,L              NO, RETURN
000746  000001 6202 15     1807     EAX     0,FIRST+1,Q      YES, SET UP QUEUE TO APPEAR EMPTY
000747  000001 7402 15     1808     STX     0,LAST,Q         BY MAKING LAST POINT TO FIRST+1
                           1809     INHIB   RESTORE
000750  000000 7100 17     1810     TRA     0,L              RETURN
```

```
000751         USE     CODE
               HEAD    Q
        *
        *
        *                              BRANCH MACRO
        *
        *      THIS MACRO CREATES AN ASYNCHRONOUS TASK TO BE PREFORMED AT A
        *      LATER TIME.  IT CAN GIVE THE CREATED TASK THE CURRENT TBLOCK
        *      OR GIVE IT A NEW TBLOCK.  EITHER WAY FOUR PARAMETERS MAY BE
        *      BETWEEN THE TWO TBLOCKS.
        *
        *      PAST INFORMATION IS PLACED IN TEMP1 THRU TEMP4 OF
        *      THE TASK PLACED ON THE Q$TASK QUEUE.
        *
        *      CALLS
        *              T$GETT
        *      CLOBBERS C(XX), C(X0)
        *
        *
        BRANCH MACRO   PASS,XFER ADD,C(TEMP1),C(TEMP2),C(TEMP3),C(TEMP4)
               TSX     0,T$GETT        GET A NEW TBLOCK
               EAX     X,0,T           C(XX) POINTS TO NEW TBLOCK
               LDX     T,T$LINK,X      C(XT) POINTS TO OLD TBLOCK
               INE     '#3','',2
               LDQ     #3              SAVE FIRST PARAMETER
               STQ     T$TEMP1,X
               INE     '#4','',2
               LDQ     #4              SAVE SECOND PARAMETER
               STQ     T$TEMP2,X
               INE     '#5','',2
               LDQ     #5              SAVE THIRD PARAMETER
               STQ     T$TEMP3,X
               INE     '#6','',2
               LDQ     #6              SAVE FOURTH PARAMETER
               STQ     T$TEMP4,X
               INE     '#1','PASS',1   T IS THE BLOCK THAT IS PASSED
               EAX     T,0,X           PASS NEW BLOCK
               EAX     0,#2            POINT TO TRANSFER ADDRESS
               STX     0,T$TRA,T       INTO QUEUE BLOCK
               EAX     0,Q$OFFST,T     PREPARE TO QUEUE
               XED     Q$XADD+Q$TASK   GET ON THE TASK QUEUE
               IFE     '#1','PASS',1   WHICH BLOCK TO WE GIVE BACK AS CURRENT
               EAX     T,0,X           GIVE BACK NEW BLOCK
               INE     '#1','PASS',1
               LDX     T,T$LINK,X      NO, GIVE BACK OLD BLOCK
               BUGXR   (0,X)
               ENDM    BRANCH
```

```
000751          USE      CODE
                HEAD     Q
          *
          *
          *                                  DIJKSTRA DESIGN
          *
          *           DR. E.W. DIJKSTRA IN HIS PAPER ON 'COOPERATING
          *      SEQUENTIAL PROCESSES' EXPLAINS A METHODOLOGY FOR
          *      SYNCHRONIZING ASYNCHRONOUS TASKS BY MEANS OF 'SEMAPHORES'
          *      AND TWO PRIMITIVES--THE 'P-OPERATION' AND THE 'V-OPERATION'.
          *      THESE PRIMITIVES OPERATE UPON SEMAPHORES AND REPRESENT
          *      THE ONLY WAY IN WHICH THE CONCURRENT PROCESSES (TASKS) MAY
          *      ACCESS THE SEMAPHORES.
          *
          *           DEFINITION:  THE V-OPERATION IS AN OPERATION WITH ONE
          *      ARGUMENT, WHICH MUST BE THE IDENTIFICATION OF A SEMAPHORE.
          *      ITS FUNCTION IS TO INCREASE THE VALUE OF ITS ARGUMENT
          *      SEMAPHORE BY 1 AND AWAKEN ANY TASK WAITING ON THAT SEMAPHORE.
          *      THIS INCREASE AND AWAKENING IS TO BE REGARDED AS AN INDIVIS-
          *      IBLE OPERATION.
          *
          *           DEFINITIONS: THE P-OPERATION IS AN OPERATION WITH ONE
          *      ARGUMENT, WHICH MUST BE THE IDENTIFICATION OF A SEMAPHORE.
          *      ITS FUNCTION IS TO DECREASE THE VALUE OF ITS ARGUMENT SEMA-
          *      PHORE AND IF THE ARGUMENT SEMAPHORE VALUE SHOULD GO NEGATIVE
          *      THE TASK DOING THE P-OPERATION IS BLOCKED WAITING FOR
          *      THE SEMAPHORE TO GO NON-NEGATIVE.  THIS DECREASE AND POTENTIAL
          *      BLOCKING IS TO BE REGARDED AS AN INDIVISIBLE OPERATION.
          *
          *           IT IS THE P-OPERATION, WHICH REPRESENTS THE POTENTIAL
          *      DELAY, VIZ. WHEN A PROCESS INITIATES A P-OPERATION ON A SEMA-
          *      PHORE, THAT AT THAT MOMENT IS = 0, IN THIS CASE THE TASK
          *      EXECUTING THE P-OPERATION IS BLOCKED AND CAN NOT BE RESTARTED
          *      UNTIL A V-OPERATION IS DONE ON THE SAME SEMAPHORE.
```

```
                  000751        USE     CODE
                                HEAD    Q
                         *
                         *
                         *                              P (DOWN)
                         *
                         *      P DECREMENTS A SEMAPHORE COUNT AND IF THE COUNT SHOULD
                         *      GO NEGATIVE SUSPENDS THE CURRENT TASK UNTIL THE SEMAPHORE
                         *      IS RAISED.
                         *
                         P      MACRO   SEMAPHORE-ADDRESS
                                EAX     Q,#1            MAKE Q POINT TO SEMAPHORE
                                TSX     L,Q$P           CALL SUBROUTINE
                                ENDM    P
                         *
                         *
                         *              P -- SUBROUTINE
                         *
                         *      THIS SOURBOUTINE RETURNS IMMEDIATELY IF THERE IS NO NEED
                         *      TO QUEUE.  IF NECESSARY TO DO SO, IT SUSPENDS EXECUTION OF
                         *      THE CURRENT TASK UNTIL A WAKE-UP IS GENERATED BY A V (UP)
                         *      FOR THIS QUEUE AND THE RESOURCE CAN BE ALLOCATED TO THIS TASK.
                         *
                         *      CALL WITH
                         *              C(XQ) = SEMAPHORE-ADDRESS
                         *              C(XT) = TBLOCK-ADDRESS
                         *              C(XL) = RETURN-ADDRESS
                         *      ENTER BY
                         *              TSX L,Q$P
                         *      DESTROYS ALL BUT C(XT), C(XJ), AND C(XL)
                         *      USES NO LOCAL TEMPORARIES
                         *
                                INHIB   SAVE,ON
                  000751 P      BSS     0
000751  000000 7242 15          LXL     Z,0,Q           GET SEMAPHORE
000752  000001 1242 03          SBLX    Z,1,DU          LET SEM:=SEM-1
000753  000000 4442 15          SXL     Z,0,Q           AND SAVE IT
000754  000767 6022 00          TNC     P1              WILL HE HAVE TO HANG?
                  000755        BUGA                    NO, BUG REGISTERS
                  525204 BUGBUG SET     BUGBUG+1
000755  525204 2352 03          LDA     BUGBUG,DU
000756  525204 2752 07          ORA     BUGBUG,DL
                  000757        BUGQ                    AND RETURN
                  525205 BUGBUG SET     BUGBUG+1
000757  525205 2362 03          LDQ     BUGBUG,DU
000760  525205 2762 07          ORQ     BUGBUG,DL
                  000761        BUGXR   (0,X,Y,Z,Q)
                  525206 BUGBUG SET     BUGBUG+1
000761  525206 2202 03          LDX     0,BUGBUG,DU
000762  525206 2222 03          LDX     X,BUGBUG,DU
```

Q QUEUE MANAGEMENT -- P (DOWN)

```
000763  525206 2232 03                    LDX     Y,BUGBUG,DU
000764  525206 2242 03                    LDX     Z,BUGBUG,DU
000765  525206 2252 03                    LDX     Q,BUGBUG,DU
000766  000000 7102 17         1936       TRA     0,L
000767  000004 7472 11         1937 P1    STX     L,TSTRA,T          SAVE RESTART ADDRESS
000770  000000 7412 15         1938       STX     T,0,Q              STORE POINTER TO TBLOCK
                    000771     1939       EXIT
000771  001547 7102 00                    TRA     $EXIT
                               1940       INHIB   RESTORE
```

```
                     000772            USE     CODE
                                       HEAD    Q
                                *
                                *
                                *                              V (UP)
                                *
                                *      V INCREMENTS A SEMAPHORE AND RESTARTS ANY TASK WAITING
                                *      ON THAT SEMAPHORE.
                                *
                                V      MACRO   SEMAPHORE-ADDRESS
                                       EAX     Q,#1            MAKE Q POINT TO SEMAPHORE
                                       TSX     L,Q$V           CALL SUBROUTINE
                                       ENDM    V
                                *
                                *
                                *              V -- SUBROUTINE
                                *
                                *      THIS SUBROUTINE INCREMENTS THE SEMAPHORE FLAG.  IF THE COUNT
                                *      IS STILL LESS THAN OR EQUAL TO ZERO, THEN THE SUBROUTINE
                                *      TAKES THE SLEEPING TASK AND WAKES HIM UP BY PLACING THE
                                *      TASK ON THE Q$TASK QUEUE.  THIS SUBROUTINE ALWAYS RETURNS
                                *      TO THE CALLER IMMEDIATELY.
                                *
                                *      BY DESIGN, NO MORE THAN ONE TASK EVER WAITS ON A
                                *      PARTICULAR RESOURCE.  SAVVY??
                                *
                                *      CALL WITH
                                *              C(XQ) = SEMAPHORE-ADDRESS
                                *              C(XT) = TBLOCK-ADDRESS
                                *              C(XL) = RETURN-ADDRESS
                                *      ENTER BY
                                *              TSX L,Q$V
                                *      DESTROYS C(X0), C(XX)
                                *      USES NO LOCAL TEMPORARIES
                                *
                                       INHIB   SAVE,ON
                     000772     V      BSS     0
END OF BINARY CARD LPCP0018
000772  000000 7222 15                 LXL     X,0,Q           GET SEMAPHORE
000773  000001 0222 03                 ADLX    X,1,DU          SEM:=SEM+1
000774  000000 4422 15                 SXL     X,0,Q           SAVE IT
000775  000777 6002 00                 TZE     *+2             IF SEM<=0, THEN AWAKEN
000776  000000 6052 17                 TPL     0,L             NO, JUST RETURN TO CALLER
000777  000000 2202 15                 LDX     0,0,Q           GET POINTER TO TBLOCK
001000  000004 6202 10                 EAX     0,OFFST,0       GET OFFSET POINTER
001001  777777 6002 00                 TZE     $ERROR          ***DBG
001002  777777 6042 00                 TMI     $ERROR          ***DBG
001003  003262 7172 00                 XED     XADD+TASK       ADD IT TO TASK QUEUE
                     001004            BUGU    (0,Q)           BUG IT
                     525207     BUGBUG SET     BUGBUG+1
```

```
         G                                  QUEUE MANAGEMENT -- V (UP)

001004  525207 2202 03                        LDX     0,BUGBUG,DU
001005  000000 7402 15                        STX     0,0,Q
                    001006    1990            BUGXR   (0,X)
                    525210           BUGBUG   SET     BUGBUG+1
001006  525210 2202 03                        LDX     0,BUGBUG,DU
001007  525210 2222 03                        LDX     X,BUGBUG,DU
001010  000000 7102 17        1991            TRA     0,L             RETURN TO CALLER
                              1992            INHIB   RESTORE
```

```
                                        1994          HEAD    Q
                                        1995 *
                                        1996 *
                                        1997 *                                     Q$TASK QUEUE
                                        1998 *
                                        1999 *        THIS QUEUE IS USED TO SCHEDULE THE ACTIVITY OF THE PROCESSOR
                                        2000 *        ENTRIES ARE MADE BY THE USE OF THE NORMAL XED Q$ADD
                                        2001 *        FEATURE AND ALSO BY AN XED CHAIN INSTIGATED BY THE ACTION OF
                                        2002 *        A TRAP BEING SPRUNG BY THE EXECUTIVE.  ENTRIES ARE REMOVED BY
                                        2003 *        A SPECIAL PROGRAM MODULE WHICH IS CALLED BY THE EXIT MACRO
                                        2004 *        WHENEVER THE PROCESSOR IS FREE TO WORK ON A NEW TASK.
                                        2005 *
                             001011     2006          QUEUE   TASK
                             003260                   USE     QSTOR
                             003260                   EVEN
                             003260           TASK    BSS     0
      003260  777777 0000 00                          ARG     $ERROR
      003261  003261 0000 00                          ARG     Q$TASK+Q$FIRST+1
      003262  003261 7400 54                          STX     0,Q$TASK+Q$LAST,DI
      003263  003261 7400 00                          STX     0,Q$TASK+Q$LAST
      003264  003260 6250 00                          EAX     Q,Q$TASK
      003265  000674 7070 00                          TSX     L,Q$ENQ
   END OF BINARY CARD LPCP0019
      003266  003260 6250 00                          EAX     Q,Q$TASK
      003267  000724 7070 00                          TSX     L,Q$DEQ
      003270  000000 0000 00                          ARG     0
      003271  000000000000                            DEC     0
      003272  000000000000                            DEC     0
      003273  000000000000                            DEC     0
      003274  000000000000                            DEC     0
      003275  124101123113                            UASCI   1,TASK              ABBREVIATION
```

```
                                       2008          HEAD    Q
                                       2009 *
                                       2010 *
                                       2011 *                                  Q$CORE QUEUE
                                       2012 *
                                       2013 *        THIS QUEUE HOLDS THOSE TASKS WHICH REQUIRE MEMORY ALLOCATIONS
                                       2014 *        WHEN A PRIOR MEMORY REQUEST IS IN OPERATION.  THIS IS NOT
                                       2015 *        TO BE CONFUSED WITH THE THE FREE MEMORY LIST.
                                       2016 *
                            003276     2017          QUEUE   CORE
                            003276                   USE     QSTOR
                            003276                   EVEN
                            003276          CORE     BSS     0
    003276  777777 0000 00                           ARG     $ERROR
    003277  003277 0000 00                           ARG     Q$CORE+Q$FIRST+1
    003300  003277 7400 54                           STX     0,Q$CORE+Q$LAST,DI
    003301  003277 7400 00                           STX     0,Q$CORE+Q$LAST
    003302  003276 6250 00                           EAX     Q,Q$CORE
    003303  000674 7070 00                           TSX     L,Q$ENQ
    003304  003276 6250 00                           EAX     Q,Q$CORE
    003305  000724 7070 00                           TSX     L,Q$DEQ
    003306  000000 0000 00                           ARG     0
    003307  000000000000                             DEC     0
    003310  000000000000                             DEC     0
    003311  000000000000                             DEC     0
    003312  000000000000                             DEC     0
 END OF BINARY CARD LPCP0020
    003313  103117122105                             UASCI   1,CORE             ABBREVIATION
```

```
                 001011                USE      CODE
                                       HEAD     R
                                 *
                                 *                         GENERAL INTRODUCTION
                                 *
                                 *     BELOW IS THE FREE MEMORY LIST.  THE LIST CONSISTS OF A
                                 *     POSSIBLY EMPTY LINKED LIST OF BLOCKS.  THE FORWARD/BACKWARD
                                 *     POINTERS OF A BLOCK POINT TO THE FIRST WORD OF THE SUCCEEDING/
                                 *     PRECEEDING BLOCKS, RESPECTIVELY. THE LINK POINTERS ARE
                                 *     STORED IN WORDS 0 AND 1, RESPECTIVELY, OF THE BLOCK.
                                 *     HENCE THE MINIMAL SIZE OF A BLOCK IS TWO (2) WORDS.  THE
                                 *     TOTAL LENGTH OF THE BLOCK IS ALSO KEPT IN WORD 0.  BY
                                 *     DESIGN CONVENTIONS, THE POINTERS ARE UPPER HALF QUANTITIES
                                 *     AND THE LENGTH IS A LOWER HALF QUANTITY.  THE EMPTY LIST
                                 *     IS DENOTED BY THE FORWARD LINK OF THE 'FIRST' POINTING TO
                                 *     'LAST' AND BY THE BACKWARD LINK OF 'LAST' POINTING TO 'FIRST'.
                                 *
                                 *
                                 *     NOTE THAT ALL ALLOCATIONS ARE DONE IN MULTIPLES OF EIGHT.
                                 *     A BLOCK MUST BE DE-ALLOCATED AS ONE ENTITY.  NO PARTIAL RELEASES
                                 *     ARE ALLOWED.
                                 *
                                 *              FREE MEMORY LIST
                                 *
                 003353                USE      STORE
003353  003770 000000             FIRST ZERO    $NEXTF,0         FORWARD LINK/ LENGTH OF BLOCK
003354  000000 000000                   ZERO    0,               BACKWARD LINK/ <NOT USED>
                                 *
                                 *
003355  000000 000000             LAST ZERO     0,0              FORWARD LINK/ LENGTH OF BLOCK
003356  003770 000000                   ZERO    $NEXTB,          BACKWARD LINK/ <NOT USED>
                 001011                USE      PREVIOUS
```

```
001011          USE     CODE
                HEAD    R
      *
      *
      *                                  MACROS
      *
      *
      *         THE FOLLOWING MACROS ARE USED TO ALLOCATE/DEALLOCATE BLOCKS
      *         OF CORE.  ONLY INDEX REGISTER J IS GUARANTEED ACCROSS THESE
      *         CALLS.
      *
      *                                  GETC MACRO
      *
      *         CALL WITH
      *                 C(AL) = NUMBER-OF-WORDS-REQUESTED
      *         ENTER BY
      *                 TSX L.R$GETC
      *         RETURNS TO 0.L
      *         RETURNS WITH
      *                 C(AU) = BUFFER-ADDRESS
      *                 C(AL) = BUFFER-LENGTH
      *
       GETC     MACRO   WORD-COUNT-ADDRESS/ 'A'
                INE     '#1','A'
                LDA     #1
                TSX     L.R$GETC
                ENDM    GETC
      *
      *
      *                                  RELC MACRO
      *
      *         CALL WITH
      *                 C(AU) = BUFFER-ADDRESS
      *                 C(AL) = BUFFER-ADDRESS
      *         ENTER BY
      *                 TSX L.R$RELC
      *         RETURNS TO 0.L
      *         RETURNS WITH
      *                 DESTROYS C(A)
      *
       RELC     MACRO   RELEASE-ADDRESS&COUNT/ 'A'
                INE     '#1','A'
                LDA     #1
                TSX     L.R$RELC
                ENDM    RELC
```

```
                    001011      2098         USE     CODE
                                2099         HEAD    R
                                2100 *
                                2101 *
                                2102 *                               ALLOCATION
                                2103 *
                                2104 *       THIS SUBROUTINE REMOVES A BLOCK OF N WORDS FROM THE FREE
                                2105 *       MEMORY LIST.  IF THERE IS NO BLOCK OF N WORDS OR GREATER
                                2106 *       ON THE FREE LIST. THERE A REQUEST FOR MORE MEMORY IS MADE.
                                2107 *       AND THE PROCESS IS REPEATED.  SINCE MEMORY REQUESTS ARE
                                2108 *       OF THE TRAPPING MME BRAND. IT IS NECESSARY TO FIRST
                                2109 *       CHECK TO SEE IF A MEMORY REQUEST IS CURRENTLY IN OPERATION.
                                2110 *       IF SO. THEN THIS REQUEST IS PUT TO SLEEP BY QUEUEING IT
                                2111 *       ON THE Q$CORE QUEUE.  WHEN THE MEMORY REQUEST COMPLETES
                                2112 *       THE NEXT ITEM (IF ANY) ON Q$CORE IS AWAKENED.  CORE ALLO-
                                2113 *       CATION IS ON A FIRST FIT BASIS.  THIS IS TO ALLOW HOLES
                                2114 *       TO FLOW TOWARD HIGHER MEMORY.
                                2115 *
                                2116 *       CALL BY
                                2117 *               TSX L.R$GETC
                                2118 *       CALL WITH
                                2119 *               C(AL) = NUMBER-OF-WORDS-REQUESTED
                                2120 *               C(T)  = TRAP BLOCK
                                2121 *       CALLS
                                2122 *               ENQ CORE
                                2123 *               R$MORE
                                2124 *               R$RELC
                                2125 *       DEQ     CORE
                                2126 *       RETURNS WITH
                                2127 *               C(AU) = ADDRESS
                                2128 *               C(AL) = NUMBER OF WORDS
                                2129 *       EXIT TO 0.L
                                2130 *       PRESERVES ALL REGISTERS EXCEPT C(A)
                                2131 *       USES TEMPORARIES T$TEM9 THRU T$TEM16
                                2132 *
                    001011      2133         USE     CODE
                    001011      2134 GETC    BSS     0
001011  000010 7530 11          2135         SREG    T$TEM16.T       SAVE ALL REGISTERS
                    001012      2136         ENQ     CORE            AND GET ON THE CORE QUEUE
001012  003302 7170 00                       XED     Q$CORE+Q$XENQ
                    001013      2137 GETC1   BSS     0
001013  000014 2350 11          2138         LDA     T$TEM12.T       RESTORE A AND
001014  000007 0350 07          2139         ADLA    7.DL            ROUND UP LENGTH
001015  777770 3750 07          2140         ANA     =0777770.DL     TO A MULTIPLE OF 8
001016  777777 6000 00          2141         TZE     $ERROR          ***BLEWIT
001017  004001 1150 07          2142         CMPA    $RQMAX.DL       HOW MUCH IS REQUESTED
001020  777777 6030 00          2143         TRC     $ERROR          TOO MUCH!
001021  000014 7550 11          2144         STA     T$TEM12.T       SAVE NEW LENGTH
001022  000000 6350 05          2145         EAA     0.AL            MOVE LENGTH TO AU
001023  000016 7550 11          2146         STA     T$TEM10.T       SAVE FOR INDEX REGISTER OPERATIONS
```

```
                   001024     2147 GETC3  BSS     0
001024  000014 2350 11        2148        LDA     T$TEM12,T         GET BACK REQUEST
001025  003353 2240 00        2149        LDX     Z,FIRST           GET PTR TO FIRST FREE BLOCK
                   001026     2150 GETC6  BSS     0
001026  000302 5002 00        2151        RPL     ,TNC,TZE          RUN DOWN LINKED LIST
001027  000000 1150 14        2152        CMPA    0,Z                  FOR A BLOCK BIG ENOUGH
END OF BINARY CARD LPCP0021
001030  001035 6000 00        2153        TZE     GETC7             FOUND A BLOCK THAT IS 'JUST RIGHT'
001031  001035 6020 00        2154        TNC     GETC7             MORE THAT BIG ENOUGH
001032  000000 2240 14        2155        LDX     Z,0,Z             GET BACK PTR OF WHERE WE LEFT OFF
001033  001026 6010 00        2156        TNZ     GETC6             IF NOT AT END OF LIST, KEEP SEARCHING
001034  001077 7100 00        2157        TRA     MORE              WON'T FIT ANYWHERE, GET MORE CORE
001035  000014 7440 11        2158 GETC7  STX     Z,T$TEM12,T       SAVE PTR TO BLOCK TO RETURN TO CALLER
                              2159 *
                              2160 *       Z POINTS TO BLOCK TO DELINK
                              2161 *
001036  000001 2230 14        2162        LDX     Y,LINKB,Z         ASSIGN Y TO PREDECESSOR
001037  000000 2220 14        2163        LDX     X,LINKF,Z         AND X TO SUCCESSORE
001040  000000 7200 14        2164        LXL     0,LEN,Z           GET THE LENGTH OF THIS BLOCK
001041  000016 1200 11        2165        SBLX    0,T$TEM10,T       MINUS THE AMOUNT REQUIRED
001042  001051 6000 00        2166        TZE     GETC4             AH HA, BLOCK JUST FITS
001043  777777 6040 00        2167        TMI     $ERROR            ***BLEWIT
001044  000016 0240 11        2168        ADLX    Z,T$TEM10,T       SET Z TO START OF EXCESS OF BLOCK
001045  000000 4400 14        2169        SXL     0,LEN,Z           SET THE LENGTH OF THE REMAINING BLOCK
001046  000000 7420 14        2170        STX     X,LINKF,Z         SET FORWARD LINK OF REMAINING BLOCK
001047  000001 7440 12        2171        STX     Z,LINKB,X          "  BACKWARD  "  "      "      "
001050  000000 6220 14        2172        EAX     X,0,Z             COPY Z INTO X FOR FUDGE
001051  000000 7420 13        2173 GETC4  STX     X,LINKF,Y         SET FORWARD LINK OF PREDECESSOR
001052  000001 4500 12        2174        STZ     LINKB,X           ***DBG
001053  000001 7430 12        2175        STX     Y,LINKB,X         SET BACKWARD LINK OF SELF
                   001054     2176        DEQ     CORE              SEE IF ANYONE ELSE WANTS CORE
001054  003304 7170 00                    XED     Q$CORE+Q$XDEQ
                              2177 *
                              2178 *       ZERO BLOCK FOR USER
                              2179 *
001055  000014 2350 11        2180        LDA     T$TEM12,T         GET LENGTH                         .
END OF BINARY CARD LPCP0022
001056  777777 3750 07        2181        ANA     -1,DL             MASK TO COUNT
001057  777777 6240 05        2182        EAX     Z,-1,AL           SAVE COUNT MINUS ONE IN Z
001060  000010 7350 00        2183        ALS     10-2              PUT COUNT IN REPEAT FIELD
001061  001400 6200 05        2184        EAX     0,B$ABIT+B$BBIT,AL  TERMINATE CONDITIONS AND COUNT
001062  000000 4310 07        2185        FLD     0,DL              CLEAR AQ
001063  000014 2220 11        2186        LDX     X,T$TEM12,T       GET STARTING ADDRESS
001064  000002 6230 12        2187        EAX     Y,2,X             SET SET INDEX REGISTER
001065  000000 5602 04        2188 GETC8  RPDX    ,4                ZERO MEMORY LIKE MAD
001066  000000 7570 12        2189        STAQ    0,X               CHONK
001067  000000 7570 13        2190        STAQ    0,Y               CHONK
001070  002000 1240 03        2191        SBLX    Z,$1K,DU          REMOVE A K AS DONE
001071  001065 6030 00        2192        TRC     GETC8             TEST FOR MORE
                              2193 *
```

```
                                2194 *       FINISH UP
                                2195 *
                      001072    2196 GETC5   BSS      0
001072  000016 3220 11          2197         LCX      X,T$TEM10,T      GET COMPLEMENT OF LENGTH TO RETURN
001073  003320 0420 00          2198         ASX      X,$AVAIL         REDUCE AVAIL BY THAT AMOUNT
001074  777777 6040 00          2199         TMI      $ERROR           ***BLEWIT
001075  000010 0730 11          2200         LREG     T$TEM16,T        RESTORE REGISTERS
001076  000000 7100 17          2201         TRA      0,L              RETURN
```

```
                  001077           USE     CODE
                                   HEAD    R
                              *
                              *
                              *                                REQUEST MORE
                              *
                              *    THIS SUBROUTINE DOES A REQUEST FOR MORE MEMORY.  IT ASKS
                              *    FOR ONE CORE UNIT.  UPON THE SUCCESSFUL COMPLETION OF THE
                              *    REQUEST. THE NEW MEMORY IS LINKED ON THE FREE LIST BY
                              *    CALLING RELC.
                              *
                              *    CALL BY
                              *            TRA MORE  (ONLY *GETC* SHOULD CALL THIS ROUTINE)
                              *    CALL WITH
                              *            C(T) = TRAP-BLOCK
                              *    CALLS
                              *            $CHSEG
                              *            R$RELC
                              *    EXIT    TO R$GETC3
                              *    DESTROYS <ALL REGISTERS>
                              *    USES NO LOCAL TEMPORARIES
                              *
                  001077      MORE BSS     0
 001077  003323 2220 00            LDX     X.$MTOP          GET CURRENT TOP OF MEMORY
 001100  001000 0220 03            ADLX    X.$MQUAN.DU      ADD ONE CORE UNIT
 001101  003323 7420 00            STX     X.$MTOP          SAVE NEW EXPECTED TOP
 001102  001301 7070 00            TSX     L.MREQ           REQUEST MEMORY
                  001103      MORE1 BSS    0
END OF BINARY CARD LPCP0023
 001103  003323 2350 00            LDA     $MTOP            GET BACK OLD TOP OF MEMORY
 001104  001000 1350 03            SBLA    $MQUAN.DU        IN AU
 001105  001000 2750 07            ORA     $MQUAN.DL        AND LENGTH IN AL
                  001106           RELC    A                NOW DO A RELEASE TO LINK ON FREE LIST
 001106  001110 7070 00            TSX     L.R$RELC
 001107  001024 7100 00            TRA     GETC3            TRY AGAIN
```

```
                  001110          USE     CODE
                                  HEAD    R
                          *
                          *
                          *                         DE-ALLOCATION
                          *
                          *       THIS ROUTINE LINKS THE 'RELEASED' BLOCK OF MEMORY INTO
                          *       THE FREE MEMORY LIST ACCORDING TO THE BLOCK'S ADDRESS.
                          *       SINCE THE FREE LIST IS ORDERED BY BLOCK ADDRESSES FROM
                          *       LOWEST TO HIGHEST, IN ORDER TO MAKE INSERTIONS AND DE-
                          *       LETIONS EASIEST THERE IS ASSOCIATED WITH EACH BLOCK A
                          *       FORWARD AND BACKWARD POINTER AS WELL AS A COUNT OF THE
                          *       TOTAL NUMBER OF WORDS IN THE BLOCK.
                          *
                          *       CALL BY
                          *               TSX L,R$RELC
                          *       CALL WITH
                          *               C(AU) = BLOCK-ADDRESS
                          *               C(AL) = LENGTH
                          *       CALLS
                          *               R$MEMCK (CONDITIONALLY)
                          *       EXIT TO 0,L
                          *       PRESERVES ALL REGISTERS EXCEPT C(A)
                          *       USES LOCAL TEMPORARIES MREG THRU MREG+7
                          *
                  001110          USE     CODE
                  001110    RELC  BSS     0              RELEASE A BLOCK OF MEMORY
 001110  003400 7530 00           SREG    MREG           SAVE REGISTERS
 001111  777777 3750 03           ANA     -1,DU          ISOLATE RELEASE ADDRESS
 001112  003406 7550 00           STA     TEMP           SAVE FOR TEST
 001113  003322 2350 00           LDA     $MTOPO         TEST TO SEE IF ADDRESS
 001114  003323 2360 00           LDQ     $MTOP          OF BLOCK TO BE RELEASED
 001115  003406 1110 00           CWL     TEMP           IS IN BUFFER AREA
 001116  777777 6010 00           TNZ     $ERROR         BLEWIT
 001117  003406 2220 00           LDX     X,TEMP         GET INSERT ADDRESS
 001120  003404 7200 00           LXL     0,MREG+4       GET LENGTH
 001121  777777 6000 00           TZE     $ERROR         ***BLEWIT
 001122  003406 0400 00           ASX     0,TEMP         ADD TO STARTING ADDRESS
 001123  000000 4400 12           SXL     0,LEN,X        SAVE IN BLOCK
 001124  003406 1110 00           CWL     TEMP           TEST IF ALL IS IN RANGE
 001125  777777 6010 00           TNZ     $ERROR         ***BLEWIT
 001126  003353 2220 00           LDX     X,FIRST        GET POINTER TO FIRST FREE BLOCK OF FREE LIST
 001127  001131 7100 00           TRA     *+2            ENTER SEARCH LOOP
                          *
                          *       LOCATE WHERE TO INSERT BLOCK IN FREE LIST ACCORDING TO ADDRESS
                          *
 END OF BINARY CARD LPCP0024
 001130  000000 2220 12           LDX     X,LINKF,X      GET PTR TO NEXT BLOCK ON FREE LIST
 001131  001144 6000 00           TZE     RELC2          DID WE JUST FALL OFF THE END OF THE LIST?
 001132  003404 1020 00           CMPX    X,INSRT        NO,ARE WE POINTING PAST THE HOLE?
```

```
001133  001130 6020 00          TNC     *-3             NO. LOOK AGAIN
                          *
                          *
                          *
                          *      LOCATED POSITION OF BLOCK WITH XR-X POINTING TO SUCCESSOR
                          *
               001134       RELC1   BSS     0
001134  003404 2240 00              LDX     Z,INSRT         NOW Z POINTS TO INSERT
001135  000001 2230 12              LDX     Y,LINKB,X       *** Y POINTS TO PREDECESSOR
                                                            *** X POINTS TO SUCCESSOR
001136  000001 4500 14              STZ     LINKB,Z         ***DBG
001137  000001 7430 14              STX     Y,LINKB,Z       SET BACKWARD LINK OF INSERT
001140  000000 7420 14              STX     X,LINKF,Z        "  FORWARD  "    "    "
001141  000001 7440 12              STX     Z,LINKB,X       RESET SUCCESSOR'S BACKWARD LINK
001142  000000 7440 13              STX     Z,LINKF,Y       AND FINISH LINKING BY RESETTING PREDECESSOR'
                                                            ***FORWARD LINK.
001143  001146 7100 00              TRA     RELC3           NOW DO SOME RE-COMBINING
                          *
                          *      RELEASED BLOCK FITS BETWEEN THE LAST FREE BLOCK AND 'LAST'
                          *
               001144       RELC2   BSS     0
001144  003355 6220 00              EAX     X,LAST          SO MAKE X POINT TO SUCCESSOR
001145  001134 7100 00              TRA     RELC1           AND TREAT NORMALLY
                          *
                          *
                          *      NOW THAT THE BLOCK HAS BEEN CORRECTLY LINKED ON THE FREE
                          *      LIST TRY TO RECOMBINE WITH ITS BUDDIES.
                          *      THERE ARE FOUR (4) CASES TO CONSIDER:
                          *
                          *              CASE I: THERE EXISTS A BUDDY ABOVE IN MEMORY
                          *              CASE II:THERE EXISTS A BUDDY BELOW IN MEMORY
                          *              CASE III:ALL OF THE ABOVE
                          *              CASE IV: NONE OF THE ABOVE
                          *
               001146       RELC3   BSS     0
001146  001206 7070 00              TSX     L,RELC4         CASE I: TRY REJOINING WITH BUDDY ABOVE
001147  000000 6220 14              EAX     X,0,Z           LET X POINT TO INSERT
001150  000000 6240 13              EAX     Z,0,Y           LET Z POINT TO ITS PREDECESSOR
001151  003404 7440 00              STX     Z,INSRT         NOW CALL THE PREDECESSOR INSERT
001152  001206 7070 00              TSX     L,RELC4         FUDGE CASE II TO CASE I
```

```
                              2327 *
                              2328 *        WE'RE DONE, BLOCK IS CORRECTLY LINKED IN LIST.
                              2329 *        GRAB CALLER'S REGISTERS AND RETURN.
                              2330 *
   001153  003404 7220 00     2331          LXL     X,MREG+4          GET BACK AMOUNT RELEASED
   001154  003320 0420 00     2332          ASX     X,$AVAIL          STATISTIC: FREE CORE
   001155  003323 2220 00     2333          LDX     X,$MTOP           GET TOP OF MEMORY
END OF BINARY CARD LPCP0025
   001156  003322 1220 00     2334          SBLX    X,$MTOP0          COMPUTE BUFFER SIZE
   001157  001000 1220 03     2335          SBLX    X,$MQUAN,DU       MINUS A CORE QUANTUM
   001160  001202 6020 00     2336          TNC     RELCX             OK, FORGET ABOUT IT
   001161  003321 1020 00     2337          CMPX    X,$MEMRQ          COMPARE IT TO MEMORY REQUIREMENTS
   001162  001202 6020 00     2338          TNC     RELCX             CAN WE AFFORD TO RELEASE SOME MEMORY?
   001163  003306 2340 00     2339          SZN     Q$BUSY+Q$CORE     BUT FIRST IS MEMORY BUSY?
   001164  001202 6010 00     2340          TNZ     RELCX             DON'T UNDER ANY CIRCUMSTANCES TRY TO CALL MEMCK.
   001165  003360 2210 03     2341          LDX     T,TRAP,DU         GET DUMMY TCB
   001166  003360 7260 07     2342          LXL     J,TRAP,DL         AND DUMMY JCB
                   001167     2343          BRANCH  NOPASS,MEMCK      OK TO SET UP TASK TO RELEASE MEMORY
   001167  001325 7000 00                   TSX     0,T$GETT
   001170  000000 6220 11                   EAX     X,0,T
   001171  000005 2210 12                   LDX     T,T$LINK,X
   001172  000000 6210 12                   EAX     T,0,X
   001173  001223 6200 00                   EAX     0,MEMCK
   001174  000004 7400 11                   STX     0,T$TRA,T
   001175  000004 6200 11                   EAX     0,Q$OFFST,T
   001176  003262 7170 00                   XED     Q$XADD+Q$TASK
   001177  000005 2210 12                   LDX     T,T$LINK,X
                   001200                   BUGXR   (0,X)
                   525211          BUGBUG   SET     BUGBUG+1
   001200  525211 2200 03                   LDX     0,BUGBUG,DU
   001201  525211 2220 03                   LDX     X,BUGBUG,DU
                   001202     2344 RELCX    BSS     0                 EXIT
   001202  003400 0730 00     2345          LREG    MREG              RESTORE REGISTERS
                   001203     2346          BUGA                      REMIND HIM THAT A IS INVALID
                   525212          BUGBUG   SET     BUGBUG+1
   001203  525212 2350 03                   LDA     BUGBUG,DU
END OF BINARY CARD LPCP0026
   001204  525212 2750 07                   ORA     BUGBUG,DL
   001205  000000 7100 17     2347          TRA     0,L               RETURN TO CALLER
```

```
                              2349 *
                              2350 *        THIS SUBROUTINE TRIES TO RECOMBINE A BLOCK OF MEMORY
                              2351 *        AND ITS BUDDY ABOVE (IF IT IS ALSO IN THE FREE LIST)
                              2352 *        INTO A SINGLE BLOCK.
                              2353 *
                              2354 *        CALL BY
                              2355 *                TSX L,RELC4
                              2356 *        CALL WITH
                              2357 *                C(Z) = INSERT-BLOCK-ADDRESS
                              2358 *                C(X) = INSERT'S SUCCESSOR
                              2359 *        EXIT TO 0,L
                              2360 *        DESTROYS 0,X,Z
                              2361 *
                    001206    2362 RELC4  BSS     0
001206  000000 7200 14        2363        LXL     0,LEN,Z        COMPUTE ADDRESS OF BUDDY IN
001207  003404 0200 00        2364        ADLX    0,INSRT        ...UPPER MEMORY
001210  000000 1000 14        2365        CMPX    0,LINKF,Z      IS IT PART OF THE FREE LIST?
001211  000000 6010 17        2366        TNZ     0,L            NO, SO RETURN TO CALLER
                              2367 *
                              2368 *        FOUND BUDDY
                              2369 *
001212  000000 2200 12        2370        LDX     0,LINKF,X      RESET INSERT'S FORWARD LINK TO
001213  000000 7400 14        2371        STX     0,LINKF,Z        THAT OF BUDDY'S.
001214  000001 7440 10        2372        STX     Z,LINKB,0      RESET PREDECESSOR'S BACKWARD POINTER
001215  000000 7200 12        2373        LXL     0,LEN,X        GET THE LENGTH OF BUDDY
001216  003406 7400 00        2374        STX     0,TEMP           SAVE IT
001217  000000 7200 14        2375        LXL     0,LEN,Z        GET LENGTH OF INSERT
001220  003406 0200 00        2376        ADLX    0,TEMP         TO GET TOTAL LENGTH
001221  000000 4400 14        2377        SXL     0,LEN,Z        AND RESET LENGTH OF BLOCK
001222  000000 7100 17        2378        TRA     0,L            RETURN TO CALLER
                              2379 *
                              2380 *
                    003357    2381        USE     STORE
                    003360    2382        EIGHT
                    003360    2383 TRAP   BSS     T$LEN          DUMMY TCB WHEN CREATING MEMCK
                    003400    2384 MREG   EQU     TRAP+T$LEN-8   TEMP REGISTER STORAGE
                    003404    2385 INSRT  EQU     MREG+4         POINTER TO INSERT BLOCK
                    003406    2386 TEMP   EQU     MREG+6         FOR SCRATCH WORK
                    001223    2387        USE     PREVIOUS
```

```
                      001223     2389          USE     CODE
                                 2390          HEAD    R
                                 2391 *
                                 2392 *
                                 2393 *                              MEMORY RELEASE
                                 2394 *
                                 2395 *        THIS TASK CHECKS TO SEE IF IT CAN GIVE BACK MEMORY TO THE
                                 2396 *        SYSTEM.  IF SO,  IT RETURNS BLOCKS IN MULTIPLES OF $MQUAN.
                                 2397 *        THE ALGORITHM IS AS FOLLOWS:
                                 2398 *        GIVEN:
                                 2399 *                MEMRQ -- AMOUNT OF MEMORY NEEDED THOUGH NOT NECESSARIYLY IN USE
                                 2400 *                TOTAL -- TOTAL BUFFER AREA SIZE
                                 2401 *                AVAIL -- AMOUNT OF FREE CORE THOUGH NOT NECESSARILY CONTIGOUS
                                 2402 *                USED  -- AMOUNT OF CORE BUSY (TOTAL-AVAIL)
                                 2403 *        REQUIREMENT:
                                 2404 *                USED <= MEMRQ
                                 2405 *        ENTERED WHEN:
                                 2406 *                TOTAL-MEMRQ => MQUAN
                                 2407 *        RESTRICTIONS:
                                 2408 *                (1)  CAN RELEASE ONLY THE LAST PHYSICAL BLOCK
                                 2409 *                (2)  A RELEASE CAN BE EFFECTED ONLY IF AFTER THE RELEASE
                                 2410 *                     THERE IS AT LEAST ONE BLOCK OF MEMORY THAT IS OF
                                 2411 *                     MEMRQ-USED SIZE.
                                 2412 *        MEMCK IS AN ASYNCHRONOUS TASK, HENCE IT MAY ALL TEMP'S
                                 2413 *        USES NO LOCAL TEMPORARIES
                                 2414 *
                      001223     2415 MEMCK    BSS     0
                      001223     2416          ENQ     CORE               GET ON CORE QUEUE
   001223  003302 7170 00                      XED     Q$CORE+Q$XENQ
   001224  003323 2220 00        2417          LDX     X,$MTOP            GET TOP OF BUFFER AREA
   001225  003322 1220 00        2418          SBLX    X,$MTOPO           MINUS BOTTOM = TOTAL
   001226  000027 7420 11        2419          STX     X,T$TEMP1,T        C(T$TEMP1,T) = TOTAL
 END OF BINARY CARD LPCP0027
   001227  003320 1220 00        2420          SBLX    X,$AVAIL           MINUS AVAIL = USED
   001230  777777 6040 00        2421          TMI     $ERROR             ***BLEWIT
   001231  000026 7420 11        2422          STX     X,T$TEMP2,T        C(T$TEMP2,T) = USED
   001232  003321 2350 00        2423          LDA     $MEMRQ             GET AMOUNT OF MEMORY REQUIRED
   001233  000026 1350 11        2424          SBLA    T$TEMP2,T          MINUS USED = AMOUNT NEEDED STILL
   001234  000025 7550 11        2425          STA     T$TEMP3,T          SAVE IT
   001235  777777 6040 00        2426          TMI     $ERROR             ***BLWEIT
   001236  001276 6000 00        2427          TZE     MEMX               EXIT IF CONDITIONS CHANGED ON US
   001237  000022 7710 00        2428          ARL     36-18              MOVE TO AL FOR RPL
                                 2429 *
                                 2430 *        SEARCH FOR BLOCK OF 'NEEDED' SIZE
                                 2431 *
   001240  003353 2240 00        2432          LDX     Z,FIRST            GET POINTER TO FIRST FREE BLOCK
   001241  000302 5002 00        2433 MEM1     RPL     ,TNC,TZE           RUN DOWN LINKED LIST
   001242  000000 1150 14        2434          CMPA    0,Z                FOR A BLOCK BIG ENOUGH
   001243  001250 6000 00        2435          TZE     MEM2               FOUND A BLOCK THAT IS 'JUST RIGHT'
   001244  001250 6020 00        2436          TNC     MEM2               MORE THAN BIG ENOUGH
```

R CORE MANAGEMENT -- MEMORY RELEASE

```
001245  000000 2240 14     2437          LDX    Z,0,Z            GET BACK PTR OF WHERE WE LEFT OFF
001246  001241 6010 00     2438          TNZ    MEM1             IF NOT AT END OF LIST, KEEP SEARCHING
001247  001276 7100 00     2439          TRA    MEMX             OTHERWISE EXIT
                           2440 *
                           2441 *        Z POINTS TO BLOCK NEEDED EVENTUALLY
                           2442 *
001250  000000 7230 14     2443 MEM2     LXL    Y,LEN,Z          GET THE LENGTH OF THIS BLOCK
001251  003355 6220 00     2444          EAX    X,LAST           GET PTR TO LAST BLOCK
001252  000001 1040 12     2445          CMPX   Z,LINKB,X        DOES Z POINT TO LAST BLOCK?
001253  001256 6010 00     2446          TNZ    *+3              NO, NOT LAST
001254  000025 1230 11     2447          SBLX   Y,T$TEMP3,T      YES, SO SUBTRACT OFF NEEDED
END OF BINARY CARD LPCP0028
001255  001260 7100 00     2448          TRA    MEM3             CONTINUE
001256  000001 2240 12     2449          LDX    Z,LINKB,X        GET PTR TO LAST FREE BLOCK
001257  000000 7230 14     2450          LXL    Y,LEN,Z          GET ITS LENGTH
001260  001000 1030 03     2451 MEM3     CMPX   Y,$MQUAN,DU      COMPARE TO CORE QUANTUM
001261  001276 6020 00     2452          TNC    MEMX             IF SMALLER, EXIT
001262  777000 3630 03     2453          ANX    Y,-$MQUAN,DU     OTHERWISE ROUND DOWN TO MULTIPLE OF MQUAN
001263  000024 7430 11     2454          STX    Y,T$TEMP4,T      SAVE AMOUNT FOR RELEASE
001264  000000 7200 14     2455          LXL    0,LEN,Z          GET BACK ITS LENGTH
001265  000024 1200 11     2456          SBLX   0,T$TEMP4,T      MINUS AMOUNT TO RELEASE
001266  000000 4400 14     2457          SXL    0,LEN,Z          RESTORE NEW LENGTH
001267  001272 6010 00     2458          TNZ    *+3              IS THE BLOCK NULL
001270  000001 2200 14     2459          LDX    0,LINKB,Z        YES, RESET LAST
001271  000001 7400 12     2460          STX    0,LINKB,X        TO POINT TO NEW LAST BLOCK
001272  000024 3220 11     2461          LCX    X,T$TEMP4,T      GET BACK AMOUNT TO BE RELEASED
001273  003323 0420 00     2462          ASX    X,$MTOP          SUBTRACT FROM TOP OF MEMORY PTR
001274  003320 0420 00     2463          ASX    X,$AVAIL         SUBTRACT FROM AVAILABLE
001275  001301 7070 00     2464          TSX    L,MREQ           RELEASE MEMORY TO SYSYTEM
                  001276   2465 MEMX     BSS    0                DONE
                  001276   2466          DEQ    CORE             RELEASE CORE QUEUE, AWAKEN NEXT TASK
001276  003304 7170 00                   XED    Q$CORE+Q$XDEQ
001277  001335 7000 00     2467          TSX    0,T$RELT         AND RELEASE TCB
                  001300   2468          EXIT                    EVAPORATE
001300  001547 7100 00                   TRA    $EXIT
```

```
                        001301   2470        USE      CODE
                                 2471        HEAD     R
                                 2472 *
                                 2473 *                          MEMOREY REQUESTS
                                 2474 *
                                 2475 *     THIS SUBROUTINE RESETS THE TOP OF MEMORY TO THE ADDRESS
                                 2476 *     SPECIFIED BY C(MTOP).
                                 2477 *
                                 2478 *     CALL BY
                                 2479 *              TSX L.MREQ
                                 2480 *     CALL WITH
                                 2481 *              C(XT) = TBLOCK-ADDRESS
                                 2482 *              C(XJ) = JBLOCK-ADDRESS
                                 2483 *              C(MTOP) = NEW SETTING
                                 2484 *     CALLS
                                 2485 *              $CHSEG
                                 2486 *     EXITS TO 0.L
                                 2487 *     RETURNS WITH
                                 2488 *              C(XT) = TBLOCK-ADDRESS
                                 2489 *              C(XJ) = JBLOCK-ADDRESS
                                 2490 *              C(XL) = RESTART ADDRESS
                                 2491 *              C(MTOP) = NEW SETTING
                                 2492 *      USES NO LOCAL TEMPORARIES, ONLY T$TEM9
                                 2493 *
  001301  000017 4470 11         2494 MREQ   SXL      L.T$TEM9.T       SAVE RETURN ADDRESS
END OF BINARY CARD LPCP0029
  001302  003323 2200 00         2495        LDX      0.$MTOP          GET PASSED SETTING
  001303  000777 0200 03         2496        ADLX     0.$MQUAN-1.DU    ROUND UP TO
  001304  777000 3600 03         2497        ANX      0.-$MQUAN.DU     NEXT CORE MULTIPLE
  001305  003323 7400 00         2498        STX      0.$MTOP          AND SAVE IT
                        001306   2499 MREQ1  CHSEG    $BUFSEG.$MTOP    MEMORY REQUEST
  001306  000605 7000 00                     TSX      0.$CHSEG
  001307  000000 0000 00                     ARG      $BUFSEG
  001310  003323 0000 00                     ARG      $MTOP
                        001311   2500        CHECK    MREQ2.B$BZ.MREQ1.B$IO.MREQ1
  001311  000000 7200 11                     LXL      0.T$SRW1.T
  001312  000077 3600 03                     ANX      0.B$STMK.DU
  001313  001321 6000 00                     TZE      MREQ2
  001314  000003 1000 03                     CMPX     0.B$BZ.DU
  001315  001306 6000 00                     TZE      MREQ1
  001316  000004 1000 03                     CMPX     0.B$IO.DU
  001317  001306 6000 00                     TZE      MREQ1
  001320  777777 7100 00                     TRA      $ERROR
                        001321   2501 MREQ2  BSS      0                SUCCESSFUL REQUEST
  001321  000017 7270 11         2502        LXL      L.T$TEM9.T       RETRIEVE RETURN ADDRESS
                        001322   2503        BUGL     (T$TEM9.T)       BUG IT
                        525213        BUGBUG SET      BUGBUG+1
  001322  525213 2200 03                     LDX      0.BUGBUG.DU
  001323  000017 4400 11                     SXL      0.T$TEM9.T
  001324  000000 7100 17         2504        TRA      0.L              RETURN TO CALLER
```

```
001325          USE     CODE
                HEAD    T
      *
      *
      *         THESE MACROS GET AND RELEASE TRAP BLOCKS.
      *         CLOBBERS C(0), C(X), C(T).
      *
      *
      *                                 GETT
      *
      GETT      MACRO   <NO-ARGUMENTS>
                TSX     0,T$GETT        CALL SUBROUTINE
                ENDM    GETT
      *
      *
      *                                 RELT MACRO
      *
      RELT      MACRO   <NO-ARGUMENTS>
                TSX     0,T$RELT        CALL SUBROUTINE
                ENDM    RELT
```

```
                    001325   2527          USE      CODE
                             2528          HEAD     T
                             2529 *
                             2530 *
                             2531 *                                   GETT
                             2532 *
                             2533 *        THIS SUBROUTINE GETS A TRAP BLOCK. C(XT) POINTS TO THE
                             2534 *        CURRENT TBLOCK WHILE C(T$LINK,T) POINTS TO THE OLD ONE
                             2535 *        ENTERED WITH
                             2536 *                 C(XT) = TCB
                             2537 *                 C(XJ) = JCB
                             2538 *        ENTERED BY
                             2539 *                 TSX 0,R$GETT
                             2540 *        CALLS
                             2541 *                 R$GETC
                             2542 *        RETURNS 0,0
                             2543 *        RETURNS WITH
                             2544 *                 C(XT) = NEW TCB
                             2545 *                 C(XJ) = JCB
                             2546 *
                             2547 *
                    001325   2548 GETT     BSS      0
                    001325   2549          GETC     (LEN,DL)          GET A BLOCK ABOUT THE SIZE OF A TBLOCK
   001325  000030 2350 07                   LDA      LEN,DL
   001326  001011 7070 00                   TSX      L,R$GETC
 END OF BINARY CARD LPCP0030
   001327  000005 7410 01    2550          STX      T,LINK,AU         POINT TO PREVIOUS TBLOCK
   001330  000000 6210 01    2551          EAX      T,0,AU            MAKE C(XT) POINT TO NEW BLOCK
   001331  777777 6000 00    2552          TZE      $ERROR            ***DBG
   001332  777777 6040 00    2553          TMI      $ERROR            ***DBG
   001333  000006 4460 11    2554          SXL      J,T$JCB,T         SET JCB POINTER
   001334  000000 7100 10    2555          TRA      0,0               RETURN TO CALLER
```

```
                  001335           USE     CODE
                                   HEAD    T
                          *
                          *
                          *                                RELT
                          *
                          *        THIS SUBROUTINE RELEASES THE CURRENT TRAP BLOCK.
                          *
                          *        ENTERED WITH
                          *                C(XT) = TCB
                          *                C(XJ) = JCB
                          *        ENTERED BY
                          *                TSX O,R$RELT
                          *        CALLS
                          *                R$RELC
                          *        RETURNS TO 0,0
                          *        RETURNS WITH
                          *                C(XJ) = JCB
                          *
                          *
                  001335    RELT   BSS     0
001335  000000 6210 11             EAX     T,0,T            CHECK FOR LEGAL RELEASE
001336  777777 6000 00             TZE     $ERROR           NG
001337  000000 6350 11             EAA     0,T              TRAP ADDRESS TO AU
001340  000030 2750 07             ORA     LEN,DL           TRAP LEN TO AL
                  001341           RELC    A                RELEASE IT
001341  001110 7070 00             TSX     L,R$RELC
                  001342           BUGXR   T                SPPML
                  525214    BUGBUG SET     BUGBUG+1
001342  525214 2210 03             LDX     T,BUGBUG,DU
001343  000000 7100 10             TRA     0,0              RETURN TO CALLER
```

```
001344          USE     CODE
                HEAD    J
     *
     *
     *          THESE MACROS GET AND RELEASE JCB'S (JOB CONTROL BLOCKS)
     *          CLOBBERS C(A).  RETURNS IMMEDIATELY TO CALLER.
     *
     *
     *                                  GETJ MACRO
     *
     GETJ       MACRO   <NO-ARGUMENT>
                TSX     L.J$GETJ        CALL SUBROUTINE
                ENDM    GETJ            RETURN C(XJ) = JCB-ADDRESS
     *
     *
     *                                  RELJ MACRO
     *
     RELJ       MACRO   C(XJ) = JCB TO BE RELEASED
                TSX     L.J$RELJ        CALL SUBROUTINE
                ENDM    RELJ            CLOBBERS C(XJ)
```

```
                              001344                USE     CODE
                                                    HEAD    J
                                          *
                                          *
                                          *                              GETJ
                                          *
                                          *         THIS SUBROUTINE GETS THE FIRST AVAILABLE JCB.
                                          *         IT RETURNS A POINTER TO IT IN C(XJ).
                                          *
                                          *         ENTERED WITH
                                          *                 C(XT) = TCB
                                          *         ENTERED BY
                                          *                 TSX L.J$GETJ
                                          *         CALLS
                                          *                 NONE
                                          *         RETURNS 0.L
                                          *         RETURNS WITH
                                          *                 C(XT) = TCB
                                          *                 C(XJ) = NEW JCB
                                          *
                                          *
                              001344        GETJ    BSS     0               ENTRY POINT
   001344  003644 6260 00                           EAX     J.JCB0          POINT TO FIRST JCB
   001345  000000 2350 07                           LDA     0.DL            SET A FOR MATCH
   001346  006300 5202 24                           RPT     JCBN.LEN.TZE    SEARCH
   001347  000000 1150 16                           CMPA    0.J             TEST
   001350  777777 6010 00                           TNZ     $ERROR          OOPS
   001351  777755 6260 16                           EAX     J.-LEN+1.J      POINT TO FREE ONE
   001352  777777 7410 16                           STX     T.ALLC.J        MARK IT BUSY
 END OF BINARY CARD LPCP0031
   001353  046300 5202 01                           RPT     LEN-1.1.TZE     CLEAN OUT BLOCK
   001354  000000 4500 16                           STZ     0.J             CLEAR IT
   001355  777755 6260 16                           EAX     J.-LEN+1.J      RESET J
   001356  000000 7100 17                           TRA     0.L             RETURN TO CALLER
```

```
                    001357             USE     CODE
                                       HEAD    J
                                *
                                *
                                *                          RELJ
                                *
                                *      THIS SUBROUTINE RELEASES THE JCB POINTED TO BY C(XJ).
                                *
                                *      ENTERED WITH
                                *              C(XT) = TCB
                                *              C(XJ) = JCB
                                *      ENTERED BY
                                *              TSX L.J$RELJ
                                *      CALLS
                                *              NONE
                                *      RETURNS 0.L
                                *      RETURNS WITH
                                *              C(XT) = TCB
                                *
                                *
                    001357        RELJ BSS     0               ENTRY POINT
001357 777777 2340 16                  SZN     ALLC.J          IS IT BUSY
001360 777777 6000 00                  TZE     $ERROR          IS SHOULD BE!
001361 777777 4500 16                  STZ     ALLC.J          MAKR IT FREE
                    001362             BUGXR   J               SPPML
                    525215      BUGBUG SET     BUGBUG+1
001362 525215 2260 03                  LDX     J.BUGBUG.DU
001363 000000 7100 17                  TRA     0.L             RETURN TO CALLER
```

```
001364          USE     CODE
                HEAD    R
      *
      *
      *                                 DESCRIPTION
      *
      *         THESE MACROS GET A RELEASE A SINGLE PERIPHERAL
      *
      *
      *                                 GETP
      *
      *         GET A PERIPHERAL
      *
      GETP      MACRO   TYPE/ 'A'
                INE     '#1','A'
                LDA     #1              GET TYPE IN AU
                TSX     L.R$GETP        CALL SUBROUTINE
                ENDM    GETP
      *
      *
      *                                 RELP MACRO
      *
      *         RELEASE A PERIPHERAL
      *
      RELP      MACRO   DEVICE NUMBER/ 'A'
                INE     '#1','A'
                LDA     #1              GET DEVICE NUMBER IN AL
                TSX     L.R$RELP        CALL SUBROUTINE
                ENDM    RELP
```

```
                    001364               USE     CODE
                                         HEAD    R
                                   *
                                   *
                                   *                           GETP
                                   *
                                   *     GETP GETS A SINGLE PERIPHERAL UNIT OF A SPECIFIED TYPE.
                                   *     IF THERE ARE MORE UNITS OF THE SAME TYPE STILL AVAILABLE
                                   *     AFTER THE GET, THEN THE NEXT JOB WAITING FOR THE SAME
                                   *     PERIPHERAL TYPE IS AWAKENED.
                                   *
                                   *     CALL BY
                                   *             TSX L,R$GETP
                                   *     CALL WITH
                                   *             C(J) = JOB NUMBER
                                   *             C(T) = TBLOCK ADDRESS
                                   *             C(L) = RETURN ADDRESS
                                   *             C(AU) = PERIPHERAL TYPE
                                   *     CALLS
                                   *             NONE
                                   *     RETURNS WITH
                                   *             C(J) = JOB NUMBER
                                   *             C(T) = TBLOCK ADDRESS
                                   *             C(AU) = PERIPHERAL TYPE
                                   *             C(AL) = DEVICE NUMBER
                                   *             C(QU) = DEVICE FRN
                                   *             C(QL) = DEVICE NUMBER
                                   *     EXIT TO 0,L
                                   *     USES
                                   *             NO LOCAL TEMPORARIES
                                   *             T$TEMP1,T$TEMP2
                                   *
                    001364     GETP      BSS     0                 ENTRY POINT
 001364  000004 4470 11                  SXL     L,T$TRA,T         SAVE RETURN ADDRESS
 001365  000027 7550 11                  STA     T$TEMP1,T         SAVE TYPE
 001366  000027 2240 11                  LDX     Z,T$TEMP1,T       GET PERIPHERAL TYPE
 001367  000003 3640 03                  ANX     Z,3,DU            MASK TO TYPE ONLY
 001370  003074 2240 14                  LDX     Z,TABLE,Z         POINT TO PERIPHERAL TYPE TABLE
 001371  777777 6000 00                  TZE     $ERROR            NO SUCH PERIPHERAL
 001372  000026 7440 11                  STX     Z,T$TEMP2,T       SAVE TYPE POINTER
                                   *
                                   *     LOOP TO LOOK FOR FREE DEVICE
                                   *
 001373  000001 3360 14                  LCQ     MAX,Z             GET NUMBER TO CHECK COMPLEMENTED
 001374  777777 6000 00                  TZE     $ERROR            ***BLEWIT
 001375  000000 2200 14                  LDX     0,PTR,Z           GET POINTER TO UNITS
 001376  700000 2350 03                  LDA     BUSY+CLOSE+RSVE,DU 'GET BITS TO CHECK
END OF BINARY CARD LPCP0032
 001377  000002 3150 10        GETP1     CANA    FLAG,0            IS THIS UNIT FREE?
 001400  001405 6000 00                  TZE     GETP2             YES, TAKE IT
```

```
001401  000003 0200 03        2747         ADLX   0,RSDEVLN,DU    STEP TO NEXT DEVICE
001402  000001 0760 07        2748         ADQ    1,DL            TEST FOR DONE
001403  001377 6040 00        2749         TMI    GETP1           NO, SO CONTINUE SEARCH
001404  777777 7100 00        2750         TRA    $ERROR          ***SOMEBODY FORGOT TO RESERVE 'EM!
                              2751 *
                              2752 *
                              2753 *       FOUND THE REQUESTED UNIT
                              2754 *
               001405         2755 GETP2   BSS    0
001405  000027 4400 11        2756         SXL    0,T$TEMP1,T     SAVE POINTER TO DEVICE
001406  000002 4460 10        2757         SXL    J,ALLC,0        MARK IT ALLOCATED TO US
001407  400000 2220 03        2758         LDX    X,BUSY,DU       SET THE BUSY BIT ON
001410  000002 2420 10        2759         ORSX   X,FLAG,0
001411  000027 2350 11        2760         LDA    T$TEMP1,T       GET RETURN WORDS FOR USER
001412  000001 2360 10        2761         LDQ    FRN,0
               001413         2762         BUGXR  (0,X,Y,Z,Q)
               525216              BUGBUG SET    BUGBUG+1
001413  525216 2200 03                     LDX    0,BUGBUG,DU
001414  525216 2220 03                     LDX    X,BUGBUG,DU
001415  525216 2230 03                     LDX    Y,BUGBUG,DU
001416  525216 2240 03                     LDX    Z,BUGBUG,DU
001417  525216 2250 03                     LDX    Q,BUGBUG,DU
001420  000004 7270 11        2763         LXL    L,T$TRA,T       RETRIEVE RETURN
               001421         2764         BUGL   (T$TRA,T)       BUG IT
               525217              BUGBUG SET    BUGBUG+1
001421  525217 2200 03                     LDX    0,BUGBUG,DU
001422  000004 4400 11                     SXL    0,T$TRA,T
001423  000027 2350 11        2765         LDA    T$TEMP1,T       GET RETURN WORD FOR CALLER
               001424         2766         BUG    (T$TEMP1,T)     BUG IT
               525220              BUGBUG SET    BUGBUG+1
001424  525220 2200 03                     LDX    0,BUGBUG,DU
END OF BINARY CARD LPCP0033
001425  000027 7400 11                     STX    0,T$TEMP1,T
001426  000027 4400 11                     SXL    0,T$TEMP1,T
               001427         2767         BUG    (T$TEMP2,T)
               525221              BUGBUG SET    BUGBUG+1
001427  525221 2200 03                     LDX    0,BUGBUG,DU
001430  000026 7400 11                     STX    0,T$TEMP2,T
001431  000026 4400 11                     SXL    0,T$TEMP2,T
001432  000000 7100 17        2768         TRA    0,L             RETURN TO CALLER
```

```
                  001433                  USE     CODE
                                          HEAD    R
                                *
                                *
                                *                           RELP
                                *
                                *         RELP RELEASES A PERIPHERAL UNIT OF A SPECIFIED TYPE.
                                *
                                *         CALL WITH
                                *                 TSX L,R$RELP
                                *         CALL WITH
                                *                 C(J) = JOB NUMBER
                                *                 C(T) = TBLOCK ADDRESS
                                *                 C(L) = RETURN ADDRESS
                                *                 C(A) = PERIPHERAL TYPE/ DEVICE ADDRESS
                                *          CALLS
                                *                 C$MESSX  (CONDITIONALLY)
                                *         RETURNS WITH
                                *                 C(J) = JOB NUMBER
                                *                 C(T) = TBLOCK ADDRESS
                                *         EXITS TO 0,L
                                *         USES
                                *                 NO LOCALS
                                *                 T$TEMP1, T$TEMP2, T$TEMP3
                                *
                                *         EXITS WITH PERIPHERAL DEALLOCATED.
                                *
                  001433          RELP    BSS     0                 ENTRY POINT
001433  000004 4470 11                    SXL     L,T$TRA,T         SAVE RETURN ADDRESS
                                *
                                *         PERFORM CONSISTANCY CHECKS
                                *
001434  000027 7550 11                    STA     T$TEMP1,T         SAVE INFORMATION
001435  000027 7220 11                    LXL     X,T$TEMP1,T       GET DEVICE ADDRESS
001436  000002 2340 12                    SZN     FLAG,X            SHOULD BE BUSY
001437  777777 6000 00                    TZE     $ERROR            IT ISN'T!
001440  777777 2360 03                    LDQ     -1,DU             SET FOR LOWER HALF COMPARE
001441  000002 2350 12                    LDA     ALLC,X            GET ALLOCATED JOB NUMBER
001442  000006 2110 11                    CMK     T$JCB,T           CHECK FOR CORRECT JCB
001443  777777 6010 00                    TNZ     $ERROR            SHOULD BE THE SAME
                  001444                  BUGL    (ALLC,X)          OK, DESTROY IT
                  525222          BUGBUG  SET     BUGBUG+1
001444  525222 2200 03                    LDX     0,BUGBUG,DU
001445  000002 4400 12                    SXL     0,ALLC,X
                                *
                                *         CHECK IF CLOSE REQUESTED
                                *
001446  000002 2350 12                    LDA     FLAG,X            GET PERIPHERAL FLAG
001447  100000 3150 03                    CANA    RSVE,DU           IS IT RESERVED?
001450  001517 6000 00                    TZE     RELP5             NO, SO CONTINUE
```

```
                                 2817 *
                                 2818 *        CLOSE REQUESTED -- CLOSE AND LOG
                                 2819 *
   001451  300000 6750 03        2820          ERA     CLOSE+RSVE,DU    MARK IT CLOSED AND NOT RESERVED
 END OF BINARY CARD LPCP0034
   001452  000002 7550 12        2821          STA     FLAG,X           RESTORE FLAG WORD
   001453  000001 2230 12        2822          LDX     Y,FRN,X          GET FRN OF DEVICE
   001454  000025 7430 11        2823          STX     Y,T$TEMP3,T      SAVE FOR CLOSE
                  001455         2824          BUGU    (FRN,X)          BUG FRN
                  525223             BUGBUG SET     BUGBUG+1
   001455  525223 2200 03                      LDX     0,BUGBUG,DU
   001456  000001 7400 12                      STX     0,FRN,X
                                 2825 *
                                 2826 *        CLOSE PERIPHERAL
                                 2827 *
                  001457         2828 RELP1    CLOSE   (T$TEMP3,T)      CLOSE THE DEVICE
   001457  000616 7000 00                      TSX     0,$CLOSE
   001460  000025 0000 11                      ARG     T$TEMP3,T
                  001461         2829          CHECK   RELP2,B$BZ,RELP1
   001461  000000 7200 11                      LXL     0,T$SRW1,T
   001462  000077 3600 03                      ANX     0,B$STMK,DU
   001463  001467 6000 00                      TZE     RELP2
   001464  000003 1000 03                      CMPX    0,B$BZ,DU
   001465  001457 6000 00                      TZE     RELP1
   001466  777777 7100 00                      TRA     $ERROR
                  001467         2830 RELP2    BSS     0
                                 2831 *
                                 2832 *        INFORM MINITOR
                                 2833 *
   001467  000006 2240 11        2834          LDX     Z,T$NCB,T        GET POINTER TO NCB
   001470  000031 2350 14        2835          LDA     C$STATE,Z        GET STATE
   001471  000026 7550 11        2836          STA     T$TEMP2,T        SAVE AS STATE FOR CAUSE
   001472  000027 7220 11        2837          LXL     X,T$TEMP1,T      GET BACK POINTER TO DEVICE
   001473  000000 2350 12        2838          LDA     NAME,X           GET NAME
   001474  000007 3750 07        2839          ANA     7,DL             ISOLATE UNIT NUMBER
   001475  000022 7350 00        2840          ALS     18               MOVE TO MESSAGE FIELD
   001476  006000 2750 03        2841          ORA     B$REL,DU         MARK AS RELEASED
   001477  000025 7550 11        2842          STA     T$TEMP3,T        SAVE AS MESSAGE
                  001500         2843          BRANCH  NOPASS,C$MESSX,(T$TEMP2,T),(T$TEMP3,T)
 END OF BINARY CARD LPCP0035
   001500  001325 7000 00                      TSX     0,T$GETT
   001501  000000 6220 11                      EAX     X,0,T
   001502  000005 2210 12                      LDX     T,T$LINK,X
   001503  000026 2360 11                      LDQ     T$TEMP2,T
   001504  000027 7560 12                      STQ     T$TEMP1,X
   001505  000025 2360 11                      LDQ     T$TEMP3,T
   001506  000026 7560 12                      STQ     T$TEMP2,X
   001507  000000 6210 12                      EAX     T,0,X
   001510  001613 6200 00                      EAX     0,C$MESSX
   001511  000004 7400 11                      STX     0,T$TRA,T
```

```
001512  000004 6200 11                    EAX     0,Q$OFFST,T
001513  003262 7170 00                    XED     Q$XADD+Q$TASK
001514  000005 2210 12                    LDX     T,T$LINK,X
               001515                     BUGXR   (0,X)
               525224           BUGBUG    SET     BUGBUG+1
001515  525224 2200 03                    LDX     0,BUGBUG,DU
001516  525224 2220 03                    LDX     X,BUGBUG,DU
                        2844 *
                        2845 *            PERIPHERAL RELEASED
                        2846 *
               001517   2847 RELP5        BSS     0
001517  000027 7220 11  2848              LXL     X,T$TEMP1,T      GET BACK POINTER TO DEVICE
001520  400000 2350 03  2849              LDA     BUSY,DU          GET THE BUSY BIT
001521  000002 6550 12  2850              ERSA    FLAG,X           UNSET IT
               001522   2851 RELP6        BSS     0                BUG WHAT MUST BE BUGGED
               001522   2852              BUG     (T$TEMP1,T)
               525225           BUGBUG    SET     BUGBUG+1
001522  525225 2200 03                    LDX     0,BUGBUG,DU
001523  000027 7400 11                    STX     0,T$TEMP1,T
001524  000027 4400 11                    SXL     0,T$TEMP1,T
               001525   2853              BUG     (T$TEMP2,T)
               525226           BUGBUG    SET     BUGBUG+1
001525  525226 2200 03                    LDX     0,BUGBUG,DU
END OF BINARY CARD LPCP0036
001526  000026 7400 11                    STX     0,T$TEMP2,T
001527  000026 4400 11                    SXL     0,T$TEMP2,T
               001530   2854              BUG     (T$TEMP3,T)
               525227           BUGBUG    SET     BUGBUG+1
001530  525227 2200 03                    LDX     0,BUGBUG,DU
001531  000025 7400 11                    STX     0,T$TEMP3,T
001532  000025 4400 11                    SXL     0,T$TEMP3,T
               001533   2855              BUGXR   (X,Y,Z,Q)
               525230           BUGBUG    SET     BUGBUG+1
001533  525230 2220 03                    LDX     X,BUGBUG,DU
001534  525230 2230 03                    LDX     Y,BUGBUG,DU
001535  525230 2240 03                    LDX     Z,BUGBUG,DU
001536  525230 2250 03                    LDX     Q,BUGBUG,DU
               001537   2856              BUGA
               525231           BUGBUG    SET     BUGBUG+1
001537  525231 2350 03                    LDA     BUGBUG,DU
001540  525231 2750 07                    ORA     BUGBUG,DL
               001541   2857              BUGQ
               525232           BUGBUG    SET     BUGBUG+1
001541  525232 2360 03                    LDQ     BUGBUG,DU
001542  525232 2760 07                    ORQ     BUGBUG,DL
001543  000004 7270 11  2858              LXL     L,T$TRA,T        RETRIEVE RETURN
               001544   2859              BUGL    (T$TRA,T)
               525233           BUGBUG    SET     BUGBUG+1
001544  525233 2200 03                    LDX     0,BUGBUG,DU
001545  000004 4400 11                    SXL     0,T$TRA,T
```

R PERIPHERAL MANAGEMENT -- RELP

```
001546 000000 7100 17        2860        TRA      0,L              RETURN TO CALLER
```

```
                 001547     2862          USE     CODE
                            2863          HEAD    R
                            2864 *
                            2865 *
                            2866 *                                     PERIPHERAL TYPE TABLE
                            2867 *
                            2868 *        THE PERIPHERAL TYPE TABLE HAS AN ENTRY FOR EACH
                            2869 *        TYPE OF RESOURCE.  THUS THE ENTRY POINTER POINTS TO THE
                            2870 *        RESOURCE DEVICE HEADER.  THE HEADER CONTIANS SUCH INFORMATION
                            2871 *        AS THE TOTAL NUMBER OF DEVICES OF THIS TYPE, THE NUMBER CUR-
                            2872 *        RENTLY AVAILABLE, A POINTER TO THE FIRST DEVICE, ETC.  THE
                            2873 *        PERIPHERAL DEVICE ITSELF CONTAINS SUFFICIENT INFORMATION FOR
                            2874 *        ANY OPERATION ON THE PERIPHERAL.
                            2875 *
                            2876 *
                            2877 *        TABLE OF PERIPHERAL TYPES
                            2878 *
                 003074     2879          USE     CONST
                 003074     2880 TABLE    BSS     0
003074  000000 0000 00      2881          ARG     0                0 = INVALID
003075  003410 0000 00      2882 TABCP    ARG     CPTAB            1 = CARD PUNCH
003076  003413 0000 00      2883 TABLP    ARG     LPTAB            2 = LINE PRINTER
003077  000000 0000 00      2884          ARG     0                3 = INVALID
                 001547     2885          USE     PREVIOUS
                            2886 *
                            2887 *
                 000001     2888 TYPCP    EQU     TABCP-TABLE      TYPE: CARD PUNCH
                 000002     2889 TYPLP    EQU     TABLP-TABLE      TYPE: LINE PRINTER
```

```
001547            USE     CODE
                  HEAD    R
             *
             *
             *                                PERIPHERAL HEADER TABLE
             *
             *    A DEVICE HEADER IS THE ITEM THAT AN ENTRY IN THE TABLE OF
             *    PERIPHERAL TYPES POINTS TO.  THE HEADER CONTAINS THE POINTER
             *    TO THE DEVICES OF A CERTAIN TYPE (I.E. LINE PRINTERS).
             *    IT ALSO CONTAINS SUCH INFORMATION AS THE CONFIGURATION OF THE
             *    SYSTEM (MAXIMUM NUMBER OF DEVICES OF A CERTAIN TYPE).
             *    LASTLY. IT CONTAINS A POINTER TO THE CORRESPONDING QUEUE.
             *
             *
             *    FORMAT OF DEVICE HEADER
             *
000000       PTR    EQU     0                 POINTER TO DEVICE TABLE
000001       MAX    EQU     PTR+1             MAX NUMBER IN EXISTANCE
000002       SPARE  EQU     MAX+1             SPARE
             *
             *
             *    DEVICE HEADER GENERATING MACRO
             *
             DEVHDR MACRO   NAME,MAX
             #1TAB  ARG     #1
                    VFD     36/#2
                    DEC     0                 SPARE
                    ENDM    DEVHDR
```

```
                            2920 *
                            2921 *
                            2922 *          HERE IS THE LIST OF DEVICE HEADERS.
                            2923 *
                            2924 *
                   003410   2925            USE     STORE
                   003410   2926 DEVHR      BSS     0                START OF DEVICE HEADER LIST
                            2927 *
                            2928 *                  CARD PUNCHES (CP)
                            2929 *
                   003410   2930            DEVHDR  CP,CPMAX
 END OF BINARY CARD LPCP0037
   003410  003416 0000 00        CPTAB      ARG     CP
   003411  000000000001                     VFD     36/CPMAX
   003412  000000000000                     DEC     0
                            2931 *
                            2932 *                  LINE PRINTERS (LP)
                            2933 *
                   003413   2934            DEVHDR  LP,LPMAX
   003413  003421 0000 00        LPTAB      ARG     LP
   003414  000000000002                     VFD     36/LPMAX
   003415  000000000000                     DEC     0
                   001547   2935            USE     PREVIOUS
```

```
001547          USE     CODE
                HEAD    R
         *
         *
         *                                  PERIPHERAL DEVICE TABLE
         *
         *      THERE IS A ONE-TO-ONE CORRESPONDENCE BETWEEN A 'DEVICE' AND
         *      PHYSICAL DEVICE IN THE MACHINE ROOM.  THE DEVICE CONTAINS THE
         *      NAME OF THE DEVICE. THE FRN WHEN OPEN. FLAG BITS TELLING ITS
         *      CURRENT STATUS, AND IF BUSY. WHO IS RESPONSIBLE.
         *
         *
         *      FORMAT OF PERIPHERAL DEVICE
         *
         *
000000   NAME   EQU     0                   FOUR CHAR ASCII ABBREVIATION
000001   FRN    EQU     NAME+1              (UPPER) FRN OF PERIPHERAL WHEN OPEN
000002   FLAG   EQU     FRN+1               (UPPER) FLAG BITS FOR THE PERIPHERAL
000002   ALLC   EQU     FLAG                (LOWER) JOB NUMBER USING IT WHEN BUSY
000003   DEVLN  EQU     ALLC+1-NAME         DEVICE ENTRY LENGTH
         *
         *
         *      BITS FOR FLAG
         *
400000   BUSY   EQU     B$SIGN              ON IF NOT ALLOCATABLE
200000   CLOSE  EQU     BUSY/2              ON IF CLOSED
100000   RSVE   EQU     CLOSE/2             ON IF OPERATOR REQUESTED A CLOSE
         *
         *
         *      DEVICE GENERATING MACRO
         *
         DEVICE MACRO   NAME
                UASCI   1,#1                NAME
         BUGBUG SET     BUGBUG+1
                ZERO    BUGBUG,TT           FRN/ UNIT NUMBER
                ZERO    BUSY+CLOSE,0
                ENDM    DEVICE
         *
         *
         DEVT   MACRO   NAME,(LIST OF DEVICE NUMBERS)
         #1     BSS     0
         TT     SET     0                   INITIALIZE COUNTER
                IDRP    #2
         SET    SET     #2
                DEVICE  #1#2
         TT     SET     TT+1
                IDRP
         #1MAX  SET     SET+1
                ENDM    DEVT
```

```
                           2987 *
                           2988 *
                           2989 *        HERE IS THE TABLE OF DEVICES
                           2990 *
                           2991 *
                  003416   2992          USE      STORE
                  003416   2993 DEVTB    BSS      0                START OF DEVICE TABLE
                           2994 *
                           2995 *                 CARD PUNCHES (CP)
                           2996 *
                  003416   2997          DEVT     CP,(00)
                  003416        CP       BSS      0
                  000000        TT       SET      0
                  000000        SET      SET      00
                  003416                 DEVICE   CP00
003416  103120060060                     UASCI    1,CP00           NAME
                  525234        BUGBUG   SET      BUGBUG+1
003417  525234 000000                    ZERO     BUGBUG,TT
003420  600000 000000                    ZERO     BUSY+CLOSE,0
                  000001        TT       SET      TT+1
                  000001        CPMAX    SET      SET+1
                           2998 *
                           2999 *                 LINE PRINTERS (LP)
                           3000 *
                  003421   3001          DEVT     LP,(00,01)
                  003421        LP       BSS      0
                  000000        TT       SET      0
                  000000        SET      SET      00
                  003421                 DEVICE   LP00
003421  114120060060                     UASCI    1,LP00           NAME
                  525235        BUGBUG   SET      BUGBUG+1
003422  525235 000000                    ZERO     BUGBUG,TT
003423  600000 000000                    ZERO     BUSY+CLOSE,0
                  000001        TT       SET      TT+1
                  000001        SET      SET      01
                  003424                 DEVICE   LP01
003424  114120060061                     UASCI    1,LP01           NAME
                  525236        BUGBUG   SET      BUGBUG+1
003425  525236 000001                    ZERO     BUGBUG,TT
003426  600000 000000                    ZERO     BUSY+CLOSE,0
                  000002        TT       SET      TT+1
                  000002        LPMAX    SET      SET+1
                  001547   3002          USE      PREVIOUS
```

```
                                      HEAD    J
                                *
                                *
                                *                        JOB TABLE
                                *
                                *     THE JOB TABLE IS A TABLE OF ONE WORD ENTRIES, INDEXED
                                *     BY THE JOB NUMBER.  INFORMATION IN THE JOB TABLE IS
                                *     THAT WHICH MUST BE LOCATED OR MATCHED FOR CONSISTENCY
                                *     CHECKS AND FOR DEBUGGING PURPOSES.  THE TABLE CONTAINS
                                *     MAXJB ENTRIES.
                                *
                    003427            USE     STORE
                    000020      MAXJB EQU     16                 MAXIMUM NUMBER OF JOBS IN THE SYSTEM
                                *
                                *
                    003430            EIGHT                      TO MAKE DEBUGGING EASIER
                    003430      JTAB  BSS     0                  JOB TABLE
                    003430            DUP     1,MAXJB            (UPPER) PTR TO JCB/ (LOWER) PTR TO DEVICE
   003430  000000 000000              ZERO    0,0                INITIALLY OFF
 END OF BINARY CARD LPCP0038
                    001547            USE     PREVIOUS
```

```
         J                              EXIT

                    001547        3025         USE     CODE
                                  3026         HEAD
                                  3027 *
                                  3028 *
                                  3029 *                               EXIT
                                  3030 *
                                  3031 *       COMPLETE ONE TASK AND BEGIN ANOTHER FROM Q$TASK QUEUE
                                  3032 *
                                  3033 *       ENTER BY
                                  3034 *               TRA $EXIT
                                  3035 *       RETURN TO NEXT TASK
                                  3036 *       RETURNS WITH
                                  3037 *               C(J) = JOB NUMBER
                                  3038 *               C(T) = TRAP BLOCK
                                  3039 *               C(L) = TASK-START-ADDRESS
                                  3040 *
                    001547        3041 EXIT    BSS     0                   ENTRY POINT
                    001547        3042         CKPT                        DEBUGGING
  001547  000474 7170 00                       XED     X$CKPT
                                  3043         INHIB   SAVE,ON             LOCK OUT UNWANTED INTERRUPTS
                    001550        3044 EXIT1   BSS     0                   TIME TO FOOL WITH THE Q$TASK QUEUE
  001550  003261 6202 00          3045         EAX     0,Q$FIRST+1+Q$TASK  ADDRESS OF FIRST ELEMENT
  001551  003261 1002 00          3046         CMPX    0,Q$LAST+Q$TASK  *DOES LAST POINT TO IT?
  001552  001610 6002 00          3047         TZE     WAIT
  001553  003260 2212 00          3048         LDX     T,Q$FIRST+Q$TASK  *OFFSET POINTER TO BLOCK
  001554  777777 6002 00          3049         TZE     $ERROR              ***PROBLEM
  001555  003261 1012 00          3050         CMPX    T,Q$LAST+Q$TASK  *IS THIS LAST?
  001556  001561 6012 00          3051         TNZ     *+3                 NO
END OF BINARY CARD LPCP0039
  001557  003261 6202 00          3052         EAX     0,Q$FIRST+1+Q$TASK  YES, SET THIS QUEUE
  001560  003261 7402 00          3053         STX     0,Q$LAST+Q$TASK  *TO EMPTY STATUS
  001561  000004 1212 03          3054         SBLX    T,Q$OFFST,DU        RELATE THE BEGINNING OF BLOCK
  001562  777777 6002 00          3055         TZE     $ERROR              ***DBG
  001563  777777 6042 00          3056         TMI     $ERROR              ***DBG
  001564  000003 2222 11          3057         LDX     X,Q$LINK,T          GET OFFSET POINTER TO NEXT BLOCK
  001565  003260 7422 00          3058         STX     X,Q$FIRST+Q$TASK  *AND MAKE IT NOW FIRST
                                  3059         INHIB   RESTORE             RESUME NORMAL TELECAST
  001566  000006 7260 11          3060         LXL     J,T$JCB,T           RESTORE JCB POINTER
  001567  777777 6000 00          3061         TZE     $ERROR              ***DBG
  001570  777777 6040 00          3062         TMI     $ERROR              ***DBG
  001571  000004 2270 11          3063         LDX     L,T$TRA,T           AND TRANSFER ADDRESS
  001572  777777 6000 00          3064         TZE     $ERROR              ***DBG
  001573  777777 6040 00          3065         TMI     $ERROR              ***DBG
                    001574        3066         BUGU    (T$TRA,T)           BUG RETURN
                    525237             BUGBUG SET     BUGBUG+1
  001574  525237 2200 03                       LDX     0,BUGBUG,DU
  001575  000004 7400 11                       STX     0,T$TRA,T
                    001576        3067         BUGXR   (0,X,Y,Z,Q)         BUG THE REGISTERS
                    525240             BUGBUG SET     BUGBUG+1
  001576  525240 2200 03                       LDX     0,BUGBUG,DU
```

```
                                              EXIT

     001577  525240 2220 03                    LDX    X,BUGBUG,DU
     001600  525240 2230 03                    LDX    Y,BUGBUG,DU
     001601  525240 2240 03                    LDX    Z,BUGBUG,DU
     001602  525240 2250 03                    LDX    Q,BUGBUG,DU
                       001603      3068        BUGA
                       525241           BUGBUG SET    BUGBUG+1
     001603  525241 2350 03                    LDA    BUGBUG,DU
     001604  525241 2750 07                    ORA    BUGBUG,DL
                       001605      3069        BUGQ
                       525242           BUGBUG SET    BUGBUG+1
  END OF BINARY CARD LPCP0040
     001605  525242 2360 03                    LDQ    BUGBUG,DU
     001606  525242 2760 07                    ORQ    BUGBUG,DL
     001607  000000 7100 17        3070        TRA    0,L              AND AWAY WE GO!
                                   3071 *
                                   3072 *
                                   3073 *             WAIT FOR SOMETHING TO HAPPEN
                                   3074 *
                                   3075 *
                                   3076        INHIB  SAVE,ON
     001610  000017 2202 03        3077 WAIT   LDX    0,$,PAUSE,DU     PAUSE AND START AT
     001611  000000 0012 00        3078        MME                     ANY INTERRUPT
                                   3079        INHIB  RESTORE
     001612  001550 7100 00        3080        TRA    EXIT1            SKIP CHECKPOINT
```

```
001613            USE      CODE
                  HEAD     C
         *
         *
         *        COMMUNICATIONS NETWORK STRUCTURE
         *
         *             THERE EXEISTS A PRIVATE COMMUNICATIONS NETWORK AMONG THE
         *        THE MONITOR AND ALL OF ITS SUB-MODULES (I.E. PERIPHERAL DRIVERS).
         *        AT STARTUP TIME FOR THE MONITOR, IT CREATES THE NETWORK BY
         *        OPENING THREE SCRATCH EVENTS AND PASSING A FRN OF EACH TO EACH
         *        SUB-MODULES SPAWNED.  FOR THE SUB-MODULES, THESE EVENTS ARE
         *        REFERENCED BY CANONICAL NUMBERS:
         *
         *                 $FRN0 --
         *        THIS IS THE COMMAND EVENT FOR THE DRIVERS.  EACH DRIVER IS
         *        ALLOWED NOTIFY ACCESS ONLY.  COMMANDS ARE CHANNELED TO THE
         *        SPECIFIED SUB-MODULE BY THE STATE WHEN CAUSED.
         *
         *                 $FRN1 --
         *        THIS IS THE COMMAND REPLY EVENT FOR THE DRIVERS.  EACH DRIVER
         *        IS ALLOWED CAUSE ACCESS ONLY.  IN ORDER TO INFORM THE
         *        MONITOR THE PERIPHERAL DRIVER CAUSES THIS EVENT WITH ITS STATE.
         *
         *                 $FRN2 --
         *        AS IMPLIED ABOVE, $FRN0 AND $FRN1 ARE AN INPUT/OUTPUT PAIR.
         *        $FRN2 HOWEVER IS NOT PAIRED AT ALL.  THE MONITOR USES THIS
         *        EVENT AS A PASS EVENT SENDING FILES TO BE PROCESSED AND DEVICES
         *        DOWN TO ITS SONS.  THE SONS NEVER EVER PASS ANYTHING BACK TO
         *        THE FATHER.  THEY SIMPLY CLOSE FILES.
         *
         *
         *        MESSAGE FORMATS:  RETURNED IN T$SRW2.T (UPPER)
         *
         *        FOR $FRN2 --
         *                 BITS 0 - 3 = JOB NUMBER (WITH 0 ILLEGAL)
         *                 BITS 4     = BANNER (ON MEANS SUPPLY BANNER)
         *                 BITS 5     = OUTPUT MODE:  512/ 320 (ON MEANS 320)
         *                 BITS 6 -17 = START ADDRESS (IN ELEMENTS)
         *
         *        FOR $FRN0 --
         *                 BITS 0 - 3 = JOB NUMBER (MUST BE ZERO)
         *                 BITS 4 -14 = <NOT USED>
         *                 BITS 15-17 = DEVICE UNIT NUMBER (0 THRU 7)
```

```
         C                                 COMMUNICATIONS -- SEND MESSAGE

                          001613     3126          USE      CODE
                                     3127          HEAD     C
                                     3128 *
                                     3129 *
                                     3130 *                                  SEND A MESSAGE
                                     3131 *
                                     3132 *        THIS TASK SENDS A 36 BIT MESSAGE, WHICH IS BIT CODED TO THE
                                     3133 *        MONITOR ON THE OUTPUT EVENT FILE $FRN1.  THE STATE AND
                                     3134 *        MESSAGE ARE PASSED TO IT IN T$TEMP1,T AND T$TEMP2,T,
                                     3135 *        RESPECTIVELY.  AFTER SUCCESFULLY TRANSMITTING THE MESSAGE
                                     3136 *        THE TASK EVAPORATES.
                                     3137 *
                                     3138 *        ENTER WITH
                                     3139 *                 C(XT) = TBLOCK-ADDRESS
                                     3140 *                 C(T$TEMP1,T) = STATE
                                     3141 *                 C(T$TEMP2,T) = MESSAGE
                                     3142 *        SINCE AN ASYNCHRONOUS TASK, IT CAN USE ALL TEMP'S
                                     3143 *        CALLS ONLY  R$RELT  WHEN DONE.
                                     3144 *
                          001613     3145 MESSX    BSS      0                  ENTRY POINT
                          001613     3146          CAUSE    ($FRN1,DU),(1,DU),(T$TEMP1,T),(T$TEMP2,T),(0,DU),(0,DU)
    001613  000657 7000 00                         TSX      0,$CAUSE
    001614  000001 0000 03                         ARG      $FRN1,DU
    001615  000001 0000 03                         ARG      1,DU
    001616  000027 0000 11                         ARG      T$TEMP1,T
    001617  000026 0000 11                         ARG      T$TEMP2,T
    001620  000000 0000 03                         ARG      0,DU
    001621  000000 0000 03                         ARG      0,DU
                          001622     3147          CHECK    MESS1,B$BZ,MESSX
    001622  000000 7200 11                         LXL      0,T$SRW1,T
    001623  000077 3600 03                         ANX      0,B$STMK,DU
    001624  001630 6000 00                         TZE      MESS1
    001625  000003 1000 03                         CMPX     0,B$BZ,DU
    001626  001613 6000 00                         TZE      MESSX
    001627  777777 7100 00                         TRA      $ERROR
    001630  000000 2220 11           3148 MESS1    LDX      X,T$SRW1,T         GET NUMBER OF PEOPLE NOTIFIED
    001631  001613 6000 00           3149          TZE      MESSX              NONE, RE-SEND
                                     3150 *
                                     3151 *        MESSAGE SENT
                                     3152 *
                          001632     3153          RELT                        RELEASE TRAP BLOCK
  END OF BINARY CARD LPCP0041
    001632  001335 7000 00                         TSX      0,T$RELT
                          001633     3154          EXIT                        EVAPORATE
    001633  001547 7100 00                         TRA      $EXIT
```

```
                001634    3156          USE     CODE
                          3157          HEAD    C
                          3158 *
                          3159 *                                CTRAP SERVICE
                          3160 *
                          3161 *        THIS SUBROUTINE IS ENTERED WHENEVER AN OUTSTANDING
                          3162 *        NOTIFY IS CAUSED.  THE ROUTINE ASCERTAINS THE REASON
                          3163 *        FOR THE CAUSE AND TRANSFERS CONTROL TO THE AP-
                          3164 *        PRIATE SUBROUTINE.
                          3165 *
                          3166 *        CALL WITH
                          3167 *                C(XT) = CTRAP ADDRESS
                          3168 *        CALLS
                          3169 *                C$NSRVX  (TO RE-ISSUE NOTIFY)
                          3170 *                OR APPROPIATE SUBROUTINE
                          3171 *
                001634    3172 NSRV     BSS     0
                001634    3173          CHECK   NSRV1,B$TLE,NSRVX
001634  000000 7200 11                  LXL     0,T$SRW1,T
001635  000077 3600 03                  ANX     0,B$STMK,DU
001636  001642 6000 00                  TZE     NSRV1
001637  000011 1000 03                  CMPX    0,B$TLE,DU
001640  001644 6000 00                  TZE     NSRVX
001641  777777 7100 00                  TRA     $ERROR
                001642    3174 NSRV1    BSS     0                 IT WAS REALLY CAUSED
001642  000005 7270 11    3175          LXL     L,C$RLINK,T       WELL, WHO GETS IT?
001643  000000 7100 17    3176          TRA     0,L               LET HIM HANDLE IT
```

C COMMUNICATIONS -- RE-ISSUE NOTIFY

```
                    001644              USE     CODE
                                        HEAD    C
                                   *
                                   *
                                   *                            RE-ISSUE NOTIFY
                                   *
                                   *    THIS SUBROUTINE SIMPLY RE-ISSUES THE NOTIFY FOR
                                   *    ANY NCB (NOTIFY CONTROL BLOCK).  AFTER A SUCCESSFUL OPERATION
                                   *    THE TASK EVAPORATES.
                                   *
                                   *    CALL WITH
                                   *            C(XT) = NCB ADDRESS
                                   *
                    001644         NSRVX BSS    0
                    001644              GETT                    GET A TRAP BLOCK
 001644 001325 7000 00                  TSX     0,T$GETT
 001645 000000 6220 11                  EAX     X,0,T           SAVE A PTR TO IT
 001646 000005 2210 11                  LDX     T,T$LINK,T      LET T POINT TO ORIGINAL TCB
 001647 001634 6200 00                  EAX     0,NSRV          RESTART ADDRESS
                    001650              SETUP                   RESTART CTRAP
 001650 000510 7170 00                  XED     $SETUP
 001651 000000 6210 12                  EAX     T,0,X           GET BACK PTR TO NEW TCB
 001652 000005 2230 11             NSX1 LDX     3,T$LINK,T      MAKE X3 POINT TO NCB
                    001653              NOTIF   (ERN,3),(STATE,3)
 001653 000646 7000 00                  TSX     0,$NOTIF
 001654 000030 0000 13                  ARG     ERN,3
 001655 000031 0000 13                  ARG     STATE,3
                    001656              CHECK   NSX2,B$BZ,NSX1
END OF BINARY CARD LPCP0042
 001656 000000 7200 11                  LXL     0,T$SRW1,T
 001657 000077 3600 03                  ANX     0,B$STMK,DU
 001660 001664 6000 00                  TZE     NSX2
 001661 000003 1000 03                  CMPX    0,B$BZ,DU
 001662 001652 6000 00                  TZE     NSX1
 001663 777777 7100 00                  TRA     $ERROR
                    001664         NSX2 BSS     0               SUCCESSFULLY RE-ISSUED NOTIFY
                    001664              RELT                    RELEASE TBLOCK
 001664 001335 7000 00                  TSX     0,T$RELT
                    001665              EXIT                    EXIT
 001665 001547 7100 00                  TRA     $EXIT
```

```
         C                              COMMUNICATIONS -- NOTIFY CONTROL BLOCKS
                                3205 *
                                3206 *
                                3207 *                                   NOTIFY CONTROL BLOCKS
                                3208 *
                                3209 *                         LINE PRINTER COMMAND EVENT
                                3210 *
                       001666   3211              NCB    LPNCB0,C$COMD,$FRN0,$LPST,0,R$TYPLP,B$LP
                       003450                     USE    STORE
                       003450                     EIGHT
                       003450             LPNCB0  BSS    0
      003450  000000 000035                       ZERO   0,B$TRO
      003451  000000 000000                       ZERO   0,0
      003452  000000 000000                       ZERO   0,0
      003453  000000 000000                       ZERO   0,0
      003454  000000 000000                       ZERO   **,0
      003455  000000 001666                       ZERO   0,C$COMD
      003456  003450 003450                       ZERO   *-C$NCB,*-C$JCB
      003457  000000000000                        DEC    0
                       003460                     DUP    1,16
      003460  000000000000                        DEC    0
   END OF BINARY CARD LPCP0043
      003500  000000 000000                       ZERO   $FRN0,
      003501  003000 000000                       ZERO   $LPST,0
      003502  000000000000                        VFD    36/0
      003503  000002 000400                       ZERO   R$TYPLP,B$LP
                       001666                     USE    PREVIOUS
                                3212 *
                                3213 *                         LINE PRINTER PASS FILE EVENT
                                3214 *
                       001666   3215              NCB    LPNCB2,$INIT,$FRN2,$LPST,0,R$TYPLP,B$LP
                       003504                     USE    STORE
                       003510                     EIGHT
                       003510             LPNCB2  BSS    0
      003510  000000 000035                       ZERO   0,B$TRO
      003511  000000 000000                       ZERO   0,0
      003512  000000 000000                       ZERO   0,0
      003513  000000 000000                       ZERO   0,0
   END OF BINARY CARD LPCP0044
      003514  000000 000000                       ZERO   **,0
      003515  000000 002023                       ZERO   0,$INIT
      003516  003510 003510                       ZERO   *-C$NCB,*-C$JCB
      003517  000000000000                        DEC    0
                       003520                     DUP    1,16
      003520  000000000000                        DEC    0
      003540  000002 000000                       ZERO   $FRN2,
      003541  003000 000000                       ZERO   $LPST,0
   END OF BINARY CARD LPCP0045
      003542  000000000000                        VFD    36/0
      003543  000002 000400                       ZERO   R$TYPLP,B$LP
                       001666                     USE    PREVIOUS
```

```
                                 3217 *
                                 3218 *
                                 3219 *                 CARD PUNCH COMMAND EVENT
                                 3220 *
                        001666   3221          NCB      CPNCB0,C$COMD,$FRN0,$CPST,0,R$TYPCP,B$CP
                        003544                 USE      STORE
                        003550                 EIGHT
                        003550         CPNCB0  BSS      0
   003550  000000 000035                       ZERO     0,B$TR0
   003551  000000 000000                       ZERO     0,0
   003552  000000 000000                       ZERO     0,0
   003553  000000 000000                       ZERO     0,0
   003554  000000 000000                       ZERO     **,0
   003555  000000 001666                       ZERO     0,C$COMD
   003556  003550 003550                       ZERO     *-C$NCB,*-C$JCB
   003557  000000000000                        DEC      0
                        003560                 DUP      1,16
   003560  000000000000                        DEC      0
 END OF BINARY CARD LPCP0046
   003600  000000 000000                       ZERO     $FRN0,
   003601  001000 000000                       ZERO     $CPST,0
   003602  000000000000                        VFD      36/0
   003603  000001 000200                       ZERO     R$TYPCP,B$CP
                        001666                 USE      PREVIOUS
                                 3222 *
                                 3223 *
                                 3224 *                 CARD PUNCH PASS FILE EVENT
                                 3225 *
                        001666   3226          NCB      CPNCB2,$INIT,$FRN2,$CPST,0,R$TYPCP,B$CP
                        003604                 USE      STORE
                        003610                 EIGHT
                        003610         CPNCB2  BSS      0
   003610  000000 000035                       ZERO     0,B$TR0
   003611  000000 000000                       ZERO     0,0
   003612  000000 000000                       ZERO     0,0
   003613  000000 000000                       ZERO     0,0
   003614  000000 000000                       ZERO     **,0
   003615  000000 002023                       ZERO     0,$INIT
   003616  003610 003610                       ZERO     *-C$NCB,*-C$JCB
   003617  000000000000                        DEC      0
                        003620                 DUP      1,16
 END OF BINARY CARD LPCP0047
   003620  000000000000                        DEC      0
   003640  000002 000000                       ZERO     $FRN2,
   003641  001000 000000                       ZERO     $CPST,0
   003642  000000000000                        VFD      36/0
   003643  000001 000200                       ZERO     R$TYPCP,B$CP
                        001666                 USE      PREVIOUS
```

```
                  001666     3228         USE      CODE
                             3229         HEAD     C
                             3230 *
                             3231 *
                             3232 *                          COMMANDS
                             3233 *
                             3234 *       THIS ROUTINE DECODES A COMMAND SENT TO IT FROM THE MONITOR.
                             3235 *       IF A LEGAL COMMAND, THE APPROPRIATE ACTION IS TAKEN.  IF
                             3236 *       ILLEGAL, WELL IT IS INGORED (MORE OFTEN THAN NOT)
                             3237 *
                             3238 *
                  001666     3239 COMD    BSS      0               ENTER HERE FROM C$NSRV
   001666  000001 2360 11    3240         LDQ      T$SRW2,T        GET COMMAND
 END OF BINARY CARD LPCP0048
   001667  000012 7720 00    3241         QRL      18-4-4          RIGHT JUSTIFY IN QU
   001670  000017 3760 03    3242         ANQ      =017,DU         MASK TO COMMAND
   001671  000006 1160 03    3243         CMPQ     CMAX,DU         TEST VALIDITY OF COMMAND
   001672  001702 6030 00    3244         TRC      COMDX           NOPE; EXIT
   001673  001674 7100 22    3245         TRA      *+1,QU*         BRANCH TO SUBROUTINE
   001674  777777 0000 00    3246 CMDTB   ARG      $ERROR          0 = ILLEGAL
   001675  001703 0000 00    3247         ARG      GET             1 = GET PERIPHERAL
   001676  001714 0000 00    3248         ARG      KILL            2 = KILL PERIPHERAL NOW
   001677  001724 0000 00    3249         ARG      REL             3 = RELEASE PERIPHERAL WHEN NOT BUSY
   001700  777777 0000 00    3250         ARG      $ERROR          4 = SPARE
   001701  001747 0000 00    3251         ARG      RSTRT           5 = RESTART A PERIPHERAL
                  000006     3252 CMAX    EQU      *-CMDTB         NUMBER OF COMMANDS
                             3253 *
                             3254 *
                             3255 *
                             3256 *                          COMDX
                             3257 *
                             3258 *       RE-ISSUE NOTIFY ON THIS NCB
                             3259 *
                  001702     3260 COMDX   BSS      0
   001702  001644 7100 00    3261         TRA      NSRVX           RE-ISSUE THE NOTIFY
```

```
         C                               COMMUNICATIONS -- COMMAND GET

                    001703      3263          USE     CODE
                                3264          HEAD    C
                                3265 *
                                3266 *
                                3267 *                                  COMMAND GET
                                3268 *
                                3269 *        THIS SUBROUTINE 'GETS' THE SPECIFIED PERIPHERAL.
                                3270 *
                                3271 *        ENTER WITH
                                3272 *                C(XT) = NCB-ADDRESS
                                3273 *
                    001703      3274 GET      BSS     0
    001703  002005 7070 00      3275          TSX     L.PERI              GET PERIPHERAL INFORMATION
                                3276 *
                                3277 *        CHECK IF CURRENTLY CLOSED
                                3278 *
    001704  000002 2350 14      3279          LDA     R$FLAG.Z            GET PERIPHERAL FLAGS
    001705  200000 3150 03      3280          CANA    R$CLOSE.DU          CHECK IF CLOSED
    001706  001702 6000 00      3281          TZE     COMDX               IGNORE COMMAND IF NOT
    001707  000000 2220 11      3282          LDX     X.T$SRW1.T          GET PASSED FRN
    001710  000001 7420 14      3283          STX     X.R$FRN.Z           SAVE IN TABLE
    001711  077777 2220 03      3284          LDX     X.-1-R$BUSY-R$CLOSE-R$RSVE.DU  SET OFF A FEW BITS
    001712  000002 3420 14      3285          ANSX    X.R$FLAG.Z          IN THE PERIPHERAL TABLE
    001713  001702 7100 00      3286          TRA     COMDX               AND EXIT
```

```
                   001714                    USE     CODE
                                             HEAD    C
                                   *
                                   *
                                   *                              COMMAND KILL
                                   *
                                   *         THIS ROUTINE STOPS THE SPECIFIED PERIPHERAL IMMEDIATELY.
                                   *
                                   *         ENTER WITH
                                   *                 C(XT) = NCB-ADDRESS
                                   *
                   001714            KILL    BSS     0
 END OF BINARY CARD LPCP0049
    001714  002005 7070 00                   TSX     L.PERI          GET PERIPHERAL INFORMATION
                                   *
                                   *         CHECK IF PERIPHERAL IS IN USE
                                   *
    001715  000002 2350 14                   LDA     R$FLAG,Z        GET PERIPHERAL FLAG
    001716  400000 3150 03                   CANA    R$BUSY,DU       CHECK IF BUSY
    001717  001702 6000 00                   TZE     COMDX           EXIT IF NOT
    001720  000002 7260 14                   LXL     J,R$ALLC,Z      GET POINTER TO JCB OF DEVICE OWNER
    001721  400000 2350 07                   LDA     B$KILL,DL       TURN ON THE KILL BIT
    001722  000000 2550 16                   ORSA    J$FLAGS,J       IN THE JOB FLAG WORD
    001723  001702 7100 00                   TRA     COMDX           AND EXIT
```

```
                    001724     3312          USE     CODE
                               3313          HEAD    C
                               3314 *
                               3315 *
                               3316 *                          COMMAND RELEASE
                               3317 *
                               3318 *        THIS ROUTINE RELEASES A PERIPHERAL AS SOON AS IT IS FREE
                               3319 *
                               3320 *        ENTER WITH
                               3321 *                C(XT) = NCB-ADDRESS
                               3322 *
                    001724     3323 REL      BSS     0
001724  002005 7070 00         3324          TSX     L.PERI           GET PERIPHERAL INFORMATION
                               3325 *
                               3326 *        CHECK FOR ALREADY CLOSED
                               3327 *
001725  000002 2350 14         3328          LDA     R$FLAG.Z         GET PERIPHERAL FLAGS
001726  200000 3150 03         3329          CANA    R$CLOSE.DU       CHECK IF CLOSED
001727  001702 6010 00         3330          TNZ     COMDX            EXIT IF SO -- MONITOR SHOULD KNOW IT IS CLOSED
001730  100000 2750 03         3331          ORA     R$RSVE.DU        ASK TO HAVE IT CLOSED
001731  000002 7550 14         3332          STA     R$FLAG.Z         RESTORE FLAGS
001732  400000 3150 03         3333          CANA    R$BUSY.DU        CHECK IF CURRENTLY BUSY
001733  001702 6010 00         3334          TNZ     COMDX            BUSY. IT WILL CLOSE EVENTUALLY
                               3335 *
                               3336 *        DEIVCE IDLE -- SEIZE IT
                               3337 *
                    001734     3338 REL2     BSS     0
001734  400000 2750 03         3339          ORA     R$BUSY.DU        MARK IT BUSY
001735  000002 7550 14         3340          STA     R$FLAG.Z         IN THE PERIPHERAL FLAG WORD
001736  000002 4460 14         3341          SXL     J.R$ALLC.Z       AND ALLOCATED TO US
001737  000006 4460 11         3342          SXL     J.T$JCB.T        FUDGE
001740  000031 2350 11         3343          LDA     C$STATE.T        GET STATE
 END OF BINARY CARD LPCP0050
001741  000011 7710 00         3344          ARL     9                MAP INTO TYPE
001742  000027 7550 11         3345          STA     T$TEMP1.T        SAVE FOR RELEASE
001743  000027 4440 11         3346          SXL     Z.T$TEMP1.T      AND SAVE DEVICE NUMBER
                    001744     3347          RELP    (T$TEMP1.T)      RELEASE IT
001744  000027 2350 11                       LDA     T$TEMP1.T
001745  001433 7070 00                       TSX     L.R$RELP
001746  001702 7100 00         3348          TRA     COMDX            EXIT
```

```
                 001747      3350          USE      CODE
                             3351          HEAD     C
                             3352 *
                             3353 *
                             3354 *                                    COMMAND RESTART
                             3355 *
                             3356 *        THIS COMMAND RESTARTS THE JOB ON THE SPECIFIED PER-
                             3357 *        IPHERAL TO THE SPECIFIED ELEMENT NUMBER.
                             3358 *
                 001747      3359 RSTRT    BSS      0
    001747  002005 7070 00   3360          TSX      L.PERI              GET PERIPHERAL INFORMATION
    001750  000027 7440 11   3361          STX      Z.T$TEMP1.T         SAVE PTR TO DEVICE
                             3362 *
                             3363 *        CHECK IF PERIPHERAL IS IN USE
                             3364 *
    001751  000002 2350 14   3365          LDA      R$FLAG.Z            GET PERIPHERAL FLAG
    001752  400000 3150 03   3366          CANA     R$BUSY.DU           CHECK IF BUSY
    001753  001702 6000 00   3367          TZE      COMDX               EXIT IF NOT
                             3368 *
                             3369 *        HALT READING IN ORDER TO DO SET POINTER
                             3370 *
    001754  000002 7260 14   3371          LXL      J.R$ALLC.Z          GET PTR TO JCB OF DEVICE OWNER
    001755  000006 4460 11   3372          SXL      J.T$JCB.T           SAVE JCB POINTER
    001756  010000 2350 07   3373          LDA      B$HALT.DL           GET THE HALT BIT
    001757  000000 2550 16   3374          ORSA     J$FLAGS.J           MARK READING HALTED
                             3375 *
                             3376 *        SET POINTER TO RESTART ADDRESS REQUEST (I.E. 0)
                             3377 *
    001760  000006 4500 16   3378          STZ      J$RSTRT.J           ***RESET TO BEGINNING FOR NOW
    001761  000026 4500 11   3379          STZ      T$TEMP2.T           FIRST FIND CURRENT SETTING
                 001762      3380 RST1     SPTR     (J$IFRN.J).(T$TEMP2.T)
    001762  000564 7000 00                 TSX      0.$SPTR
    001763  000002 0000 16                 ARG      J$IFRN.J
    001764  000026 0000 11                 ARG      T$TEMP2.T
                 001765      3381          CHECK    RST2.B$BZ.RST1
    001765  000000 7200 11                 LXL      0.T$SRW1.T
 END OF BINARY CARD LPCP0051
    001766  000077 3600 03                 ANX      0.B$STMK.DU
    001767  001773 6000 00                 TZE      RST2
    001770  000003 1000 03                 CMPX     0.B$BZ.DU
    001771  001762 6000 00                 TZE      RST1
    001772  777777 7100 00                 TRA      $ERROR
                 001773      3382 RST2     BSS      0
    001773  000001 2350 11   3383          LDA      T$SRW2.T            GET CURRENT SETTING
    001774  777777 3750 07   3384          ANA      -1.DL               ONLY
    001775  000006 0750 16   3385          ADA      J$RSTRT.J           PLUS THE AMOUNT IT HAS MOVED ON US (ASYNC)
    001776  000000 5310 00   3386          NEG                          COMPUTE REQUESTED MINUS CURRENT FOR CORRECT SIGN
    001777  000000 6350 05   3387          EAA      0.AL                MOVE TO AL
    002000  000026 7550 11   3388          STA      T$TEMP2.T           SAVE FOR SET POINTER
    002001  001762 6010 00   3389          TNZ      RST1                SET PROPERLY
```

C COMMUNICATIONS -- COMMAND RESTART

```
                          *
                          *                    RESET FLAG BITS
                          *
002002  010000 2350 07            LDA     B$HALT,DL      UNSET HALT BIT
002003  000000 6550 16            ERSA    J$FLAGS,J
002004  001702 7100 00            TRA     COMDX          AND EXIT THRU COMMON ROUTINE
```

```
                          002005             USE     CODE
                                             HEAD    C
                                        *
                                        *
                                        *                                PERI
                                        *
                                        *    THIS ROUTINE CHECKS TO SEE IF THE PERIPHERAL NAME EXITS
                                        *    AND IF SO RETURNS A POINTER TO THE DEVICE ENTRY BLOCK.
                                        *    OTHERWISE IT EXITS
                                        *
                                        *    ENTER WITH
                                        *            C(XT) = NCB-ADDRESS
                                        *    RETURN WITH
                                        *            C(XZ) = DEVICE ENTRY BLOCK
                                        *
                          002005          PERI   BSS     0
    002005  000033 2220 11                       LDX     X,C$RES,T          GET PERIPHERAL TYPE
    002006  003074 2200 12                       LDX     0,R$TABLE,X        GET POINTER TO TYPE TABLE
    002007  000000 2240 10                       LDX     Z,R$PTR,0          GET PTR TO FIRST DEVICE OF THIS TYPE
    002010  000001 2350 11                       LDA     T$SRW2,T           GET BACK UNIT NUMBER
    002011  000022 7710 00                       ARL     36-18              RIGHT JUSTIFY IN AL
    002012  000007 3750 07                       ANA     7,DL               MASK TO UNIT NUMBER
END OF BINARY CARD LPCP0052
    002013  000000 2750 14                       ORA     R$NAME,Z           GET NAME
    002014  000001 3360 10                       LCQ     R$MAX,0            GET THE NUMBER OF DEVICES TO CHECK
    002015  000000 1150 14                PERI1  CMPA    R$NAME,Z           TEST FOR MATCH
    002016  000000 6000 17                       TZE     0,L                RETURN WITH Z POINTING TO DEVICE
    002017  000003 0240 03                       ADLX    Z,R$DEVLN,DU       NO. SKIP TO NEXT DEVICE
    002020  000001 0760 07                       ADQ     1,DL               TEST FOR DONE
    002021  002015 6040 00                       TMI     PERI1              NO. LOOP
    002022  001702 7100 00                       TRA     COMDX              INGORE MONITOR
```

C JOB INITIALIZATION ROUTINE

```
                    002023      USE     CODE
                                HEAD
                          *
                          *
                          *                                JOB INIT
                          *
                          *     UPON RECEIPT ON A FILE TO PROCESS. THIS ROUTINE IS ENTERED.
                          *     IT DOES ALL OF THE NECESSARY INITIALIZATION AND CHECKING IN
                          *     ORDER THAT THE JOB MAY BE STARTED.  IT DOES THE FOLLOWING:
                          *
                          *             GET A JCB
                          *             INITIALIZES IT WITH THE ENTIRE JOB DESCRIPTION
                          *             CHECK ELEMENT SIZE
                          *             POSITIONS THE READ POINTER IF NECESSARY
                          *             CREATES THE READ TASK AND THE WRITE TASK
                          *             RE-ISSUES THE NOTIFY
                          *             EXIT
                          *
                          *     ENTER WITH
                          *             C(XT) = NCB-ADDRESS
                          *
                    002023  INIT  BSS     0
 002023  000001 2360 11         LDQ     T$SRW2.T         GET MESSAGE WORD
 002024  000004 7370 00         LLS     4                SELECT JOB NUMBER
 002025  000017 3750 07         ANA     B$BJBMK.DL       ONLY
 002026  001666 6000 00         TZE     C$COMD           TEST IF IT IS A PASSED PERIPHERAL
                    002027      GETJ                     GET A JCB
 002027  001344 7070 00         TSX     L.J$GETJ
 002030  000006 4460 11         SXL     J.T$JCB.T        SAVE JCB PTR
                          *
                          *     INITIALIZE JCB
                          *
 002031  000001 2360 11         LDQ     T$SRW2.T         GET MESSAGE WORD
 002032  000001 7560 16         STQ     J$FLAG.J         SAVE AS IS
 002033  777777 3760 03         ANQ     -1.DU            MASK TO MESSAGE
 002034  000000 7560 16         STQ     J$FLAGS.J        SAVE IN FLAGS. TOO
 002035  000033 7220 11         LXL     X.C$RES.T        GET RESOURCE TYPE
 002036  000000 4420 16         SXL     X.J$FLAGS.J      SAVE
 002037  000004 7370 00         LLS     4                MOVE JOB NUMBER IN A
END OF BINARY CARD LPCP0053
 002040  000017 3750 07         ANA     B$BJBMK.DL       MASK TO JOB NUMBER
 002041  003430 2340 05         SZN     J$JTAB.AL        TEST FOR POOR BOOKKEEPING
 002042  777777 6010 00         TNZ     $ERROR           ONE OF US IS WRONG
 002043  003430 7460 05         STX     J.J$JTAB.AL      MARK IT BUSY
 002044  000001 7370 00         LLS     1                GET HEADER BIT
 002045  000001 3750 07         ANA     B$BHDR.DL        ISOLATE IT
 002046  002051 6000 00         TZE     *+3              NO HEADER NEED
 002047  300000 2350 07         LDA     B$RHDR+B$WHDR.DL *TURN ON HEADER BITS
 002050  000000 2550 16         ORSA    J$FLAGS.J
 002051  000001 7370 00         LLS     1                GET OUTPUT MODE
```

```
    002052  000001 3750 07     3476          ANA    B$BOTMK,DL         ISOLATE IT
    002053  003100 2350 05     3477          LDA    BSZ,AL             GET PROPER BUFFER SIZE
    002054  000005 7550 16     3478          STA    J$BUFSZ,J          SAVE FOR R/W TASKS
                   003100      3479          USE    CONST
    003100  044000 001000      3480 BSZ      ZERO   36*512,512         BITS/ELEMENT // BUFFER SIZE
    003101  026400 000500      3481          ZERO   36*320,320
                   002055      3482          USE    PREVIOUS
    002055  000014 7370 00     3483          LLS    12                 GET RESTART ADDRESS
    002056  007777 3750 07     3484          ANA    B$SSTMK,DL         MASK TO ADDRESS IN ELEMENTS
    002057  000006 7550 16     3485          STA    J$RSTRT,J          SAVE
    002060  000000 2220 11     3486          LDX    X,T$SRW1,T         GET PASSED FRN
    002061  000002 7420 16     3487          STX    X,J$IFRN,J         SAVE AS INPUT FRN
 END OF BINARY CARD LPCP0054
    002062  000002 2360 07     3488          LDQ    J$N,DL             GET NUMBER OF WORKING BUFFERS
    002063  000011 7560 16     3489          STQ    J$EMPTY,J          SET EMPTY AND
    002064  000012 4500 16     3490          STZ    J$FULL,J           FULL COUNTS
    002065  000020 6350 16     3491          EAA    J$RESET,J          POINT TO FIRST WORD OF BUF Q -1
    002066  000013 7550 16     3492          STA    J$RPTR,J           SET READ POINTER
    002067  000014 7550 16     3493          STA    J$WPTR,J           AND WRITE POINTER
    002070  000001 0350 03     3494          ADLA   1,DU               NOW POINT TO FIRST WORD OF BUF Q
    002071  000000 6220 01     3495          EAX    X,0,AU             MOVE TO X
    002072  000020 4420 16     3496          SXL    X,J$RESET,J        WRAP POINTER
                   002073      3497          DUP    2,J$N              RESET BUFFER QUEUE
    002073  000000 4500 12     3498          STZ    0,X                RESET IT
    002074  000001 0220 03     3499          ADLX   X,1,DU             STEP TO NEXT
    002077  000020 7420 16     3500          STX    X,J$RESET,J        END POINTER
                   002100      3501          GETP   (C$RES,T)          GET REQUESTED RESOURCE
    002100  000033 2350 11                   LDA    C$RES,T
    002101  001364 7070 00                   TSX    L,R$GETP
    002102  000016 7550 16     3502          STA    J$RES,J            SAVE RESOURCE POINTERS IN JCB
    002103  000003 7560 16     3503          STQ    J$OFRN,J           SAVE FRN/ UNIT NUMBER IN JCB
                   002104      3504 INIT2    BSS    0
                               3505 *
                               3506 *        CHECK ELEMENT SIZE
                               3507 *
                   002104      3508          RQST   (J$IFRN,J)         REQUEST STATUS
    002104  000575 7000 00                   TSX    0,$RQST
    002105  000002 0000 16                   ARG    J$IFRN,J
                   002106      3509          CHECK  INIT3,B$BZ,INIT2
    002106  000000 7200 11                   LXL    0,T$SRW1,T
    002107  000077 3600 03                   ANX    0,B$STMK,DU
 END OF BINARY CARD LPCP0055
    002110  002114 6000 00                   TZE    INIT3
    002111  000003 1000 03                   CMPX   0,B$BZ,DU
    002112  002104 6000 00                   TZE    INIT2
    002113  777777 7100 00                   TRA    $ERROR
    002114  000000 2220 11     3510 INIT3    LDX    X,T$SRW1,T         GET NUMBER OF BITS/ELEMENT
    002115  000005 1020 16     3511          CMPX   X,J$BUFSZ,J        IS IT WHAT WE EXPECTED?
    002116  777777 6010 00     3512          TNZ    $ERROR             NOPE
                               3513 *
```

```
                                  3514 *       SET READ POINTER
                                  3515 *
 002117  000027 4500 11           3516         STZ     T$TEMP1,T         FIRST FIND CURRENT SETTING
                     002120       3517 INIT4   BSS     0
                     002120       3518         SPTR    (J$IFRN,J),(T$TEMP1,T)
 002120  000564 7000 00                        TSX     0,$SPTR
 002121  000002 0000 16                        ARG     J$IFRN,J
 002122  000027 0000 11                        ARG     T$TEMP1,T
                     002123       3519         CHECK   INIT5,B$BZ,INIT4
 002123  000000 7200 11                        LXL     0,T$SRW1,T
 002124  000077 3600 03                        ANX     0,B$STMK,DU
 002125  002131 6000 00                        TZE     INIT5
 002126  000003 1000 03                        CMPX    0,B$BZ,DU
 002127  002120 6000 00                        TZE     INIT4
 002130  777777 7100 00                        TRA     $ERROR
                     002131       3520 INIT5   BSS     0
 002131  000001 2350 11           3521         LDA     T$SRW2,T          GET CURRENT SETTING
 002132  777777 3750 07           3522         ANA     -1,DL             ONLY
 002133  000006 1750 16           3523         SBA     J$RSTRT,J         MINUS REQUESTED
 002134  000000 5310 00           3524         NEG                       COMPUTE REQUESTED MINUS CURRENT FOR CORRECT SIGN
 002135  000000 6350 05           3525         EAA     0,AL              MOVE TO AU
END OF BINARY CARD LPCP0056
 002136  000027 7550 11           3526         STA     T$TEMP1,T         SAVE FOR SET POINTER
 002137  002120 6010 00           3527         TNZ     INIT4             SET PROPERLY
                                  3528 *
                                  3529 *       CREATE READ TASK
                                  3530 *
                     002140       3531         GETT                      GET A TASK BLOCK
 002140  001325 7000 00                        TSX     0,T$GETT
 002141  000015 7410 16           3532         STX     T,J$RTASK,J       SAVE PTR TO READ TASK
 002142  002304 2220 03           3533         LDX     X,RTASK,DU        GET RETSTART ADDRESS
 002143  000004 7420 11           3534         STX     X,T$TRA,T         SAVE IN TCB
 002144  000004 6200 11           3535         EAX     0,Q$OFFST,T       PREPARE TO START READ TASK
 002145  000005 2220 11           3536         LDX     X,T$LINK,T        GET BACK PTR TO OLD TCB (IE NCB)
 002146  000006 7420 11           3537         STX     X,T$NCB,T         SAVE PTR TO NCB
 002147  000000 6210 12           3538         EAX     T,0,X             RESTORE T
 002150  003262 7170 00           3539         XED     Q$XADD+Q$TASK     PLACE ON THE  MASTER TASK QUEUE
                                  3540 *
                                  3541 *       CREATE WRITE TASK
                                  3542 *
                     002151       3543         GETT                      GET A TCB
 002151  001325 7000 00                        TSX     0,T$GETT
 002152  000015 4410 16           3544         SXL     T,J$WTASK,J       SAVE PTR TO WRITE TASK
 002153  002661 2220 03           3545         LDX     X,WTASK,DU        GET RESTART ADDRESS
 002154  000004 7420 11           3546         STX     X,T$TRA,T         SAVE IN TCB
 002155  000004 6200 11           3547         EAX     0,Q$OFFST,T       PREPARE TO START WRITE TASK
 002156  000005 2220 11           3548         LDX     X,T$LINK,T        GET BACK PTR TO OLD TCB (IE NCB)
 002157  000006 7420 11           3549         STX     X,T$NCB,T         SAVE PTR TO NCB
 002160  000000 6210 12           3550         EAX     T,0,X             RESTORE T
 002161  003262 7170 00           3551         XED     Q$XADD+Q$TASK     PLACE ON THE MASTER TASK QUEUE
```

JOB INITIALIZATION ROUTINE

```
                              3552 *
                              3553 *      SET NEW MEMORY REQUIREMENTS
                              3554 *
    002162  000005 2360 16    3555        LDQ     J$BUFSZ,J        GET BUFFER SIZE
    002163  777777 3760 07    3556        ANQ     -1,DL            ONLY
 END OF BINARY CARD LPCP0057
    002164  000002 4020 07    3557        MPY     J$N,DL           TIMES NUMBER OF BUFFERS
    002165  000060 0760 07    3558        ADQ     2*T$LEN,DL       PLUS TASK CONTROL BLOCKS
    002166  000000 6360 06    3559        EAQ     0,QL             MOVE TO QU
    002167  003321 0560 00    3560        ASQ     $MEMRQ           UPDATE MEMORY REQUIREMENT WORD
                              3561 *
    002170  001644 7100 00    3562        TRA     C$NSRVX          RE-ISSUE NOTIFY
```

```
002171          USE     CODE
                HEAD
           *
           *
           *                               DIJKSTRA IMPLEMENTATION
           *
           *          JOB PROCESSING IS BASED ON THE DIJKSTRA DESIGN MENTIONED
           *    PREVIOUSLY.  IT ENTAILS THE MANAGEMENT OF COOPERATING SE-
           *    QUENTIAL PROCESSES.  (SEE P- AND V- MACROS.)  EACH JOB
           *    TO BE PROCESSED IS REPRESENTED BY TWO ASYNCHRONOUS TASKS:
           *    THE 'READ TASK' (HEREAFTER CALLED RTASK) AND THE 'WRITE TASK'
           *    (HEREAFTER CALLED WTASK).  THE TWO RUN IN PARALLEL.  THE
           *    FLOWCHARTS ARE THE FOLLOWING:
           *
           *            RTASK              WTASK
           *            -----              -----
           *            P(EMPTY)           P(FULL)
           *            FILL               EMPTY
           *            V(FULL)            V(EMPTY)
           *            LOOP RTASK         LOOP WTASK
           *            - - - - - -        - - - - - -
           *            EXIT               EXIT
```

READ TASK -- FILL

```
                           002171              USE      CODE
                                               HEAD
                                          *
                                          *
                                          *                              FILL
                                          *
                                          *    FILL FILLS AN INPUT BUFFER.
                                          *
                                          FILL MACRO    <NO-ARGUMENTS>
                                               TSX      L.$FILL          CALL SUBROUTINE
                                               ENDM     FILL
                                          *
                                          *
                                          *             FILL -- SUBROUTINE
                                          *
                                          *    THIS SUBROUTINE DOES THE ACTUAL READING OF THE INPUT FILE.
                                          *    IT IS CHARGED WITH ALLOCATING THE CORRECT SIZE BUFFER AND
                                          *    WITH READING THE RIGHT NUMBER OF ELEMENTS.  IT HANDLES ALL
                                          *    ERRORS AND RETURNS THE STATUS IN J$FLAG.J FOR THE CALLER.
                                          *
                                          *    ENTER WITH
                                          *             C(XJ) = JCB-ADDRESS
                                          *             C(XT) = TBLOCK-ADDRESS
                                          *    ENTER BY
                                          *             TSX L.FILL
                                          *    CALLS
                                          *             $READ
                                          *    EXIT TO 0.L
                                          *    USES
                                          *             NO LOCALS
                                          *             T$TEMP1
                                          *
                           002171         FILL BSS      0
 002171  000004 4470 11                        SXL      L.T$TRA.T        SAVE RETURN ADDRESS
                           002172              GETC     (J$BUFSZ.J)      ALLOCATE AN INPUT BUFFER
 002172  000005 2350 16                        LDA      J$BUFSZ.J
 002173  001011 7070 00                        TSX      L.R$GETC
 002174  000013 2220 16                        LDX      X.J$RPTR.J       GET OLD BUFFER POINTER
 002175  000001 0220 03                        ADLX     X.1.DU           BUMP TO NEW
 002176  000020 1020 16                        CMPX     X.J$RESET.J      TEST FOR WRAP
 002177  002201 6020 00                        TNC      *+2              OK
 002200  000020 7220 16                        LXL      X.J$RESET.J      WRAP
 002201  000013 7420 16                        STX      X.J$RPTR.J       SAVE NEW POINTER
 002202  000000 7550 12                        STA      0.X              SAVE BUFFER ADDRESS
 002203  000027 7550 11                        STA      T$TEMP1.T        AND HERE FOR EASY ACCESS
 002204  000000 2350 16                   FILL0 LDA     J$FLAGS.J        GET STATUS BITS
 002205  010000 3150 07                        CANA     B$HALT.DL        ARE WE TO HALT READING?
 002206  002277 6010 00                        TNZ      FILL6            YES. HALT FOR AWHILE
 002207  400000 3150 07                        CANA     B$KILL.DL        ARE WE TO KILL THIS JOB?
 002210  002230 6010 00                        TNZ      FILL2            YES. SO DO IT
```

READ TASK -- FILL

```
 END OF BINARY CARD LPCP0058
     002211  200000 3150 07          3635         CANA     B$RHDR,DL          ARE WE SUPPOSED TO OUTPUT A HEADER?
     002212  002256 6010 00          3636         TNZ      FILL4              YES, SO DO IT
                                     3637 *
                                     3638 *
                                     3639 *        INITIATE A READ
                                     3640 *
                     002213          3641 FILL1   READ     (J$IFRN,J),(T$TEMP1,T),(1,DU),(0,DU)
     002213  000536 7000 00                       TSX      0,$READ
     002214  000002 0000 16                       ARG      J$IFRN,J
     002215  000027 0000 11                       ARG      T$TEMP1,T
     002216  000001 0000 03                       ARG      1,DU
     002217  000000 0000 03                       ARG      0,DU
                     002220          3642         CHECK    FILL3,B$BZ,FILL0,B$EOF,FILL2
     002220  000000 7200 11                       LXL      0,T$SRW1,T
     002221  000077 3600 03                       ANX      0,B$STMK,DU
     002222  002236 6000 00                       TZE      FILL3
     002223  000003 1000 03                       CMPX     0,B$BZ,DU
     002224  002204 6000 00                       TZE      FILL0
     002225  000016 1000 03                       CMPX     0,B$EOF,DU
     002226  002230 6000 00                       TZE      FILL2
     002227  777777 7100 00                       TRA      $ERROR
                                     3643 *
                                     3644 *       INFORMATION READ
                                     3645 *
                     002230          3646 FILL2   BSS      0                  EOF REACHED
                     002230          3647         RELC     (T$TEMP1,T)        RELEASE THE BUFFER
     002230  000027 2350 11                       LDA      T$TEMP1,T
     002231  001110 7070 00                       TSX      L,R$RELC
     002232  000013 2220 16          3648         LDX      X,J$RPTR,J         GET POINTER TO INPUT BUFFER
     002233  000000 4500 12          3649         STZ      0,X                MARK AS SUCH
     002234  040000 2350 07          3650         LDA      B$ENDR,DL          GET EOF FLAG FOR READ TASK
     002235  000000 2550 16          3651         ORSA     J$FLAGS,J          MARK AS SUCH
                     002236          3652 FILL3   BSS      0                  SUCCESSFUL READ
     002236  000004 7270 11          3653         LXL      L,T$TRA,T          RETRIEVE RETURN ADDRESS
                     002237          3654         BUGA                        BUG ALL REGISTERS EXCEPT T AND J
                     525243               BUGBUG SET      BUGBUG+1
 END OF BINARY CARD LPCP0059
     002237  525243 2350 03                       LDA      BUGBUG,DU
     002240  525243 2750 07                       ORA      BUGBUG,DL
                     002241          3655         BUGQ
                     525244               BUGBUG SET      BUGBUG+1
     002241  525244 2360 03                       LDQ      BUGBUG,DU
     002242  525244 2760 07                       ORQ      BUGBUG,DL
                     002243          3656         BUGXR    (0,X,Y,Z,Q)
                     525245               BUGBUG SET      BUGBUG+1
     002243  525245 2200 03                       LDX      0,BUGBUG,DU
     002244  525245 2220 03                       LDX      X,BUGBUG,DU
     002245  525245 2230 03                       LDX      Y,BUGBUG,DU
     002246  525245 2240 03                       LDX      Z,BUGBUG,DU
```

READ TASK -- FILL

```
002247  525245 2250 03                  LDX    0,BUGBUG,DU
                 002250    3657         BUG    (T$TEMP1,T)
                 525246         BUGBUG  SET    BUGBUG+1
002250  525246 2200 03                  LDX    0,BUGBUG,DU
002251  000027 7400 11                  STX    0,T$TEMP1,T
002252  000027 4400 11                  SXL    0,T$TEMP1,T
                 002253    3658         BUGL   (T$TRA,T)
                 525247         BUGBUG  SET    BUGBUG+1
002253  525247 2200 03                  LDX    0,BUGBUG,DU
002254  000004 4400 11                  SXL    0,T$TRA,T
002255  000000 7100 17     3659         TRA    0,L               RETURN TO CALLER
                           3660 *
                           3661 *
                           3662 *       SEND OUT HEADER
                           3663 *
                 002256    3664 FILL4   BSS    0
002256  200000 6750 07     3665         ERA    B$RHDR,DL         TURN OFF READ HEADER BIT
002257  000000 7550 16     3666         STA    J$FLAGS,J         AND SAVE IT
                           3667 *
                           3668 *       DETERMINE WHERE HEADER IS TO BE SENT
                           3669 *
002260  000400 3150 07     3670         CANA   B$LP,DL           IS IT A LINE PRINTER HEADER?
002261  002271 6000 00     3671         TZE    FILL5             NO, MUST BE CARD PUNCH
002262  003102 6220 00     3672         EAX    X,LPHDR           C(X) = FROM POINTER
002263  000013 2230 36     3673         LDX    Y,J$RPTR,J*       C(Y) = TO POINTER
 END OF BINARY CARD LPCP0060
002265  161700 5602 01     3674         RPD    LPHLN,1,TZE       MOVE IN HEADER DATA
002266  000000 2360 12     3675         LDQ    0,X
002267  000000 7560 13     3676         STQ    0,Y
002270  002236 7100 00     3677         TRA    FILL3             FUDGE TO LOOK LIKE A READ
                 002271    3678 FILL5   BSS    0                 OUTPUT HEADER FOR CARD PUNCH
002271  003172 6220 00     3679         EAX    X,CPHDR           C(X) = FROM POINTER
002272  000013 2230 36     3680         LDX    Y,J$RPTR,J*       C(Y) = TO POINTER
002273  067700 5602 01     3681         RPD    CPHLN,1,TZE       MOVE IN HEADER DATA
002274  000000 2350 12     3682         LDA    0,X
002275  000000 7550 13     3683         STA    0,Y
002276  002236 7100 00     3684         TRA    FILL3             FUDGE TO LOOK LIKE A READ
                 002277    3685 FILL6   BSS    0
002277  002204 6200 00     3686         EAX    0,FILL0           FUDGE RESTART ADDRESS
002300  000004 7400 11     3687         STX    0,T$TRA,T
002301  000004 6200 11     3688         EAX    0,Q$OFFST,T       GET OFFSET PTR TO TCB
002302  003262 7170 00     3689         XED    Q$XADD+Q$TASK     PLACE SELF ON Q$TASK
                 002303    3690         EXIT                     AND WAIT FOR AWHILE--TO DO SPTR.
002303  001547 7100 00                  TRA    $EXIT
```

READ TASK -- BANNERS

```
                     002304                 USE     CODE
                                            HEAD
                                    *
                                    *
                                    *                                BANNERS
                                    *
                                    *       THESE ARE THE OUTPUT BANNERS TO THE LINE PRINTERS AND
                                    *       CARD PUNCHES, RESPECTIVELY.
                                    *
                                    *
                                    *       THIS IS THE LINE PRINTER BANNER.  IS SAYS  FILE
                                    *
                     003102                 USE     CONST
                     003102           LPHDR BSS     0                    LINE PRINTER BANNER
   003102  772017171717                     OCT     772017171717
   003103  676767676767                     BCI     5,XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 END OF BINARY CARD LPCP0061
   003110  770217171717                     OCT     770217171717
   003111  676767676767                     BCI     5,XXXXXXX  XXX  X        XXXXXXX
   003116  770117171717                     OCT     770117171717
   003117  672020202020                     BCI     4,X          X  X        X
   003123  770117171717                     OCT     770117171717
   003124  672020202020                     BCI     4,X          X  X        X
   003130  770117171717                     OCT     770117171717
   003131  676767676720                     BCI     5,XXXXX      X  X        XXXXX
 END OF BINARY CARD LPCP0062
   003136  770117171717                     OCT     770117171717
   003137  672020202020                     BCI     4,X          X  X        X
   003143  770117171717                     OCT     770117171717
   003144  672020202020                     BCI     4,X          X  X        X
   003150  770117171717                     OCT     770117171717
   003151  672020202020                     BCI     4,X          X  X        X
   003155  770117171717                     OCT     770117171717
   003156  672020202020                     BCI     5,X        XXX  XXXXXXX  XXXXXXX
   003163  770217171717                     OCT     770217171717
 END OF BINARY CARD LPCP0063
   003164  676767676767                     BCI     5,XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   003171  772017171717                     OCT     772017171717
                     000070           LPHLN EQU     *-LPHDR              HEADER LENGTH IN WORDS
```

```
                                   *
                                   *
                                   *
                                   *      THIS IS THE CARD PUNCH BANNER.  IT SAYS  FILE
                                   *
                                   *
                          003172   CPHDR  BSS     0              CARD PUNCH BANNER
   003172  777740014001                   OCT     777740014001
                          003173          DUP     1,5
   003173  400140014001                   OCT     400140014001
   003200  477545014501                   OCT     477545014501
   003201  450145014501                   OCT     450145014501
   003202  440144014401                   OCT     440144014401
   003203  400140014001                   OCT     400140014001
   003204  400147754001                   OCT     400147754001
   003205  400140014001                   OCT     400140014001
   003206  477540054005                   OCT     477540054005
   003207  400540054005                   OCT     400540054005
   003210  400540054005                   OCT     400540054005
   003211  400140014001                   OCT     400140014001
 END OF BINARY CARD LPCP0064
   003212  477544454445                   OCT     477544454445
   003213  444544454445                   OCT     444544454445
   003214  440544054005                   OCT     440544054005
                          003215          DUP     1,7
   003215  400140014001                   OCT     400140014001
   003224  400177770000                   OCT     400177770000
                          000033   CPHLN  EQU     *-CPHDR        HEADER LENGTH IN WORDS
                          002304          USE     PREVIOUS
```

```
                    002304          USE     CODE
                                    HEAD
                               *
                               *
                               *                                 READ TASK
                               *
                               *    THIS SECTION OF CODE IS EXECUTED BY THE READ TASK TO READ
                               *     AN INPUT FILE INTO CORE. IT WILL READ IN BLOCKS OF THE
                               *    UNIT SIZED SPECIFIED WHEN OPENED BY THE MASTER.  IT WILL
                               *    FILL AS MANY BUFFERS AS TOLD AND WILL KEEP RE-FILLING THEM
                               *    UNTIL THE END OF FILE IS REACHED.  IT WILL THEN TERMINATE
                               *    ITSELF.
                               *
                               *    THE NUMBER OF BUFFERS TO FILL IS PRE-DEFINED BY AN ASSEMBLY
                               *    TIME CONSTANT TO BE 2.  THIS IMPLIES THAT ALL INPUT (AND
                               *    OBVIOUSLY ALL OUTPUT) IS DOUBLE-BUFFERED.  TO BECOME MULTI-
                               *    BUFFERED. SIMPLY CHANGE THE 2 TO N WHERE N IS THE NUMBER OF
                               *    BUFFERS TO WORK WITH.
                               *
                               *    ENTER WITH
                               *            C(XT) = TBLOCK-ADDRESS
                               *            C(XJ) = JOB-CONTROL-BLOCK-ADDRESS
                               *    CALLS
                               *            Q$P
                               *            FILL
                               *            Q$V
                               *            T$RELT
                               *    USES NO TEMPORARIES ITSELF
                               *
                    002304     RTASK   BSS     0
                    002304             P       (J$EMPTY,J)     DO A DOWN ON THE NUMBER OF EMPTY BUFFERS
002304  000011 6250 16                 EAX     Q,J$EMPTY,J
002305  000751 7070 00                 TSX     L,Q$P
                    002306             FILL                    FILL THIS BUFFER
002306  002171 7070 00                 TSX     L,$FILL
                    002307             V       (J$FULL,J)      DO AN UP ON NUMBER OF FULL BUFFERS
002307  000012 6250 16                 EAX     Q,J$FULL,J
002310  000772 7070 00                 TSX     L,Q$V
002311  000000 2350 16                 LDA     J$FLAGS,J       GET JOB STATUS FLAGS
002312  040000 3150 07                 CANA    B$ENDR,DL       TEST FOR DONE
002313  002304 6000 00                 TZE     RTASK           YES, BY HOOK OR CROOK
                               *
                               *    DONE, EVAPORATE PROCESSOR
                               *
                    002314     RTSK1   BSS     0
                    002314             RELT                    RELEASE TBLOCK
002314  001335 7000 00                 TSX     0,T$RELT
                    002315             EXIT                    POOF.
002315  001547 7100 00                 TRA     $EXIT
```

WRITE TASK -- WRITE

```
                       002316          USE     CODE
                                       HEAD
                                   *
                                   *
                                   *                                 WRITE
                                   *
                                   *
                                 WRITE MACRO   <NO-ARGUMENTS>
                                       TSX     L.$WRITE            CALL SUBROUTINE
                                       ENDM    WRITE
                                   *
                                   *
                                   *           WRITE -- SUBROUTINE
                                   *
                                   *
                                   *   THIS SUBROUTINE DOES THE ISSUING OF THE APPEND PRIMITIVE
                                   *   TO TRANSFER THE DATA TO THE OUTPUT DEVICE.  IT DOES ALL OF
                                   *   ITS OWN ERROR RECOVERY.
                                   *
                                   *
                                   *   ENTER WITH
                                   *           C(XJ) = JBC-ADDRESS
                                   *           C(XT) = TBLOCK-ADDRESS
                                   *           C(T$TEMP1,T) = STARTING ADDRESS
                                   *           C(T$TEMP2,T) = NUMBER TO SEND
                                   *           C(T$TEMP3,T) = MODE
                                   *   ENTER BY
                                   *           TSX L.WRITE
                                   *   CALLS
                                   *           $APEND
                                   *           C$MESSX  (CONDITIONALLY)
                                   *   RETURN TO
                                   *           TRA 0,L
                                   *   RETURN WITH
                                   *           C(XJ) = JBLOCK-ADDRESS
                                   *           C(XT) = TBLOCK-ADDRSS
                                   *           C(XL) = RETURN-ADDRESS
                                   *   USES
                                   *           NO LOCALS
                                   *           TEMP1, TEMP2, TEMP3 ARE PASSED TO IT
                                   *           TEMP4, TEMP5  (CONDITIONALLY)
                                   *
                                   *
                                   *
                       002316    WRITE BSS     0
END OF BINARY CARD LPCP0065
 002316  000022 7470 11                STX     L.T$TEMP6,T         SAVE RETURN ADDRESS
 002317  000017 4500 16                STZ     J$RTRY,J            RESET ERROR COUNTER
 002320  000000 2350 16                LDA     J$FLAGS,J           GET FLAG BITS
 002321  000200 3150 07                CANA    B$CP,DL             IS IT A CARD PUNCH JOB?
```

WRITE TASK -- WRITE

```
002322  002327 6000 00       3850         TZE      WT1              NO. GO WRITE IT OUT
002323  000004 2340 16       3851         SZN      J$CARD1.J        YES. WELL IS IT THE FIRST?
002324  002327 6010 00       3852         TNZ      WT1              NO. GO WRITE IT OUT
002325  000004 5540 16       3853         STC1     J$CARD1.J        YES. MARK IT AS SCUH
002326  002357 7100 00       3854         TRA      WT2.1            TELL OPER TO REMOVE CARDS AND READY PUNCH
               002327        3855 WT1     APEND    (J$OFRN.J).(T$TEMP1.T).(T$TEMP2.T).(T$TEMP3.T)
002327  000551 7000 00                    TSX      0.$APEND
002330  000003 0000 16                    ARG      J$OFRN.J
002331  000027 0000 11                    ARG      T$TEMP1.T
002332  000026 0000 11                    ARG      T$TEMP2.T
002333  000025 0000 11                    ARG      T$TEMP3.T
               002334        3856         CHECK    WT4.B$BZ.WT1.B$HDWE.WT2
002334  000000 7200 11                    LXL      0.T$SRW1.T
002335  000077 3600 03                    ANX      0.B$STMK.DU
002336  002430 6000 00                    TZE      WT4
002337  000003 1000 03                    CMPX     0.B$BZ.DU
002340  002327 6000 00                    TZE      WT1
002341  000012 1000 03                    CMPX     0.B$HDWE.DU
002342  002344 6000 00                    TZE      WT2
002343  777777 7100 00                    TRA      $ERROR
               002344        3857 WT2     BSS      0                HARDWAPE ERROR
                             3858 *
                             3859 *       HARDWARE ERROR
                             3860 *
END OF BINARY CARD LPCP0066
002344  000001 7220 11       3861         LXL      X.T$SRW2.T       GET PHYSICAL STATUS
002345  000000 2350 16       3862         LDA      J$FLAGS.J        IT IS A PRINTER
002346  000400 3150 07       3863         CANA     B$LP.DL          OR PUNCH ERROR
002347  002357 6000 00       3864         TZE      WT2.1            PUNCH ERROR -- ALWAYS DO ACCEPT ATTENTION
002350  000020 3020 03       3865         CANX2    =020.DU          ***GDP TEST***
002351  002357 6010 00       3866         TNZ      WT2.1            YES. DO AN ACCEPT ATTENTION
002352  000017 0540 16       3867         AOS      J$RTRY.J         BUMP RETRY COUNT
002353  000017 2350 16       3868         LDA      J$RTRY.J         GET CURRENT COUNT
002354  000005 1150 07       3869         CMPA     $RTMAX.DL        COMPARE TO MAX
002355  002327 6020 00       3870         TNC      WT1              OK -- RETRY AGAIN
002356  002430 7100 00       3871         TRA      WT4              NOPE -- SKIP ENTIRE BLOCK
               002357        3872 WT2.1   BSS      0                DO ACCEPT ATTENTION
002357  000006 2220 11       3873         LDX      X.T$NCB.T        GET PTR TO NCB
002360  000031 2350 12       3874         LDA      C$STATE.X        GET STATE FOR CAUSE
002361  000024 7550 11       3875         STA      T$TEMP4.T        SAVE FOR STATE
002362  000016 7220 16       3876         LXL      X.J$RES.J        GET DEVICE PTR
002363  000000 2350 12       3877         LDA      R$NAME.X         GET NAME
002364  000007 3750 07       3878         ANA      7.DL             ISOLATE UNIT NUMBER
002365  000022 7350 00       3879         ALS      18               MOVE TO MESSAGE FIELD
002366  016000 2750 03       3880         ORA      B$RDY.DU         MARK IT TO BE READIED
002367  000023 7550 11       3881         STA      T$TEMP5.T        SAVE FOR MESSAGE
               002370        3882         BRANCH   NOPASS.C$MESSX.(T$TEMP4.T).(T$TEMP5.T)
002370  001325 7000 00                    TSX      0.T$GETT
002371  000000 6220 11                    EAX      X.0.T
END OF BINARY CARD LPCP0067
```

```
002372 000005 2210 12                LDX    T,T$LINK,X
002373 000024 2360 11                LDQ    T$TEMP4,T
002374 000027 7560 12                STQ    T$TEMP1,X
002375 000023 2360 11                LDQ    T$TEMP5,T
002376 000026 7560 12                STQ    T$TEMP2,X
002377 000000 6210 12                EAX    T,0,X
002400 001613 6200 00                EAX    0,C$MESSX
002401 000004 7400 11                STX    0,T$TRA,T
002402 000004 6200 11                EAX    0,Q$OFFST,T
002403 003262 7170 00                XED    Q$XADD+Q$TASK
002404 000005 2210 12                LDX    T,T$LINK,X
                   002405            BUGXR  (0,X)
                   525250     BUGBUG SET    BUGBUG+1
002405 525250 2200 03                LDX    0,BUGBUG,DU
002406 525250 2220 03                LDX    X,BUGBUG,DU
                   002407            BUG    (T$TEMP4,T)
                   525251     BUGBUG SET    BUGBUG+1
002407 525251 2200 03                LDX    0,BUGBUG,DU
002410 000024 7400 11                STX    0,T$TEMP4,T
002411 000024 4400 11                SXL    0,T$TEMP4,T
                   002412            BUG    (T$TEMP5,T)
                   525252     BUGBUG SET    BUGBUG+1
002412 525252 2200 03                LDX    0,BUGBUG,DU
002413 000023 7400 11                STX    0,T$TEMP5,T
002414 000023 4400 11                SXL    0,T$TEMP5,T
                              *
                              *      NOW PUT OUT AN ACCEPT ATTENTION ON DEVICE
                              *
                   002415     WT3    APEND  (J$OFRN,J),(0,DU),(0,DU),(1,DU)
002415 000551 7000 00                TSX    0,$APEND
002416 000003 0000 16                ARG    J$OFRN,J
002417 000000 0000 03                ARG    0,DU
END OF BINARY CARD LPCP0068
002420 000000 0000 03                ARG    0,DU
002421 000001 0000 03                ARG    1,DU
                   002422            CHECK  WT1,B$BZ,WT3
002422 000000 7200 11                LXL    0,T$SRW1,T
002423 000077 3600 03                ANX    0,B$STMK,DU
002424 002327 6000 00                TZE    WT1
002425 000003 1000 03                CMPX   0,B$BZ,DU
002426 002415 6000 00                TZE    WT3
002427 777777 7100 00                TRA    $ERROR
                              *
                              *      SUCCESSFUL TRANSFER, CLEAN UP
                              *
                   002430     WT4    BSS    0
002430 000022 2270 11                LDX    L,T$TEMP6,T     RETRIEVE RETURN ADDRESS
                   002431            BUG    (T$TEMP1,T)     BUG EVERYTHING
                   525253     BUGBUG SET    BUGBUG+1
002431 525253 2200 03                LDX    0,BUGBUG,DU
```

```
002432  000027 7400 11                        STX     0,T$TEMP1,T
002433  000027 4400 11                        SXL     0,T$TEMP1,T
                 002434      3896             BUG     (T$TEMP2,T)
                 525254             BUGBUG    SET     BUGBUG+1
002434  525254 2200 03                        LDX     0,BUGBUG,DU
002435  000026 7400 11                        STX     0,T$TEMP2,T
002436  000026 4400 11                        SXL     0,T$TEMP2,T
                 002437      3897             BUG     (T$TEMP3,T)
                 525255             BUGBUG    SET     BUGBUG+1
002437  525255 2200 03                        LDX     0,BUGBUG,DU
002440  000025 7400 11                        STX     0,T$TEMP3,T
002441  000025 4400 11                        SXL     0,T$TEMP3,T
                 002442      3898             BUGXR   (X,Y,Z,Q)
                 525256             BUGBUG    SET     BUGBUG+1
002442  525256 2220 03                        LDX     X,BUGBUG,DU
002443  525256 2230 03                        LDX     Y,BUGBUG,DU
002444  525256 2240 03                        LDX     Z,BUGBUG,DU
002445  525256 2250 03                        LDX     Q,BUGBUG,DU
                 002446      3899             BUGU    (T$TEMP6,T)
                 525257             BUGBUG    SET     BUGBUG+1
 END OF BINARY CARD LPCP0069
002446  525257 2200 03                        LDX     0,BUGBUG,DU
002447  000022 7400 11                        STX     0,T$TEMP6,T
                 002450      3900             BUGA
                 525260             BUGBUG    SET     BUGBUG+1
002450  525260 2350 03                        LDA     BUGBUG,DU
002451  525260 2750 07                        ORA     BUGBUG,DL
                 002452      3901             BUGQ
                 525261             BUGBUG    SET     BUGBUG+1
002452  525261 2360 03                        LDQ     BUGBUG,DU
002453  525261 2760 07                        ORQ     BUGBUG,DL
002454  000000 7100 17       3902             TRA     0,L                 RETURN TO CALLER
```

```
002455          USE     CODE
                HEAD
         *
         *
         *                                      320 OUTPUT MODE
         *                                      GECOS FORMAT
         *
         *      THIS SUBROUTINE OUTPUTS THE BUFFER IN 320 LINE EDITED
         *      FORMAT.  IT ASSUMES STANDARD GERC SYSTEM FORMAT:
         *
         *      BLOCK SIZE -- DATA BLOCKS ARE VARIABLE IN LENGTH UP TO A
         *              UP TO A MAXIMUM BLOCK SIZE OF 320 WORDS (DECIMAL).
         *      BLOCK SERIAL NUMBER--  A BLOCK SERIAL NUMBER WILL EXIST AS
         *              THE FIRST WORD OF EACH DATA BLOCK AND WILL CONTAIN TWO
         *              BINARY VALUES AS FOLLOWS:
         *              BITS 0 - 17    BLOCK SERIAL NUMBER -- THE SPEQUENTIAL NUMBER
         *              BITS 18 -35    BLOCK SIZE -- SIZE OF BLOCK IN WORDS, NOT INCLUDING

         *                              ITSELF
         *      RECORD FORMAT -- RECORDS WITHIN A BLOCK ARE VARIABLE LENGTH
         *              BITS 0 - 17    RECORD SIZE IN WORDS, NOT INCLUDING ITSELF
         *              BITS 18- 23    NOT USED UNLES 0 - 17 ARE ZERO, THEN FILE MARK CHAR
         *              BITS 24 -27    ZEROS
         *              BITS 28-29     LOGICAL RECORD CODE
         *                             0 = NOT A MEDIA CONVERSION RECORD
         *                             1 = BINARY CARD IMAGE
         *                             2 = HOLLERITH CARD IMAGE
         *                             3 = PRINT-LINE IMAGE
         *              BITS 30- 35    REPORT CODE
         *
         *
         *      ENTER WITH
         *              C(XJ) = JCB-ADDRESS
         *              C(XT) = TBLOCK-ADDRESS
         *      ENTER BY
         *              TRA TABLE,AL* (TYPE)
         *              WRITE
         *      RETURN WITH
         *              C(XT) = TBLOCK-ADDRESS
         *              C(XJ) = JCB-ADDRESS
         *      RETURNS TO EMPT1
         *      USES
         *              NO LOCALS
         *              TEMP1 IS PASSED
         *              TEMP2, TEMP3, TEMP6
```

WRITE TASK -- 320 OUTPUT MODE

```
                                3950 *
                                3951 *
                                3952 *
                      002455    3953 .320    BSS     0               320 WORD BLOCK PROCESSOR
                      002455    3954 GECOS   BSS     0               PRINT-LINE/ GECOS FORMAT
    002455  000014 2220 36      3955         LDX     X,J$WPTR,J*     GET OUTPUT BUFFER ADDRESS
    002456  000010 7420 16      3956         STX     X,J$XDATA,J     SAVE AS END OF BUFFER
    002457  000000 7240 12      3957         LXL     Z,0,X           GET BLOCK SIZE
    002460  000500 1040 03      3958         CMPX    Z,320,DU        IS IT VALID?
    002461  002520 6030 00      3959         TRC     GCOS4           NO, TOO BIG
    002462  000001 0240 03      3960         ADLX    Z,1,DU          INCLUDING ITSELF
    002463  000010 0440 16      3961         ASX     Z,J$XDATA,J     UPDATE END OF BUFFER POINTER
    002464  002470 7100 00      3962         TRA     GCOS5           ENTER LOOP
                                3963 *
                                3964 *       OUTPUT NEXT LINE
                                3965 *
                      002465    3966 GCOS1   BSS     0
    002465  000007 2240 16      3967         LDX     Z,J$RCW,J       GET ADDRESS OF OLD RCW
    002466  000000 2220 14      3968         LDX     X,0,Z           GET LENGTH OF RCW
    002467  000007 0220 16      3969         ADLX    X,J$RCW,J       TO COMPUTE ADDRESS OF NEXT RCW
                      002470    3970 GCOS5   BSS     0
    002470  000001 0220 03      3971         ADLX    X,1,DU          INCLUDE RCW ITSELF
    002471  000007 7420 16      3972         STX     X,J$RCW,J       AND SAVE IT
    002472  000001 0220 03      3973         ADLX    X,1,DU          ONE AGAIN TO POINT TO STARTING ADDRESS OF DATA
  END OF BINARY CARD LPCP0070
    002473  000027 7420 11      3974         STX     X,T$TEMP1,T     SAVE FOR WRITE SUBROUTINE
    002474  000007 2220 36      3975         LDX     X,J$RCW,J*      GET NUMBER OF WORDS IN THIS RCW
    002475  002520 6000 00      3976         TZE     GCOS4           ZERO, DONE WITH THIS BLOCK
    002476  000027 0220 11      3977         ADLX    X,T$TEMP1,T     TEST FOR VALID LENGTH
    002477  000010 1020 16      3978         CMPX    X,J$XDATA,J
    002500  002502 6000 00      3979         TZE     *+2
    002501  002520 6030 00      3980         TRC     GCOS4           NG, DONE
    002502  000007 2360 36      3981         LDQ     J$RCW,J*        GET BACK LENGTH
    002503  000022 7720 00      3982         QRL     18              IN QL
    002504  000007 2350 36      3983         LDA     J$RCW,J*        GET LOGICAL RECORD MODE
    002505  001700 3750 07      3984         ANA     B$MODMK,DL      MASK TO MEDIA ONLY
    002506  000006 7710 00      3985         ARL     6               AND RIGHT JUSTIFIED
    002507  000000 7220 16      3986         LXL     X,J$FLAGS,J     GET JOB STATUS BITS
    002510  000400 3020 03      3987         CANX    X,B$LP,DU       IS IT A PRINTER JOB?
    002511  003232 6000 25      3988         TZE     CPTAB,AL*       NO, BRANCH ON CP MODE
    002512  003225 7100 25      3989         TRA     LPTAB,AL*       BRANCH ON LP MODE
```

WRITE TASK -- 320 OUTPUT MODE

```
                              3991 *
                              3992 *
                              3993 *
                   002513     3994 GCOS7  BSS    0                 JOINED HERE FROM ABOVE TRANSFERS
  002513  000000 6360 06      3995        EAQ    0,QL              MOVE CHARACTER COUNT TO QU
  002514  000026 7560 11      3996        STQ    T$TEMP2,T         SAVE FOR WRITE
  002515  000025 7420 11      3997        STX    X,T$TEMP3,T       SAVE MODE
                   002516     3998 GCOS2  WRITE                    NOW WRITE IT OUT
  002516  002316 7070 00                  TSX    L,$WRITE
                   002517     3999 GCOS3  BSS    0
  002517  002465 7100 00      4000        TRA    GCOS1             LOOP
                              4001 *
                              4002 *      INFORMATION WRITTEN
                              4003 *
  002520  000022 7270 11      4004 GCOS4  LXL    L,T$TEMP6,T       RETRIEVE RETURN
                   002521     4005        BUGXR  (0,X,Y,Z,Q)
                   525262          BUGBUG SET    BUGBUG+1
END OF BINARY CARD LPCP0071
  002521  525262 2200 03                  LDX    0,BUGBUG,DU
  002522  525262 2220 03                  LDX    X,BUGBUG,DU
  002523  525262 2230 03                  LDX    Y,BUGBUG,DU
  002524  525262 2240 03                  LDX    Z,BUGBUG,DU
  002525  525262 2250 03                  LDX    Q,BUGBUG,DU
                   002526     4006        BUGA
                   525263          BUGBUG SET    BUGBUG+1
  002526  525263 2350 03                  LDA    BUGBUG,DU
  002527  525263 2750 07                  ORA    BUGBUG,DL
                   002530     4007        BUGQ
                   525264          BUGBUG SET    BUGBUG+1
  002530  525264 2360 03                  LDQ    BUGBUG,DU
  002531  525264 2760 07                  ORQ    BUGBUG,DL
  002532  002622 7100 00      4008        TRA    EMPT1             RETURN FROM WHENCE CALLED
```

WRITE TASK -- 320 FORMAT CONVERSIONS

```
                    002533   4010        USE     CODE
                             4011        HEAD
                             4012 *
                             4013 *
                             4014 *                           GECOS FORMAT CONVERSION
                             4015 *
                             4016 *      ONE BRANCHES THOUGH THIS TABLE TO PICK UP THE CORRECT
                             4017 *      CHARACTER COUNT AND MODE FOR PRINTING A 320 BLOCK.
                             4018 *
                    003225   4019        USE     CONST
                    003225   4020 LPTAB  BSS     0
  003225  002534 0000 00     4021        ARG     LP.0         NOT A MEDIA CONVERSION RECORD
  003226  002564 0000 00     4022        ARG     IGNOR        BINARY CAR IMAGE
  003227  002534 0000 00     4023        ARG     LP.2         HOLLERITH CARD IMAGE
  003230  002534 0000 00     4024        ARG     LP.3         PRINT LINE
  003231  002564 0000 20     4025        ARG     IGNOR,*
                    002533   4026        USE     PREVIOUS
  002533  002564 0000 00     4027        ARG     IGNOR        NOT IMPLEMENTED
                             4028 *
                             4029 *
                             4030 *
                             4031 *      ONE BRANCHES THOUGH THIS TABLE TO PICK UP THE CORRECT
                             4032 *      CHARACTER COUNT AND MODE FOR PUNCHING A 320 BLOCK.
                             4033 *
                    003232   4034        USE     CONST
                    003232   4035 CPTAB  BSS     0
  003232  002542 0000 00     4036        ARG     CP.0         NOT A MEDIA CONVERSION RECORD
  003233  002550 0000 00     4037        ARG     CP.1         BINARY CARD IMAGE
  003234  002556 0000 00     4038        ARG     CP.2         HOLLERITH CARD IMAGE
END OF BINARY CARD LPCP0072
  003235  002556 0000 00     4039        ARG     CP.3         PRINT LINE
                    003236   4040        DUP     1,16-4
  003236  002564 0000 20     4041        ARG     IGNOR,*
                    002534   4042        USE     PREVIOUS
```

WRITE TASK -- 320 OUTPUT PARAMETER ROUTINES

```
                       002534             USE      CODE
                                          HEAD
                                   *
                                   *
                                   *                         PARAMTER ROUTINES
                                   *
                                   *      THESE ROUTINES BASED ON THE LOGICAL RECORD MODE PICK
                                   *      RETURN THE NUMBER OF CHARACTERS TO SEND AMD THE MODE.
                                   *
                                   *      ENTER WITH
                                   *               C(QL) = NUMBER OF WORDS
                                   *      RETURN TO
                                   *               GCOS7 EXCEPT IGNOR RETURNS TO GCOS4
                                   *      RETURN WITH
                                   *               C(QL) = NUMBER OF CHARACTERS TO PRINT
                                   *               C(XX) = MODE
                                   *
                       002534      LP.0   BSS      0                   NOT A MEDIA CONVERSION RECORD
                       002534      LP.2   BSS      0                   HOLLERITH CARD IMMAGE
                       002534      LP.3   BSS      0                   PRINT LINE IMAGE
   002534  000006 4020 07                 MPY      6,DL                CONVERT TO CHARACTER COUNT
   002535  010000 1160 07                 CMPQ     4096,DL             TEST AGAINST MAX
   002536  002540 6020 00                 TNC      *+2                 OK
   002537  010000 2360 07                 LDQ      4096,DL             GET MAX INSTEAD
   002540  000003 2220 03                 LDX      X,B$LP3,DU          GET MODE
   002541  002513 7100 00                 TRA      GCOS7               RETURN
                                   *
                                   *
                                   *
                       002542      CP.0   BSS      0                   NOT A MEDIA CONVERSION RECORD
   002542  002564 7100 00                 TRA      IGNOR               ***FOR NOW
   002543  000016 1160 07                 CMPQ     14,DL               TEST FOR BCD
 END OF BINARY CARD LPCP0073
   002544  002556 6000 00                 TZE      CP.2                YES
   002545  000033 1160 07                 CMPQ     27,DL               TEST FOR BINARY
   002546  002550 6000 00                 TZE      CP.1                YES
   002547  002564 7100 20                 TRA      IGNOR,*
                                   *
                       002550      CP.1   BSS      0                   BINARY CARD IMAGE
   002550  000006 4020 07                 MPY      6,DL                CONVERT TO CHARACTER COUNT
   002551  000240 1160 07                 CMPQ     160,DL              TEST AGAINST MAX
   002552  002554 6020 00                 TNC      *+2                 OK
   002553  000240 2360 07                 LDQ      160,DL              GET MAX INSTEAD
   002554  000002 2220 03                 LDX      X,B$CP2,DU          GET BINARY MODE
   002555  002513 7100 00                 TRA      GCOS7               RETURN
                                   *
                       002556      CP.2   BSS      0                   HOLLERITH IMAGE
                       002556      CP.3   BSS      0
   002556  000006 4020 07                 MPY      6,DL                CONVERT TO CHARACTER COUNT
   002557  000120 1160 07                 CMPQ     80,DL               TEST AGAINST MAX
```

```
002560  002562 6020 00     4093          TNC     *+2              OK
002561  000120 2360 07     4094          LDQ     80,DL            GET MAX INSTEAD
002562  000003 2220 03     4095          LDX     X,B$CP3,DU       GET HOLLERITH MODE
002563  002513 7100 00     4096          TRA     GCOS7            RETURN
                           4097 *
                           4098 *
                   002564  4099 IGNOR    BSS     0                IGNORE ENTIRE BLOCK
002564  002520 7100 00     4100          TRA     GCOS4            RETURN AS IF DONE
```

WRITE TASK -- 512 OUTPUT MODE

```
                  002565    4102        USE     CODE
                            4103        HEAD
                            4104 *
                            4105 *
                            4106 *                                 LP512 OUTPUT MODE
                            4107 *
                            4108 *      THIS SUBROUTINE OUTPUTS THE BUFFER IN 512. WORDS EDITED
                            4109 *      CONTINOUS MODE.  IT SETS UP ALL THE PARAMETERS
                            4110 *
                            4111 *      ENTER WITH
                            4112 *              C(XJ) = JCB-ADDRESS
                            4113 *              C(XT) = TCB-ADDRESS
                            4114 *
                            4115 *      ENTER BY
                            4116 *              TRA TABLE.AL (TYPE) *
                            4117 *      RETURNS TO EMPT1
                            4118 *
                  002565    4119 .512    BSS     0               512 WORD BLOCK PROCESSOR
                  002565    4120 LP512   BSS     0
002565  000000 2350 16      4121         LDA     J$FLAGS.J       GET JOB STATUS BITS
002566  000400 3150 07      4122         CANA    B$LP.DL         SHOULD BE A PRINTER JOB
002567  777777 6000 00      4123         TZE     $ERROR          IT ISN'T!
002570  006000 2220 03      4124         LDX     X.512*6.DU      SET UP CHARACTER COUNT
                  002571    4125 LP1     BSS     0
END OF BINARY CARD LPCP0074
002571  000026 7420 11      4126         STX     X.T$TEMP2.T     SAVE FOR APPEND
002572  000002 2230 03      4127         LDX     Y.B$LP2.DU      USE 512 EDITED CONTINOUS. NO SLEW
002573  000025 7430 11      4128         STX     Y.T$TEMP3.T     PASS ON TO WRITE ROUTINE
                  002574    4129         WRITE                   DO THE WRTIE
002574  002316 7070 00                   TSX     L.$WRITE
002575  002622 7100 00      4130         TRA     EMPT1           RETURN TO EMPTY ROUTINE
```

WRITE TASK -- EMPTY

```
                     002576         USE     CODE
                                    HEAD
                             *
                             *
                             *                               EMPTY
                             *
                             *      EMPTY FORCES THE OUTPUT OF A CORE BUFFER
                             *
                             EMPTY  MACRO   <NO-ARGUMENTS>
                                    TSX     L.$EMPTY         CALL SUBROUTINE
                                    ENDM    EMPTY
                             *
                             *
                             *              EMPTY -- SUBROUTINE
                             *
                             *      THIS SUBROUTINE DOES THE ACTUAL WRITING OF THE OUTPUT.  IT
                             *      IS CHARGED WITH THE WRITING THE CORRECT NUMBER OF UNITS
                             *      IN THE PROPER FORMAT TO THE OUTPUT FILE (DEVICE).  IT THEN
                             *      DE-ALLOCATES THE BUFFER.  IT HANDLES ALL ERRORS AND RETURNS
                             *      A JOB STATUS TO THE CALLER.
                             *
                             *      ENTER WITH
                             *              C(XJ) = JCB-ADDRESS
                             *              C(XT) = TBLOCK-ADDRESS
                             *      ENTER BY
                             *              TSX L.EMPTY
                             *      CALLS
                             *              .512
                             *              .320
                             *              WRITE
                             *      RETURN WITH
                             *              C(XJ) = JCB-ADDRESS
                             *              C(XT) = TBLOCK-ADDRESS
                             *      EXIT TO 0.L
                             *      USES
                             *              NO LOCALS
                             *              TEMP1, TEMP2, TEMP3
                             *
                             *
                     002576  EMPTY  BSS     0
002576  000004 4470 11              SXL     L.TSTRA.T        SAVE RETURN ADDRESS
002577  000014 2220 16              LDX     X.JSWPTR.J       GET OLD BUFFER POINTER
002600  000001 0220 03              ADLX    X.1.DU           BUMP TO NEW
002601  000020 1020 16              CMPX    X.JSRESET.J      TEST FOR WRAP
002602  002604 6020 00              TNC     *+2              OK
002603  000020 7220 16              LXL     X.JSRESET.J      WRAP
002604  000014 7420 16              STX     X.JSWPTR.J       SAVE NEW POINTER
002605  000000 2350 12              LDA     0.X              GET ADDRESS OF NEXT BUFFER
002606  002656 6000 00              TZE     EMPTYX           DONE
002607  000027 7550 11              STA     TSTEMP1.T        SAVE FOR EASY ACCESS
```

WRITE TASK -- EMPTY

```
     002610  000000 2350 16      4182          LDA     J$FLAGS,J        GET FLAGS
     002611  400000 3150 07      4183          CANA    B$KILL,DL        TEST FOR ABORT
     002612  002622 6010 00      4184          TNZ     EMPT1            YES, JUST RELASE BUFFERS TILL EOF HIT
     002613  100000 3150 07      4185          CANA    B$WHDR,DL        DO WE NEED A HEADER?
     002614  002640 6010 00      4186          TNZ     EMPT3            YES
     002615  010000 3750 03      4187          ANA     B$OUTMK,DU       ISOLATE OUTPUT FORMAT
 END OF BINARY CARD LPCP0075
     002616  000036 7710 00      4188          ARL     36-4-1-1         RIGHT JUSTIFIED
     002617  002620 7100 25      4189          TRA     *+1,AL*          TRANSFER TO OUTPUT FORMAT ROUTINE
     002620  002565 0000 00      4190          ARG     .512             512 WORD  BLOCK
     002621  002455 0000 00      4191          ARG     .320             320 WORD BLOCK
                                 4192 *
                                 4193 *        RETURN HERE AFTER OUTPUT IS COMPLETE
                                 4194 *
                    002622       4195 EMPT1    BSS     0                JOINED HERE BY ALL OUTPUTER
     002622  000006 0540 16      4196          AOS     J$RSTRT,J        BUMP RESTART ADDRESS IN BLOCK
                    002623       4197          RELC    (J$WPTR,J*)      RELEASE BUFFER, FORCE HOLES TO TOP OF MEM
     002623  000014 2350 36                    LDA     J$WPTR,J*
     002624  001110 7070 00                    TSX     L,R$RELC
                    002625       4198 EMPT2    BSS     0                GET READY TO RETURN TO CALLER
     002625  000004 7270 11      4199          LXL     L,T$TRA,T        RETRIEVE RETURN ADDRESS
                    002626       4200          BUGA                     BUG ALL REGISTERS EXCEPT T AND J
                    525265            BUGBUG SET     BUGBUG+1
     002626  525265 2350 03                    LDA     BUGBUG,DU
     002627  525265 2750 07                    ORA     BUGBUG,DL
                    002630       4201          BUGQ
                    525266            BUGBUG SET     BUGBUG+1
     002630  525266 2360 03                    LDQ     BUGBUG,DU
     002631  525266 2760 07                    ORQ     BUGBUG,DL
                    002632       4202          BUGXR   (0,X,Y,Z,Q)
                    525267            BUGBUG SET     BUGBUG+1
     002632  525267 2200 03                    LDX     0,BUGBUG,DU
     002633  525267 2220 03                    LDX     X,BUGBUG,DU
     002634  525267 2230 03                    LDX     Y,BUGBUG,DU
     002635  525267 2240 03                    LDX     Z,BUGBUG,DU
     002636  525267 2250 03                    LDX     Q,BUGBUG,DU
     002637  000000 7100 17      4203          TRA     0,L              RETURN TO CALLER
                                 4204 *
                                 4205 *        OUTPUT HEADER
                                 4206 *
                    002640       4207 EMPT3    BSS     0                LP BANNER
                    002640       4208          DECRM   (J$RSTRT,J)      BACK UP START ADDRESS FOR HEADER
     002640  000001 3360 07                    LCQ     1,DL
     002641  000006 0560 16                    ASQ     J$RSTRT,J
     002642  100000 6750 07      4209          ERA     B$WHDR,DL        TURN OFF WRITE HEADER BIT
     002643  000000 7550 16      4210          STA     J$FLAGS,J        SAVE IT
 END OF BINARY CARD LPCP0076
     002644  000400 3150 07      4211          CANA    B$LP,DL          SEE HERE HEADER IS TO GO
     002645  002650 6000 00      4212          TZE     EMPT4            CARD PUNCH
     002646  000520 2220 03      4213          LDX     X,LPHLN*6,DU     LINE PRINT; GET NUBER OF CHARACTERS
```

WRITE TASK -- EMPTY

```
002647  002571 7100 00        4214         TRA      LP1                USE LINE PRINTER ROUTINE
                    002650    4215 EMPT4   BSS      0                  OUTPUT HEADER ON CARD PUNCH
002650  000240 2220 03        4216         LDX      X,CPHLN*6-2,DU     GET NUMBER OF CHARACTERS
002651  000026 7420 11        4217         STX      X,T$TEMP2,T        SAVE
002652  000002 2230 03        4218         LDX      Y,2,DU             MAKE MODE BINARY
002653  000025 7430 11        4219         STX      Y,T$TEMP3,T        SAVE IT
                    002654    4220         WRITE                       WRITE IT OUT
002654  002316 7070 00                     TSX      L,$WRITE
002655  002622 7100 00        4221         TRA      EMPT1              DONE
                              4222 *
                              4223 *       EOF REACHED
                              4224 *
                    002656    4225 EMPTX   BSS      0
002656  020000 2350 07        4226         LDA      B$ENDW,DL          GIVE EOF STATUS
002657  000000 2550 16        4227         ORSA     J$FLAGS,J          GIVE IT TO CALLER
002660  002625 7100 00        4228         TRA      EMPT2              ACT NORMAL
```

WRITE TASK -- MAIN

```
                    002661      4230        USE     CODE
                                4231        HEAD
                                4232 *
                                4233 *
                                4234 *                                WRITE TASK
                                4235 *
                                4236 *      THIS SECTION OF CODE IS EXECUTED BY THE WRITE TASK TO
                                4237 *      WRITE AN OUTPUT BUFFER TO PRE-OPENED FILE (DEVICE)
                                4238 *      IN A SPECIFIED FORMAT.  IT WILL WRITE OUT AS MANY BUFFERS
                                4239 *      AS TOLD. AND WILL KEEP OUTPUTTING UNITL AN END OF FILE
                                4240 *      STATUS IS REACHED.  IT WILL THEM SEND A MESSAGE TO THE
                                4241 *      MONITOR NOTIFYING HIM OF A SUCCESSFUL TERMINATION AND
                                4242 *      THEN TERMINATE ITSELF.
                                4243 *
                                4244 *      ENTER WITH
                                4245 *              C(XT) = TBLOCK-ADDRESS
                                4246 *              C(XJ) = JOB-CONTROL-BLOCK-ADDRESS
                                4247 *     CALLS
                                4248 *              Q$P
                                4249 *              EMPTY
                                4250 *              Q$V
                                4251 *              $CLOSE
                                4252 *              C$MESSC
                                4253 *              R$RELP
                                4254 *              J$RELJ
                                4255 *              T$RELT
                                4256 *
                                4257 *
                    002661      4258 WTASK  BSS     0
                    002661      4259        P       (J$FULL.J)        DO A DOWN ON NUMBER OF FULL BUFFERS
  002661  000012 6250 16                    EAX     Q.J$FULL.J
  002662  000751 7070 00                    TSX     L.Q$P
                    002663      4260        EMPTY                     EMPTY THIS BUFFER
  002663  002576 7070 00                    TSX     L.$EMPTY
                    002664      4261        V       (J$EMPTY.J)       DO AN UP ON THE NUMBER OF EMPTY BUFFERS
  002664  000011 6250 16                    EAX     Q.J$EMPTY.J
  002665  000772 7070 00                    TSX     L.Q$V
  002666  000000 2350 16        4262        LDA     J$FLAGS.J         GET JOB STATUS FLAGS
  002667  020000 3150 07        4263        CANA    B$ENDW.DL         TEST FOR DONE
  002670  002661 6000 00        4264        TZE     WTASK             NO. LOOP
                    002671      4265 WTSK1  BSS     0
                                4266 *
                                4267 *      DONE. CLOSE INPUT FILE
                                4268 *
                    002671      4269        CLOSE   (J$IFRN.J)        CLOSE INPUT FILE. WHICH ALSO UNLOCKS IT
 END OF BINARY CARD LPCP0077
  002671  000616 7000 00                    TSX     0.$CLOSE
  002672  000002 0000 16                    ARG     J$IFRN.J
                    002673      4270        CHECK   WTSK2.3$BZ.WTSK1
  002673  000000 7200 11                    LXL     0.T$SRW1.T
```

WRITE TASK -- MAIN

```
002674  000077 3600 03                          ANX     0,B$STMK,DU
002675  002701 6000 00                          TZE     WTSK2
002676  000003 1000 03                          CMPX    0,B$RZ,DU
002677  002671 6000 00                          TZE     WTSK1
002700  777777 7100 00                          TRA     $ERROR
                 002701      4271 WTSK2         BSS     0
                             4272 *
                             4273 *             INFORM MONITOR
                             4274 *
002701  014000 2350 03       4275               LDA     B$DONE,DU         ASSUME SUCCESSFUL
002702  000000 2360 16       4276               LDQ     J$FLAGS,J         GET JOB NUMBER
002703  400000 3160 07       4277               CANQ    B$KILL,DL         TEST FOR KILLED
002704  002707 6000 00       4278               TZE     *+3               NOPE
002705  000006 2350 16       4279               LDA     J$RSTRT,J         GET RESTART ADDRESS
002706  004000 2750 03       4280               ORA     B$ABORT,DU        AND ABORTED
002707  000026 7550 11       4281               STA     T$TEMP2,T         SAVE AS MESSAGE
002710  000004 7370 00       4282               LLS     4                 WHICH WILL BE STATE FOR CAUSE
002711  000017 3750 07       4283               ANA     B$BJBMK,DL        MASK TO JOB NUMBER
002712  000027 7550 11       4284               STA     T$TEMP1,T         SAVE AS STATE
002713  003430 4500 05       4285               STZ     J$JTAB,AL         RESET JOB TABLE
                             4286 *
                 002714      4287               BRANCH  NOPASS,C$MESSX,(T$TEMP1,T),(T$TEMP2,T)
002714  001325 7000 00                          TSX     0,T$GETT
002715  000000 6220 11                          EAX     X,0,T
002716  000005 2210 12                          LDX     T,T$LINK,X
 END OF BINARY CARD LPCP0078
002717  000027 2360 11                          LDQ     T$TEMP1,T
002720  000027 7560 12                          STQ     T$TEMP1,X
002721  000026 2360 11                          LDQ     T$TEMP2,T
002722  000026 7560 12                          STQ     T$TEMP2,X
002723  000000 6210 12                          EAX     T,0,X
002724  001613 6200 00                          EAX     0,C$MESSX
002725  000004 7400 11                          STX     0,T$TRA,T
002726  000004 6200 11                          EAX     0,Q$OFFST,T
002727  003262 7170 00                          XED     Q$XADD+Q$TASK
002730  000005 2210 12                          LDX     T,T$LINK,X
                 002731                         BUGXR   (0,X)
                 525270           BUGBUG        SET     BUGBUG+1
002731  525270 2200 03                          LDX     0,BUGBUG,DU
002732  525270 2220 03                          LDX     X,BUGBUG,DU
                             4288 *
                             4289 *             RELEASE RESOURCES
                             4290 *
                 002733      4291               RELP    (J$RES,J)         RELEASE PERIPHERAL
002733  000016 2350 16                          LDA     J$RES,J
002734  001433 7070 00                          TSX     L,R$RELP
002735  000005 2360 16       4292               LDQ     J$BUFSZ,J         GET BUFFER SIZE
002736  777777 3760 07       4293               ANQ     -1,DL             ONLY
002737  000002 4020 07       4294               MPY     J$N,DL            TIMES NUMBER OF BUFFERS
002740  000060 0760 07       4295               ADQ     2*T$LEN,DL        PLUS TCB'S
```

WRITE TASK -- MAIN

```
     002741  000000 5330 00          4296          NEGL                      GET 2'S COMPLEMENT
     002742  000000 6360 06          4297          EAQ      0,QL             MOVE TO QU
     002743  003321 0560 00          4298          ASQ      MEMRQ            REDUCE MEMORY REQUIREMENTS
     002744  777777 6040 00          4299          TMI      $ERROR           OOPS..
                    002745           4300          RELJ                      RELEASE JCB
  END OF BINARY CARD LPCP0079
     002745  001357 7070 00                        TSX      L.J$RELJ
                    002746           4301          RELT                      RELEASE TCB
     002746  001335 7000 00                        TSX      0.T$RELT
                    002747           4302          EXIT
     002747  001547 7100 00                        TRA      $EXIT
```

```
                         002750          USE     CODE
                                         HEAD
                                  *
                                  *
                                  *                                     INITIALIZATION
                                  *
                                  *      THIS ROUTINE INITIALIZES THE PERIPHERAL SCHEDULER
                                  *      MONITOR BY PERFORMING THE FOLLOWING FUNCTIONS.
                                  *
                                  *      FUNCTIONS
                                  *              INITIALIZE REGISTERS AND LOCATION ZERO
                                  *              SET FAULT VECTOR
                                  *              SET UP COMMUNICATIONS NETWORK
                                  *
                                  *      IT IS A NON-REENTRANT ROUTINE WHICH DOES NOT OVERLAP ANY
                                  *      OPERATION.  IT CAN BE OVERLAYED BY BUFFER STORAGE
                                  *      AFTER INITIALIZATION IF DESIRED (NOT IMPLEMENTED YET)
                                  *
                         002750   ANIT   BSS     0                      INITIALIZATION ENTRY
                         002750   UP     EQU     ANIT
                                  *
                                  *      CATCH WILD TRANSFERS TO ZERO
                                  *
002750   000000 4500 00                  STZ     0                      BURN OUR BRIDGES BEHIND US
                         002751          CKPT                           SAVE REGISTERS
002751   000474 7170 00                  XED     X$CKPT
002752   003740 2210 03                  LDX     T,SPTCB,DU             INITIALIZE T TO SPECIAL TCB
002753   003740 2260 03                  LDX     J,SPTCB,DU             AND MAKE J POINT TO SAME
002754   000006 4460 11                  SXL     J,T$JCB,T              SAVE IT FOR $EXIT
002755   004400 6340 07                  LDI     B$OVM+B$PAM,DL         MASK OFF OVERFLOW AND PARITY ERRORS
002756   000010 2350 03                  LDA     ZZ1,DU                 INITIALIZE TO AVAILABLE MEMORY
002757   003320 7550 00                  STA     $AVAIL
002760   003430 5540 00                  STC1    J$JTAB                 MAKE JOB NUMBER 0 ILLEGAL
                                  *
                                  *      SET FAULT VECTOR
                                  *
                         002761   ANIT1  SETFV   (X$FV,DU)              SET FAULT VECTOR
002761   000526 7000 00                  TSX     0,$SETFV
002762   000000 0000 03                  ARG     X$FV,DU
                         002763          CHECK   ANIT2,B$BZ,ANIT1
002763   000000 7200 11                  LXL     0,T$SRW1,T
002764   000077 3600 03                  ANX     0,B$STMK,DU
002765   002772 6000 00                  TZE     ANIT2
002766   000003 1000 03                  CMPX    0,B$BZ,DU
002767   002761 6000 00                  TZE     ANIT1
002770   777777 7100 00                  TRA     $ERROR
                         002771          RELT                           RELEASE TCB
002771   001335 7000 00                  TSX     0,T$RELT
```

```
                                  4343 *
                                  4344 *
                                  4345 *        SET UP COMMUNICATIONS NETWORK
                                  4346 *
                                  4347 *                FRN 0 = INPUT EVENT
                                  4348 *                FRN 1 = OUTPUT EVENT
                                  4349 *                FRN 2 = PASS EVENT
                                  4350 *                STATE DIFFERENTIATES BETWEEN LP AND CP MODULES
                                  4351 *
                                  4352 *                SET UP LP COMMUNICATIONS
                                  4353 *
                        002772    4354 ANIT2  BSS     0
 END OF BINARY CARD LPCP0080
     002772  003450 2210 03       4355          LDX     T,LPNCB0,DU        SET UP LP INPUT LINE
     002773  000006 7260 11       4356          LXL     J,T$JCB,T          GET JCB POINTER
                        002774    4357          BRANCH  PASS,C$NSRVX
     002774  001325 7000 00                     TSX     0,T$GETT
     002775  000000 6220 11                     EAX     X,0,T
     002776  000005 2210 12                     LDX     T,T$LINK,X
     002777  001644 6200 00                     EAX     0,C$NSRVX
     003000  000004 7400 11                     STX     0,T$TRA,T
     003001  000004 6200 11                     EAX     0,Q$OFFST,T
     003002  003262 7170 00                     XED     Q$XADD+Q$TASK
     003003  000000 6210 12                     EAX     T,0,X
                        003004                  BUGXR   (0,X)
                        525271          BUGBUG SET     BUGBUG+1
     003004  525271 2200 03                     LDX     0,BUGBUG,DU
     003005  525271 2220 03                     LDX     X,BUGBUG,DU
                        003006    4358          RELT                       RELEASE TCB
     003006  001335 7000 00                     TSX     0,T$RELT
                                  4359 *
                                  4360 *
     003007  003510 2210 03       4361          LDX     T,LPNCB2,DU        SET UP LP PASS LINE
     003010  000006 7260 11       4362          LXL     J,T$JCB,T          GET JCB POINTER
                        003011    4363          BRANCH  PASS,C$NSRVX
     003011  001325 7000 00                     TSX     0,T$GETT
     003012  000000 6220 11                     EAX     X,0,T
     003013  000005 2210 12                     LDX     T,T$LINK,X
     003014  001644 6200 00                     EAX     0,C$NSRVX
     003015  000004 7400 11                     STX     0,T$TRA,T
     003016  000004 6200 11                     EAX     0,Q$OFFST,T
     003017  003262 7170 00                     XED     Q$XADD+Q$TASK
 END OF BINARY CARD LPCP0081
     003020  000000 6210 12                     EAX     T,0,X
                        003021                  BUGXR   (0,X)
                        525272          BUGBUG SET     BUGBUG+1
     003021  525272 2200 03                     LDX     0,BUGBUG,DU
     003022  525272 2220 03                     LDX     X,BUGBUG,DU
                        003023    4364          RELT                       RELEASE TCB
     003023  001335 7000 00                     TSX     0,T$RELT
```

INITIALIZATION

```
                                   4366 *
                                   4367 *
                                   4368 *                   SET UP CP COMMUNICATIONS
                                   4369 *
   003024  003550 2210 03          4370          LDX       T,CPNCB0,DU       SET UP CP INPUT LINE
   003025  000006 7260 11          4371          LXL       J,T$JCB,T         GET JCB POINTER
                      003026       4372          BRANCH    PASS,C$NSRVX
   003026  001325 7000 00                        TSX       0,T$GETT
   003027  000000 6220 11                        EAX       X,0,T
   003030  000005 2210 12                        LDX       T,T$LINK,X
   003031  001644 6200 00                        EAX       0,C$NSRVX
   003032  000004 7400 11                        STX       0,T$TRA,T
   003033  000004 6200 11                        EAX       0,Q$OFFST,T
   003034  003262 7170 00                        XED       Q$XADD+Q$TASK
   003035  000000 6210 12                        EAX       T,0,X
                      003036                     BUGXR     (0,X)
                      525273          BUGBUG     SET       BUGBUG+1
   003036  525273 2200 03                        LDX       0,BUGBUG,DU
   003037  525273 2220 03                        LDX       X,BUGBUG,DU
                      003040       4373          RELT
   003040  001335 7000 00                        TSX       0,T$RELT
                                   4374 *
                                   4375 *
   003041  003610 2210 03          4376          LDX       T,CPNCB2,DU       SET UP CP PASS LINE
   003042  000006 7260 11          4377          LXL       J,T$JCB,T         GET JCB POINTER
                      003043       4378          BRANCH    PASS,C$NSRVX
   003043  001325 7000 00                        TSX       0,T$GETT
   003044  000000 6220 11                        EAX       X,0,T
   003045  000005 2210 12                        LDX       T,T$LINK,X
 END OF BINARY CARD LPCP0082
   003046  001644 6200 00                        EAX       0,C$NSRVX
   003047  000004 7400 11                        STX       0,T$TRA,T
   003050  000004 6200 11                        EAX       0,Q$OFFST,T
   003051  003262 7170 00                        XED       Q$XADD+Q$TASK
   003052  000000 6210 12                        EAX       T,0,X
                      003053                     BUGXR     (0,X)
                      525274          BUGBUG     SET       BUGBUG+1
   003053  525274 2200 03                        LDX       0,BUGBUG,DU
   003054  525274 2220 03                        LDX       X,BUGBUG,DU
                      003055       4379          RELT                        RELEASE TCB
   003055  001335 7000 00                        TSX       0,T$RELT
                      003056       4380          EXIT                        DONE
   003056  001547 7100 00                        TRA       $EXIT
```

ASSEMBLY CONTROL CARDS

```
                         4382        HEAD
                         4383 *
                         4384 *
                         4385 *                                 HOUSEKEEPING CARDS
                         4386 *
                         4387 *      HERE WE CLEAN ANY AND ALL ASSEMBLER BUGS.
                         4388 *      THE JCB*S ARE PRE-ALLOCATED. THE LAST USED LOCATION OF
                         4389 *      CORE IS CALCULATED AND WHAT REMAINS UP TO THE END OF THE
                         4390 *      BAR IF ANY IS LINKED ON THE FREE MEMORY LIST.
                         4391 *
              003057     4392        USE     CODE
                         4393        HEAD
              003060     4394        EIGHT
              003060     4395 ZCODEL EQU     *-ZCODE          CODE UNDER CODE
                         4396 *
                         4397 *
              003252     4398        USE     CONST
                         4399        LIT                      FORCE LITERAL POOL HERE
              003260     4400        EIGHT
              000200     4401 ZCONSL EQU     *-ZCONS          CODE UNDERCONST
                         4402 *
                         4403 *
              003314     4404        USE     QSTOR
              003320     4405        EIGHT
              000040     4406 ZQSTRL EQU     *-ZQSTR          CODE UNDER QSTOR
                         4407 *
                         4408 *
              003644     4409        USE     STORE
                         4410        HEAD    J
              003644     4411 JCB0   BSS     0                PRE-ALLOCATED JCB*S
              000003     4412 JCBN   EQU     RSLPMAX+RSCPMAX  *NUMBER TO PRE-ALLOCATE
              003644     4413        DUP     1,LEN*JCBN       GENERATE
 003644  000000000000    4414        DEC     0
END OF BINARY CARD LPCP0085
              003740     4415 JCBX   EQU     *                END OF JCB*S
                         4416        HEAD
                         4417 *
              000037     4418 TLEN   EQU     TSLEN+7          ROUND TSLEN TO
              000030     4419 TLENR  EQU     TLEN/8*8         MULTIPLE OF EIGHT
              003740     4420        EIGHT                    START OF DYNAMIC BUFFER AREA
              003740     4421 SPTCB  BSS     TLENR-1          FIRST TCB (START OF DYNAMIC BUFFERS
 003767  000000000000    4422        DEC     0                FOR THE CRUMMY LOADER
              003770     4423        EIGHT
              000450     4424 ZSTORL EQU     *-ZSTOR          CODE UNDER STORE
                         4425 *
                         4426 *
              003770     4427 LASTC  EQU     ZCODEL+ZCONSL+ZQSTRL+ZSTORL  LAST CARD
```

ASSEMBLY CONTROL CARDS

```
                           4429 *
                           4430 *
                  003740   4431 ZTOPO   EQU    SPTCB            START OF DYNAMIC BUFFER AREA
                  004767   4432 ZZ      EQU    LASTC+MQUAN-1    ROUND UP TO NEXT MULTIPLE
                  004000   4433 ZTOP    EQU    ZZ/MQUAN*MQUAN   OF CORE PAGE SIZE
                  000010   4434 ZZ1     EQU    ZTOP-LASTC       LENGTH OF LEFT OVER CORE
                           4435 *
                  003770   4440 NEXTF   EQU    LASTC            STICK EXTRA CORE ON FREE LIST
                  003770   4441 NEXTB   EQU    NEXTF
 003770  003355 000010     4442         ZERO   R$LAST,ZZ1       INITIALIZE HEADER DATA FOR LINKED LIST
 003771  003353 000000     4443         ZERO   R$FIRST,0
                           4444 *
                           4445 *
 END OF BINARY CARD LPCP0086
                  002750   4446         TCD    $UP              MARK END OF BINARY DECK
 END OF BINARY CARD LPCP0087
                           4447 *
                           4448         DCARD  2,$
                           4449 $       DKEND
                           4450 $EOD
                           4451 *
                           4452 *
                           4453 *
                  002750   4454 THIS    END    UP
 END OF BINARY CARD LPCP0090
 3772 IS THE NEXT AVAILABLE LOCATION.   GMAP VERSION  JMPA/062770 JMPB/062770 JMPC/062770
THERE WERE    NO  WARNING FLAGS IN THE ABOVE ASSEMBLY
```

OCTAL	SYMBOL	REFERENCES BY ALTER NO.
2000	1K	657 657 658 2191
2761	ANIT1	4339 4339 4340
2772	ANIT2	4354 4340 4354
2750	ANIT	4322 676 4322 4323
551	APEND	1236 1236 3855 3888
3342	APNDT	1249 1236 1239 1240 1241 1242 1249
3320	AVAIL	661 661 2198 2332 2420 2463 4334
4	B AP	227 227 228 230
3	B BZ	244 244 790 811 [illegible] 2500 2329 3147 3200 3381 3509 3519 3642 3856 3889 4270 4340
2	B CA	232 232
200	B CP	297 297 3221 3226 3849
2	B EX	228 228 229 230
4	B IO	269 269 2500
1	B LK	229 229 230
400	B LP	296 296 3211 3215 3670 3863 3987 4122 4211
4	B NO	231 231 232
0	B OK	241 241
20	B RD	225 225 226 230
10	B WT	226 226 227 230
37	B ALL	230 230 650
100000	B CAR	212 212
0	B CP0	325 325
1	B CP1	326 326
2	B CP2	327 327 4086
3	B CP3	328 328 4095
4	B CP4	329 329
13	B DBZ	251 251
16	B EOF	252 252 3642
20000	B EOV	214 214
10000	B EUN	215 215
2000	B GET	314 314
4	B IOP	245 245
5	B ITN	259 259
0	B LP0	335 335
1	B LP1	336 336
2	B LP2	337 337 4127
3	B LP3	338 338 4068
4	B LP4	339 339
5	B LP5	340 340
6	B LP6	341 341
7	B LP7	342 342
200	B MOD	220 220
200000	B NEG	211 211
40000	B OPR	279 279
40000	B OVF	213 213
4000	B OVM	216 216 4332
400	B PAM	219 219 4332
1000	B PAR	218 218
16000	B RDY	320 320 3880

OCTAL	SYMBOL	REFERENCES BY ALTER NO.
6000	B REL	316 316 2841
2000	B RPT	195 195
2000	B TAL	217 217
11	B TLE	261 261 3173
20	B TMI	201 201
2	B TNC	204 204
40	B TNZ	200 200
1	B TOV	205 205
10	B TPL	202 202
4	B TRC	203 203
35	B TRO	254 254 3211 3215 3221 3226
100	B TZE	199 199
400000	B ZER	210 210
1000	B ABIT	196 196 2184
400	B BBIT	197 197 2184
1	B BHDR	305 305 3471
200	B CBIT	198 198
14000	B DONE	319 319 4275
40000	B ENDR	292 292 293 3650 3792
20000	B ENDW	293 293 294 4226 4263
10000	B HALT	294 294 3373 3393 3631
12	B HDWE	250 250 3356
2	B IACC	243 243
7	B IELT	248 248
1	B IFRN	242 242
11	B IMOD	249 249
5	B IPTR	246 246
16	B IPWD	263 263
6	B IREQ	247 247
400000	B KILL	289 289 290 3308 3633 4183 4277
13	B LOCK	262 262
10	B LP10	343 343
11	B LP11	344 344
12	B LP12	345 345
13	B LP13	346 346
14	B LP14	347 347
15	B LP15	348 348
26	B NSTR	253 253
200000	B RHDR	290 290 291 3473 3635 3665
400000	B SIGN	275 275 289 2961
77	B STMK	274 274 2500 2829 3147 3173 3200 3381 3509 3519 3642 3856 3889 4270 4340
400000	B TERM	276 276 277
7	B UERR	260 260
100000	B WHDR	291 291 292 3473 4185 4209
10000	B XXXX	317 317
4000	BABORT	315 315 4280
17	BBJBMK	303 303 3452 3466 4283
1	BBOTMK	307 307 3476
7777	BBSTMK	309 309 3484
200000	BDELIM	277 277 278

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO.

100000    BDIGIT      278   278   279
 20000    BHDRMK      304   304
740000    BJOBMK      302   302
  1700    BMODMK      301   301  3984
 10000    BOUTMK      306   306  4187
 12000    BRSTRT      318   318
  7777    BSRTMK      308   308
  3100    BSZ        3480  3477  3480
     U    BUFSEG      656   656  2499
525200    BUG         615   615   616
525274    BUGBUG     4378   616  1748  1749  1750  1933  1934  1935  1989  1990  2343  2346  2503  2583  2665  2762
                           2764  2766  2767  2810  2824  2843  2852  2853  2854  2855  2856  2857  2859  2997  3001
                           3066  3067  3068  3069  3654  3655  3656  3657  3658  3882  3883  3884  3895  3896  3897
                           3898  3899  3900  3901  4005  4006  4007  4200  4201  4202  4287  4357  4363  4372  4378
     6    C    J           3307  3341  3342  3371  3372
     7    C    L           3175  3275  3300  3324  3347  3360
     5    C    Q
     1    C    T           3194  3197
     2    C    X           3148  3193  3282  3283  3284  3285  3413
     3    C    Y
     4    C    Z           3346  3361  3415  3423
    30    C  ERN      500   500   501  3199
  1703    C  GET     3274  3247  3274
     6    C  JCB      482   482  3211  3215  3221  3226
     6    C  NCB      481   481  3211  3215  3221  3226
  1724    C  REL     3323  3249  3323
    33    C  RES      503   503  3413  3463  3501
     2    C  RET      476   476
     4    C  TRA      478   478
     3    C  XED      477   477
     6    C CMAX     3252  3243  3252
  1666    C COMD     3239  3211  3221  3239  3453
  1714    C KILL     3299  3248  3299
     5    C LINK      479   479   480
    32    C MESS      502   502   503
  1634    C NSRV     3172  3172  3195
  1652    C NSX1     3198  3198  3200
  1664    C NSX2     3201  3200  3201
  2005    C PERI     3412  3275  3300  3324  3360  3412
  1734    C REL2     3338  3338
  1762    C RST1     3380  3380  3381  3389
  1773    C RST2     3382  3381  3382
     0    C SRW1      474   474
     1    C SRW2      475   475
    17    C TEM9      492   492
   657    CAUSE      1603  1603  3146
  3352    CAUST      1618  1603  1606  1607  1608  1609  1610  1611  1618
  1674    CCMDTB     3246  3246  3252
  1702    CCOMDX     3260  3244  3260  3281  3286  3306  3310  3330  3334  3348  3367  3395  3426
   605    CHSEG      1373  1373  2499
```

```
OCTAL    SYMBOL    REFERENCES BY ALTER NO.

 3345    CHSGT      1384  1373  1376  1377  1384
  777    CKMK        618   618
  616    CLOSE      1417  1417  2828  4269
 3346    CLOST      1427  1417  1420  1427
 1630    CMESS1     3148  3147  3148
 1613    CMESSX     3145  2843  3145  3147  3149  3882  4287
 1642    CNSRV1     3174  3173  3174
 1644    CNSRVX     3191  3173  3191  3261  3562  4357  4363  4372  4378
 2015    CPERI1     3421  3421  3425
 3172    CPHDR      3735  3679  3735  3755
   33    CPHLN      3755  3681  3755  4216
 2542    CP.0       4073  4036  4073
 2550    CP.1       4081  4037  4078  4081
 2556    CP.2       4089  4038  4076  4089
 2556    CP.3       4090  4039  4090
 3550    CPNCB0     3221  3221  4370
 3610    CPNCB2     3226  3226  4376
 1000    CPST        638   638  3221  3226
 3232    CPTAB      4035  3988  4035
    5    CRLINK      480   480  3175
 1747    CRSTRT     3359  3251  3359
    7    CSPARE      483   483
   31    CSTATE      501   501   502  2835  3199  3343  3874
   16    CTEM10      493   493
   15    CTEM11      494   494
   14    CTEM12      495   495
   13    CTEM13      496   496
   12    CTEM14      497   497
   11    CTEM15      498   498
   10    CTEM16      499   499
   27    CTEMP1      484   484
   26    CTEMP2      485   485
   25    CTEMP3      486   486
   24    CTEMP4      487   487
   23    CTEMP5      488   488
   22    CTEMP6      489   489
   21    CTEMP7      490   490
   20    CTEMP8      491   491
   37    DACC        650   650   778
    1    DBG          39    39    42   734  1137  1190  1244  1290  1333  1379  1422  1465  1508  1555  1613  1748
                            1749  1750  1933  1934  1935  1989  1990  2343  2346  2503  2583  2665  2762  2764  2766
                            2767  2810  2824  2843  2852  2853  2854  2855  2856  2857  2859  3042  3066  3067  3068
                            3069  3654  3655  3656  3657  3658  3882  3883  3884  3895  3896  3897  3898  3899  3900
                            3901  4005  4006  4007  4200  4201  4202  4287  4328  4357  4363  4372  4378
44000    DESZ        649   649   777
 3060    DTN         646   646   648   774
   14    DTSZ        648   648   775
 2622    EMPT1      4195  4008  4130  4184  4195  4221
 2625    EMPT2      4198  4198  4228
 2640    EMPT3      4207  4186  4207
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO.

   2650   EMPT4     4215  4212  4215
   2656   EMPTX     4225  4180  4225
   2576   EMPTY     4171  4171  4260
 777777   ERROR      614   614  1754  1755  1792  1798  1799  1986  1987  2006  2017  2141  2143  2167  2199  2270
                          2273  2277  2421  2426  2500  2552  2553  2579  2633  2663  2736  2742  2750  2805  2809
                          2829  3049  3055  3056  3061  3062  3064  3065  3147  3173  3200  3246  3250  3381  3468
                          3509  3512  3519  3642  3856  3889  4123  4270  4299  4340
   1550   EXIT1     3044  3044  3080
   1547   EXIT      3041  1139  1192  1246  1292  1335  1381  1424  1467  1510  1557  1615  1757  1939  2468  3041
                          3154  3203  3690  3799  4302  4380
   2204   FILL0     3630  3630  3642  3686
   2213   FILL1     3641  3641
   2230   FILL2     3646  3634  3642  3646
   2236   FILL3     3652  3642  3652  3677  3684
   2256   FILL4     3664  3636  3664
   2271   FILL5     3678  3671  3678
   2277   FILL6     3685  3632  3685
   2171   FILL      3619  3619  3789
      0   FRN0       632   632  3211  3221
      1   FRN1       633   633  3146
      2   FRN2       634   634  3215  3226
   2465   GCOS1     3966  3966  4000
   2516   GCOS2     3998  3998
   2517   GCOS3     3999  3999
   2520   GCOS4     4004  3959  3976  3980  4004  4100
   2470   GCOS5     3970  3962  3970
   2513   GCOS7     3994  3994  4069  4087  4096
     10   GEBORT
     22   GECALL
     27   GECHEK
   2455   GECOS     3954  3954
     16   GEENDC
      3   GEFADD
     12   GEFCON
     13   GEFILS
      7   GEFINI
     40   GEFRCE
     41   GEFSYE
     35   GEIDSE
     45   GEINFO
      1   GEINOS
      6   GELAPS
     37   GELBAR
     33   GELOOP
     11   GEMORE
     25   GEMREL
     43   GENEWS
     42   GEPRIO
     17   GERELC
      4   GERELS
```

OCTAL	SYMBOL	REFERENCES BY ALTER NO.							
15	GERETS								
2	GEROAD								
31	GEROLL								
30	GEROUT								
24	GERSTR								
23	GESAVE								
14	GESETS								
5	GESNAP								
44	GESNUM								
20	GESPEC								
26	GESYOT								
21	GETIME								
32	GEUSER								
34	GEWAKE								
2564	IGNOR	4099	4022	4025	4027	4041	4074	4079	4099
2104	INIT2	3504	3504	3509					
2114	INIT3	3510	3509	3510					
2120	INIT4	3517	3517	3519	3527				
2131	INIT5	3520	3519	3520					
2023	INIT	3449	3215	3226	3449				
2455	.320	3953	3953	4191					
2565	.512	4119	4119	4190					
5	.APEND	141	141	1243					
32	.CATDR	162	162						
26	.CATLG	158	158						
42	.CATLK	170	170						
47	.CAUSE	175	175	1612					
22	.CHSEG	154	154	1378					
25	.CLOSE	157	157	1421					
21	.CLSEG	153	153						
57	.CRSG	181	181						
50	.DELET	176	176						
27	.DESTR	159	159						
36	.EMM								
23	.EXSEG	155	155						
44	.LOCK	172	172	1464					
55	.MSTA	179	179						
46	.NOTIF	174	174	1554					
24	.OPEN	156	156	772					
30	.OPENS	160	160						
36	.OPENW	166	166						
52	.OPSCE	178	178						
20	.OPSEG	152	152						
17	.PAUSE	151	151	783	805	830	3077		
0	.PRIV	136	136						
34	.RDACL	164	164						
37	.RDBRN	167	167						
35	.RDDIR	165	165						
40	.RDLK	168	168						
56	.RDME	180	180						

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO.

     4    .READ      140   140  1189
    13    .RQDT      147   147   765
    14    .RQERT     148   148
    12    .RQST      146   146  1332
    63    .RQWD      184   184
     6    .RRF       142   142
    10    .SCR       144   144   797
     1    .SETFV     137   137  1136
     2    .SETSQ     138   138
    15    .SPAWN     149   149
    11    .SPTR      145   145  1289
     3    .SQUEZ     139   139
    16    .TERM      150   150   839
    51    .UNCAU     177   177
    45    .UNLCK     173   173  1507
    31    .UPDAT     161   161
    61    .WAMI      183   183
    43    .WBRAN     171   171
    33    .WRACL     163   163
     7    .WRF       143   143   818
    41    .WSINF     169   169
    60    .WTME      182   182
     6    J     J          2629  2634  2638  2665
     7    J     L
     2    J     N    455   455   456   458  3488  3497  3557  4294
     5    J     Q
     1    J     T          2635
     2    J     X
     3    J     Y
     4    J     Z
    24    J   LEN    456   456  2631  2634  2636  2638  4413
     7    J   RCW    436   436  3967  3969  3972  3975  3981  3983
    16    J   RES    449   449   450  3502  3876  4291
777777    J ALLC     427   427   456  2635  2662  2664
     1    J FLAG     429   429  3460
    12    J FULL     441   441  3490  3790  4259
  1344    J GETJ    2628  2628  3454
     2    J IFRN     430   430  3380  3487  3508  3518  3641  4269
  3644    J JCBO    4411  2629  4411
     3    J JCBN    4412  2631  4412  4413
  3740    J JCBX    4415  4415
  3430    J JTAB    3020  3020  3467  3469  4285  4335
     3    J OFRN     431   431  3503  3855  3888
  1357    J RELJ    2661  2661  4300
    13    J RPTR     443   443  3492  3622  3627  3648  3673  3680
    17    J RTRY     450   450   451  3847  3867  3868
    14    J WPTR     445   445  3493  3955  4173  4178  4197
     6    J          607   607  2632  2634  2635  2637  2638  2662  2664  3060  3309  3374  3378  3380  3385  3394
                          3455  3460  3462  3464  3469  3474  3478  3485  3487  3489  3490  3491  3492  3493  3496
                          3500  3502  3503  3508  3511  3518  3523  3532  3544  3555  3621  3622  3624  3626  3627
```

```
OCTAL   SYMBOL    REFERENCES BY ALTER NO.

                         3630 3641 3648 3651 3666 3673 3680 3788 3790 3791 3847 3848 3851 3853 3855
                         3862 3867 3868 3876 3888 3955 3956 3961 3967 3969 3972 3975 3978 3981 3983
                         3986 4121 4173 4175 4177 4178 4182 4196 4197 4208 4210 4227 4259 4261 4262
                         4269 4276 4279 4291 4292 4330 4331 4356 4362 4371 4377
    5   JBUFSZ     434    434 3478 3511 3555 3621 4292
    4   JCARD1     433    433 3851 3853
   11   JEMPTY     439    439 3489 3788 4261
    0   JFLAGS     428    428 3309 3374 3394 3462 3464 3474 3630 3651 3666 3791 3848 3862 3986 4121
                         4182 4210 4227 4262 4276
   20   JMAXJB    3016   3016 3021
   20   JRESET     451    451  456 3491 3496 3500 3624 3626 4175 4177
    6   JRSTRT     435    435 3378 3385 3485 3523 4196 4208 4279
   15   JRTASK     447    447  448 3532
   15   JWTASK     448    448  449 3544
   10   JXDATA     438    438 3956 3961 3978
    7   L          608    608 1751 1796 1806 1810 1936 1983 1991 2201 2347 2366 2378 2504 2639 2666
                         2768 2860 3063 3070 3176 3422 3454 3501 3620 3621 3647 3653 3659 3788 3789
                         3790 3846 3894 3902 3998 4004 4129 4172 4197 4199 4203 4220 4259 4260 4261
                         4291 4300
 3770   LASTC     4427   4427 4432 4434 4440
  626   LOCK      1460   1460
 3347   LOCKT     1470   1460 1463 1470
 2571   LP1       4125   4125 4214
 2565   LP512     4120   4120
 3102   LPHDR     3705   3672 3705 3727
   70   LPHLN     3727   3674 3727 4213
 2534   LP.0      4061   4021 4061
 2534   LP.2      4062   4023 4062
 2534   LP.3      4063   4024 4063
 3450   LPNCB0    3211   3211 4355
 3510   LPNCB2    3215   3215 4361
 3000   LPST       639    639 3211 3215
 3225   LPTAB     4020   3989 4020
 3321   MEMRQ      662    662 2337 2423 3560 4298
 1000   MQUAN      617    617  662 2227 2232 2233 2335 2451 2453 2496 2497 4432 4433
 3322   MTOP0      663    663 2267 2334 2418
 3323   MTOP       664    664 2226 2228 2231 2268 2333 2417 2462 2495 2498 2499
 3770   NEXTB     4441   2049 4441
 3770   NEXTF     4440   2044 4440 4441
 3351   NOTFT     1560   1549 1552 1553 1560
  646   NOTIF     1549   1549 3199
    0   OFF         36     36
    1   ON          37     37   39   42  734 1137 1190 1244 1290 1333 1379 1422 1465 1508 1555 1613
                         1748 1749 1750 1933 1934 1935 1989 1990 2343 2346 2503 2583 2665 2762 2764
                         2766 2767 2810 2824 2843 2852 2853 2854 2855 2856 2857 2859 3042 3066 3067
                         3068 3069 3654 3655 3656 3657 3658 3882 3883 3884 3895 3896 3897 3898 3899
                         3900 3901 4005 4006 4007 4200 4201 4202 4287 4328 4357 4363 4372 4378
    6   Q    J
    7   Q    L           1752 1937 2006 2017
  751   Q    P    1928   1928 3788 4259
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO.

    5     Q     Q              1750  1935  2006  2017
    1     Q     T              1746  1938
  772     Q     V        1978  1978  3790  4261
    2     Q     X              1750  1800  1802  1803  1935  1979  1980  1981  1990
    3     Q     Y              1750  1935
    4     Q     Z              1750  1929  1930  1931  1935
  767     Q    P1        1937  1932  1937
  724     Q   DEQ        1789  1789  2006  2017
  674     Q   ENQ        1738  1738  2006  2017
   16     Q   LEN         554   554
   11     Q   MAX         549   549   550  1741  1743
   15     Q  ABBR         553   553   554
   10     Q  BUSY         547   547   549  1744  1746  1791  1793  1801  2339
 3276     Q  CORE        2017  2017  2136  2176  2339  2416  2466
  716     Q  ENQ1        1752  1745  1752
    1     Q  LAST         543   543   544  1053  1055  1795  1805  1808  2006  2017  3046  3050  3053
    3     Q  LINK         558   558  1802  3057
 3260     Q  TASK        2006  1053  1055  1804  1988  2006  2343  2843  3045  3046  3048  3050  3052  3053  3058  3539
                               3551  3689  3882  4287  4357  4363  4372  4378
    2     Q  XADD         544   544   545  1756  1804  1988  2343  2843  3539  3551  3689  3882  4287  4357  4363  4372
                               4378
    6     Q  XDEQ         546   546   547  2176  2466
    4     Q  XENQ         545   545   546  2136  2416
    5     Q               606   606  1739  1740  1741  1743  1744  1746  1756  1790  1791  1793  1794  1795  1797  1801
                               1803  1805  1807  1808  1929  1931  1938  1979  1981  1984  1989  3067  3656  3788  3790
                               3898  4005  4202  4259  4261
   12     QAVAIL          550   550   551  1739  1740  1790
    0     QFIRST          542   542   543  1794  1797  1803  1807  2006  2017  3045  3048  3052  3058
    4     QOFFST          557   557   558  1753  1800  1985  2343  2843  3054  3535  3547  3688  3882  4287  4357  4363
                               4372  4378
   13     QSPAR1          551   551   552
   14     QSPAR2          552   552   553
    6     R     J              2342  2757
    7     R     L              2229  2234  2321  2325  2464  2494  2502  2731  2763  2798  2858
    5     R     Q              2762  2855
    1     R     T              2341  2343  2843
    2     R     X              2163  2170  2172  2173  2186  2197  2198  2226  2227  2228  2271  2278  2283  2285  2298
                               2307  2322  2331  2332  2333  2334  2335  2337  2343  2417  2418  2419  2420  2422  2444
                               2461  2462  2463  2758  2759  2762  2803  2837  2843  2848  2855
    3     R     Y              2162  2175  2187  2294  2297  2443  2447  2450  2451  2453  2454  2762  2822  2823  2855
    4     R     Z              2149  2155  2158  2168  2171  2182  2191  2293  2299  2300  2323  2324  2372  2432  2437
                               2445  2449  2733  2734  2735  2737  2762  2834  2855
 3416     R    CP        2997  2930  2997
 3421     R    LP        3001  2934  3001
    2     R    TT        3001  2997  3001
    1     R   FRN        2953  2761  2822  2824  2953  2954  3283
    0     R   LEN         570   570  2164  2169  2275  2363  2373  2375  2377  2443  2450  2455  2457
    1     R   MAX        2908  2741  2908  2909  3420
    0     R   PTR        2907  2743  2907  2908  3415
    1     R   SET        3001  2997  3001
```

OCTAL	SYMBOL	REFERENCES BY ALTER NO.
2	R ALLC	2955 2757 2807 2810 2955 2956 3307 3341 3371
400000	R BUSY	2961 2744 2758 2849 2961 2962 2997 3001 3284 3305 3333 3339 3366
2	R FLAG	2954 2745 2759 2804 2814 2821 2850 2954 2955 3279 3285 3304 3328 3332 3340 3365
1011	R GETC	2134 2134 2549 3621
1364	R GETP	2730 2730 3501
3355	R LAST	2048 2048 2307 2444 4442
1241	R MEM1	2433 2433 2438
1250	R MEM2	2443 2435 2436 2443
1260	R MEM3	2451 2448 2451
1276	R MEMX	2465 2427 2439 2452 2465
1077	R MORE	2225 2157 2225
3400	R MREG	2384 2264 2272 2331 2345 2384 2385 2386
1301	R MREQ	2494 2229 2464 2494
0	R NAME	2952 2838 2952 2953 2956 3419 3421 3877
1110	R RELC	2263 2234 2263 2582 3647 4197
1433	R RELP	2797 2797 3347 4291
100000	R RSVE	2963 2744 2815 2820 2963 3284 3331
3406	R TEMP	2386 2266 2269 2271 2274 2276 2374 2376 2386
3360	R TRAP	2383 2341 2342 2383 2384
200000	RCLOSE	2962 2744 2820 2962 2963 2997 3001 3280 3284 3329
1	RCPMAX	2997 2930 2997 4412
3410	RCPTAB	2930 2882 2930
3410	RDEVHR	2926 2926
3	RDEVLN	2956 2747 2956 3423
3416	RDEVTB	2993 2993
536	READ	1182 1182 3641
3341	READT	1195 1182 1185 1186 1187 1188 1195
3353	RFIRST	2044 2044 2149 2278 2432 4443
1013	RGETC1	2137 2137
1024	RGETC3	2147 2147 2235
1051	RGETC4	2173 2166 2173
1072	RGETC5	2196 2196
1026	RGETC6	2150 2150 2156
1035	RGETC7	2158 2153 2154 2158
1065	RGETC8	2188 2188 2192
1377	RGETP1	2745 2745 2749
1405	RGETP2	2755 2746 2755
3404	RINSRT	2385 2285 2293 2324 2364 2385
1	RLINKB	571 571 2162 2171 2174 2175 2294 2296 2297 2299 2372 2445 2449 2459 2460
0	RLINKF	569 569 570 571 2163 2170 2173 2283 2298 2300 2365 2370 2371
2	RLPMAX	3001 2934 3001 4412
3413	RLPTAB	2934 2883 2934
1223	RMEMCK	2415 2343 2415
1103	RMORE1	2230 2230
1306	RMREQ1	2499 2499 2500
1321	RMREQ2	2501 2500 2501
4001	RQMAX	658 658 2142
575	RQST	1328 1328 3508
3344	RQSTT	1338 1328 1331 1338
1134	RRELC1	2292 2292 2308

```
OCTAL    SYMBOL     REFERENCES BY ALTER NO.

 1144    RRELC2     2306  2284  2306
 1146    RRELC3     2320  2302  2320
 1206    RRELC4     2362  2321  2325  2362
 1202    RRELCX     2344  2336  2338  2340  2344
 1457    RRELP1     2828  2828  2829
 1467    RRELP2     2830  2829  2830
 1517    RRELP5     2847  2816  2847
 1522    RRELP6     2851  2851
    2    RSPARE     2909  2909
 3075    RTABCP     2882  2882  2888
 3074    RTABLE     2880  2735  2880  2888  2889  3414
 3076    RTABLP     2883  2883  2889
 2304    RTASK      3787  3533  3787  3793
    5    RTMAX       622   622  3869
 2314    RTSK1      3797  3797
    1    RTYPCP     2888  2888  3221  3226
    2    RTYPLP     2889  2889  3211  3215
 3340    SETFT      1142  1132  1135  1142
  526    SETFV      1132  1132  4339
  510    SETUP      1038  1038  1134  1184  1238  1286  1330  1375  1419  1462  1505  1551  1605  3196
 3740    SPTCB      4421  4329  4330  4421  4431
  564    SPTR       1284  1284  3380  3518
 3343    SPTRT      1295  1284  1287  1288  1295
    6    T    J           2554
    7    T    L           2549  2582
    5    T    Q
    1    T    T           2550  2551  2578  2583
    2    T    X
    3    T    Y
    4    T    Z
    6    T  JCB      372   372   482  2554  2808  3060  3342  3372  3455  4331  4356  4362  4371  4377
   30    T  LEN      374   374   377   500  2383  2384  2549  2581  3558  4295  4418
    6    T  NCB      371   371   372   481  2834  3537  3549  3873
    2    T  RET      361   361   476
    4    T  TRA      366   366   478  1039  1752  1937  2343  2731  2763  2764  2798  2843  2858  2859  3063  3066
                          3534  3546  3620  3653  3658  3687  3882  4172  4199  4287  4357  4363  4372  4378
    3    T  XED      362   362   477  1044
 1325    T GETT     2548  2343  2548  2843  3192  3531  3543  3882  4287  4357  4363  4372  4378
    5    T LINK      370   370   479  2343  2550  2843  3194  3198  3536  3548  3882  4287  4357  4363  4372  4378
 1335    T RELT     2577  2467  2577  3153  3202  3798  4301  4341  4358  4364  4373  4379
    0    T SRW1      359   359   474  1042  2500  2829  3147  3148  3173  3200  3282  3381  3486  3509  3510  3519
                          3642  3856  3889  4270  4340
    1    T SRW2      360   360   475  3240  3383  3416  3450  3459  3521  3861
   17    T TEM9      388   388   389   492  2494  2502  2503
    1    T           602   602  1039  1042  1044  1752  1753  1937  2135  2138  2144  2146  2148  2158  2165  2168
                          2180  2186  2197  2200  2343  2419  2422  2424  2425  2447  2454  2456  2461  2494  2500
                          2502  2503  2554  2578  2580  2731  2732  2733  2737  2756  2760  2763  2764  2765  2766
                          2767  2798  2802  2803  2808  2823  2828  2829  2834  2836  2837  2842  2843  2848  2852
                          2853  2854  2858  2859  3048  3050  3054  3057  3060  3063  3066  3146  3147  3148  3173
                          3175  3193  3194  3198  3200  3240  3282  3342  3343  3345  3346  3347  3361  3372  3379
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO.

                         3380  3381  3383  3388  3413  3416  3450  3455  3459  3463  3486  3501  3509  3510  3516
                         3518  3519  3521  3526  3532  3534  3535  3536  3537  3538  3544  3546  3547  3548  3549
                         3550  3620  3629  3641  3642  3647  3653  3657  3658  3687  3688  3846  3855  3856  3861
                         3873  3875  3881  3882  3883  3884  3889  3894  3895  3896  3897  3899  3974  3977  3996
                         3997  4004  4126  4128  4172  4181  4199  4217  4219  4270  4281  4284  4287  4329  4331
                         4340  4355  4356  4357  4361  4362  4363  4370  4371  4372  4376  4377  4378
   100    TAL       621   621
777700    TALMK     620   620
    40    TALYB     619   619
  3772    THIS     4454  4454
    37    TLEN     4418  4418  4419
    30    TLENR    4419  4419  4421
   521    TRAP     1052  1043  1052
     7    TSPARE    373   373   483
    16    TTEM10    389   389   390   493  2146  2165  2168  2197
    15    TTEM11    390   390   391   494
    14    TTEM12    391   391   392   495  2138  2144  2148  2158  2180  2186
    13    TTEM13    392   392   393   496
    12    TTEM14    393   393   394   497
    11    TTEM15    394   394   395   498
    10    TTEM16    395   395   499  2135  2200
    27    TTEMP1    377   377   378   484  2419  2732  2733  2756  2760  2765  2766  2802  2803  2837  2843  2848
                         2852  3146  3345  3346  3347  3361  3516  3518  3526  3629  3641  3647  3657  3855  3882
                         3895  3974  3977  4181  4284  4287
    26    TTEMP2    378   378   379   485  2422  2424  2737  2767  2836  2843  2853  3146  3379  3380  3388  3855
                         3882  3896  3996  4126  4217  4281  4287
    25    TTEMP3    379   379   380   486  2425  2447  2823  2828  2842  2843  2854  3855  3897  3997  4128  4219
    24    TTEMP4    380   380   381   487  2454  2456  2461  3875  3882  3883
    23    TTEMP5    381   381   382   488  3881  3882  3884
    22    TTEMP6    382   382   383   489  3846  3894  3899  4004
    21    TTEMP7    383   383   384   490
    20    TTEMP8    384   384   388   491
   636    UNLCK    1503  1503
  3350    UNLKT    1513  1503  1506  1513
  2750    UP       4323  4323  4446  4454
  1610    WAIT     3077  3047  3077
  2316    WRITE    3845  3845  3998  4129  4220
  2327    WT1      3855  3850  3852  3855  3856  3870  3889
  2344    WT2      3857  3856  3857
  2357    WT2.1    3872  3854  3864  3866  3872
  2415    WT3      3888  3888  3889
  2430    WT4      3893  3856  3871  3893
  2661    WTASK    4258  3545  4258  4264
  2671    WTSK1    4265  4265  4270
  2701    WTSK2    4271  4270  4271
     6    X     J
     7    X     L
     5    X     Q
     1    X     T
     2    X     X
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO.

    3     X     Y
    4     X     Z
    6     X    FT     682   682
    0     X    FV     676   676  4339
   51     X    IC     718   718   759   762
   24     X    OP     696   696
   34     X   DIV     704   704
   14     X   DRL     688   688
  401     X   FT1     771   771   791
  425     X   FT2     796   788   796   812
  445     X   FT3     817   810   817   837
   50     X   IC1     717   717   764
    2     X   MEM     678   678
    4     X   MME     680   680
   26     X   ONC     698   698
  160     X   REG     736   736   994   998  1003
   36     X   XEC     706   706
 3324     X CKIC     1009   991  1005  1009
  474     X CKPT      990   990  1137  1190  1244  1290  1333  1379  1422  1465  1508  1555  1613  3042  4328
   52     X DATE      722   722   767
  170     X DBGQ      739   736   739  1000  1002
   16     X LOCK      690   690
   40     X REGS      716   716   761
   55     X SBAR      725   725   823   824
   30     X STRT      700   700
  470     X TERM      838   792   813   835   838
   53     X TIME      723   723
    2     X           603   603  1802  2171  2174  2175  2187  2189  2275  2283  2294  2299  2343  2370  2373  2445
                           2449  2460  2804  2807  2810  2814  2821  2822  2824  2838  2843  2850  3057  3058  3067
                           3197  3414  3463  3464  3486  3487  3495  3496  3498  3499  3500  3510  3511  3533  3534
                           3536  3537  3538  3545  3546  3548  3549  3550  3622  3623  3624  3626  3627  3628  3648
                           3649  3656  3672  3675  3679  3682  3861  3873  3874  3876  3877  3882  3898  3955  3956
                           3957  3968  3969  3971  3972  3973  3974  3975  3977  3978  3986  3987  3997  4005  4068
                           4086  4095  4124  4126  4173  4174  4175  4177  4178  4179  4202  4213  4216  4217  4287
                           4357  4363  4372  4378
 3330     XCKREG     1011   993  1004  1011
   12     XCOMND      686   686
   20     XCONCT      692   692
   20     XDBGQN      737   737   739  1000
  370     XFAULT      758   677   679   681   683   685   687   689   691   693   695   697   699   701   703   705
                            707   758
   32     XOFLOW      702   702
   22     XPARTY      694   694
   56     XPATCH      730   730
   10     XTIMER      684   684
  414     XTRAP1      786   773   786   787   789   799   820
  435     XTRAP2      808   798   808   809
  461     XTRAP3      833   819   833   834
    3     Y           604   604  2173  2190  2300  2323  3067  3656  3673  3676  3680  3683  3898  4005  4127  4128
                           4202  4218  4219
```

```
OCTAL    SYMBOL     REFERENCES BY ALTER NO.

    4    Z            605   605  2152  2155  2162  2163  2164  2169  2170  2172  2296  2297  2298  2322  2363  2365
                           2371  2375  2377  2434  2437  2443  2450  2455  2457  2459  2735  2741  2743  2835  3067
                           3279  3283  3285  3304  3307  3328  3332  3340  3341  3365  3371  3419  3421  3656  3898
                           3957  3958  3960  3961  3967  3968  4005  4202
    0    ZCODE        107   107  4395
 3060    ZCODEL      4395  4395  4427
 3060    ZCONS        111   111  4401
  200    ZCONSL      4401  4401  4427
 3260    ZQSTR        115   115  4406
   40    ZQSTRL      4406  4406  4427
 3320    ZSTOR        119   119  4424
  450    ZSTORL      4424  4424  4427
 3740    ZTOPO       4431   663  4431
 4000    ZTOP        4433   664  4433  4434
   10    ZZ1         4434   661  4333  4434  4436  4442
 4767    ZZ          4432  4432  4433
** 20140 WORDS OF MEMORY WERE USED BY GMAP FOR THIS ASSEMBLY.
```

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Dartmouth College
Department of Mathematics
Hanover, New Hampshire

2a. REPORT SECURITY CLASSIFICATION

2b. GROUP

3. REPORT TITLE
Design and Implementation of an Input/Output Scheduler for the time-sharing system of the General Electric Corporate Research and Development Center

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Michael B. Rubens

5. AUTHOR(S) *(First name, middle initial, last name)*
Michael B. Rubens

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1972	Vol #1-280;Vol#2-152	8

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F 44620-68-C-00015	
b. PROJECT NO. 9744	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT
This document has been approved for public release and sale; its distribution is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH, other	Air Force Office of Scientific Research (SRMA) 1400 Wilson Blvd., Arlington, Va. 22209

13. ABSTRACT
The problem is to design and implement an Input/Output Scheduler for the General Electric Corporate Research and Development Center. Given the Center's current time-sharing environment of master and slave modes, a slave mode scheduling is proposed. This system is composed of two distinct levels: a monitor, which handles all external input/output and scheduling, comprises the upper level; the lower level contains all the peripheral driver modules, which, while also operating in slave mode, transfer the data to/from such peripheral devices as line printers and card punches. Just such a system has been successfully written and is operating on the Center's computer system.