Price: $5.25

SDS 940 TIME-SHARING SYSTEM
(Version 4.0)

TECHNICAL MANUAL

SDS 90 11 16C March 1969

SCIENTIFIC DATA SYSTEMS/701 South Aviation Boulevard/El Segundo, California 90245

Printed in U.S.A.

REVISION

This publication, SDS 90 11 16C, is a revision of the SDS 940 Time-Sharing System Technical Manual, 90 11 16B (dated August, 1968). Although the general organization of the manual remains the same, drawings have been corrected, new material has been added, and existing text has been rewritten. Any change from the previous manual is indicated by a vertical line in the margin of the page.

RELATED PUBLICATIONS

Title	Publication No.
SDS 940 Computer Reference Manual	90 06 40
SDS 940 Terminal Users Guide	90 11 18
SDS 940 FORTRAN II Reference Manual	90 00 10
SDS 940 BASIC Reference Manual	90 11 11
SDS 940 TAP Reference Manual	90 11 17
SDS 940 DDT Reference Manual	90 11 13
SDS 940 CAL Reference Manual	90 11 14
SDS 940 QED Reference Manual	90 11 12
SDS 940 FORTRAN II Technical Notes	90 11 42
SDS 940 Conversational FORTRAN Reference Manual	90 15 79

CONTENTS

APPENDIXES

ILLUSTRATIONS

TABLES

PREFACE

This manual describes the SDS 940 Time-Sharing System (Version 4.0). The design and implementation of the system is explained, as well as certain of its operational features. The manual covers this in three major parts: Monitor, Executive, and subsystems.

Chapters 2-11 deal with the Monitor, chapters 12-19 discuss the Executive, and chapters 20-27 explain the various system programmed operators (SYSPOPs) and branch system routines (BRS) that can be used with this system.

Illustrations and explanations are also given of important tables associated with the system, such as the PAC Table (PACT), Phantom User Queue Entry, Job Table, Pseudo Memory Table, etc.

This publication is a reference guide for experienced programmers. It assumes that the reader is familiar with the basic concepts of the SDS 940 Time-Sharing System. Additional information about the system can be obtained from the related publications list.

1. INTRODUCTION

The SDS Time-Sharing System (TSS) links the 940 CPU with up to 40 remote terminals. The system consists of three main parts: Monitor, Executive, and subsystems. The monitor executes in monitor mode while the Executive and the subsystems execute in user mode.

Both RAD and disc storage are required, the RAD for storing the subsystems and for swapping and the disc for user's files, file directories, accounting information, and copies of the Monitor and Executive

THE MONITOR

The Monitor is the portion of the system concerned with

- scheduling
- input/output operations
- interrupt processing
- memory allocation
- swapping of programs and data from disc and RAD to and from core memory
- control of active programs

Initially when the system is put into operation, the computer is operating in the 930 mode. The execution of an EOM 022000B puts the computer into the 940 Monitor mode. When the Monitor executes a branch instruction with bit 0 of the branch instruction set, user hardware relabeling is invoked and a transfer is made to user mode. Once the computer is in user mode, it will revert to Monitor mode by the occurrence of an interrupt or trap, or the execution of a SYSPOP (System Programmed Operator).

The basic program unit with which the 940 time-sharing system is concerned is called a fork – a self-contained body of code consisting of a main program and all subroutines necessary to perform a particular process. A fork may have a maximum of 16K memory active at any time. The "forking structure" concept is analogous to the "overlay structure" concept used by many batch monitors. An overlay structure consists of segments of a program that, when loaded, will reside in the same locations in memory at different times. At any given time only a part of an overlay structure will be active. When this segment completes execution it can take action to bring in another segment of the overlay structure.

Similarly, a user can have only <u>one</u> fork active at any given time. While this fork is active it can take action to establish and transfer control to a lower fork. When the lower fork has finished executing, it can transfer control back to the parent fork. The lower fork may share pages of memory with any of the parent forks, and it may acquire more memory independently of the parent. While the lower fork is executing, the pages of the parent fork that are not being shared with the lower fork have most likely been swapped out to the RAD. However, sufficient information has been retained to establish the environment of the parent fork when the lower fork terminates.

Figure 1 illustrates a forking structure for four users (the Phantom User is discussed in Chapter 2). A user may have a total of 32K in his forking structure. The user is not charged for the memory required by the reentrant part (procedural part) of the reentrant subsystems. A maximum of 8 forks is allowed in the user's forking structure.

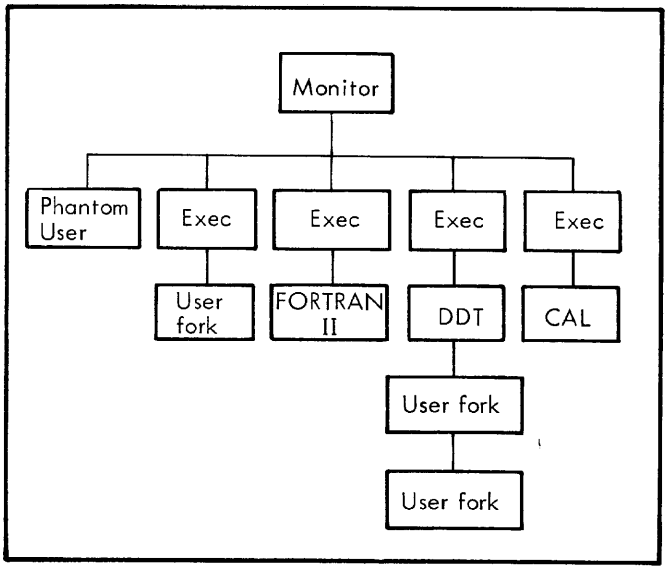


Figure 1. Typical Forking Structure

The most important aspects of the forking structure are that

1. The software deals with forks, the scheduler activates a fork, not a job or a user.
2. A user can have only one fork in his forking structure currently active. This fork is either running or waiting on one of the queues to be allotted a time slice. The rest of the forks in the user's forking structure have been dismissed and will be reactivated when the lower fork terminates.

Figure 2 is a diagram of the Monitor showing its functional parts, not the actual core layout. The Monitor pages are "locked", that is, not available for swapping. The locking is software-implemented. If a fork initiates an I/O operation, it will be dismissed to allow activation of another fork while its I/O is in progress. The page where the I/O is being performed will be locked until the I/O operation is completed, although the rest of the fork's memory may be swapped. Upon completion of the I/O, the original fork will be reactivated when the system can assign it a time slice.

The scheduler is the most important element of the 940 Monitor. It decides which fork to activate and it calls the swapper which collects from the RAD any pages the fork requires. When the swapping has been completed, the

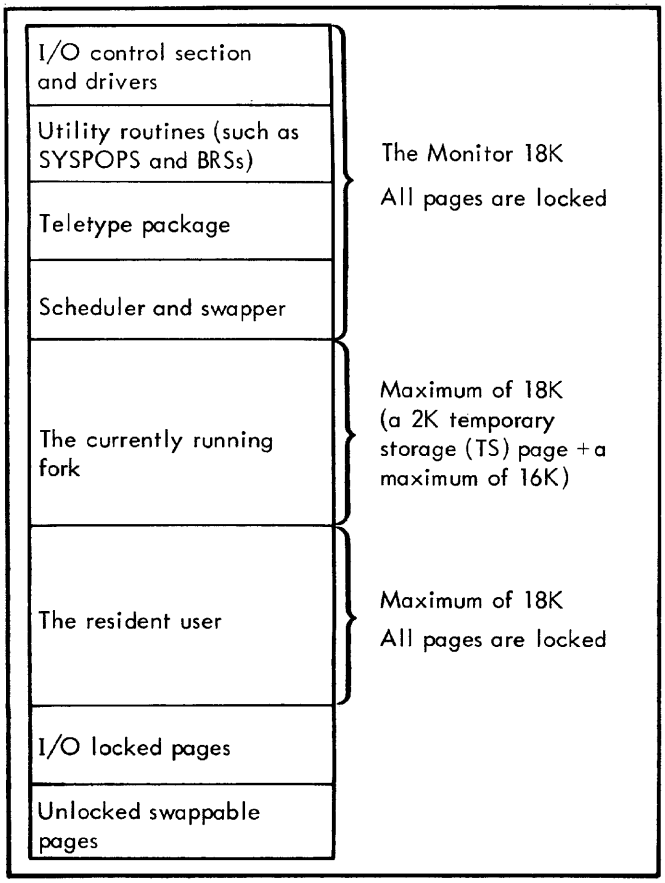


Figure 2. Concept of Locked Pages

scheduler transfers control to the fork and, by so doing, effects a transfer to the user mode. The fork will execute until a specified quantum of time has elapsed, or until the fork requests I/O and a call must be made to activate an I/O device, or until the fork takes some action that dismisses itself. At this time, control is returned to the scheduler, which finds another fork to activate.

A 64K system utilizes the "compute while swap" option. With this option, the memory a fork requires is allocated, the necessary RAD I/O operations are initiated, and the fork is placed on the swap wait queue (SWQ). This fork will remain on SWQ until all of the pages the fork requires have become core-resident.

After the fork has been placed on the SWQ, the system then activates that fork on SWQ that has all of its memory core-resident, or the resident user fork. The resident user fork is the last fork that was dismissed for quantum time overflow. Since such a fork is compute-bound, the system can be executing this fork while the other fork is being swapped into core.

When a fork becomes the resident user, the pages it requires are locked so that the system can immediately transfer to this fork when a swap has been initiated. When the resident user is activated, its pages are unlocked. The fork will execute until it is dismissed for any of the usual reasons or until a fork that is on the SWQ has all of its page core resident. Note that a fork will lose its identity as the resident user if another fork (activated from the SWQ) is dismissed for quantum overflow.

If there is no resident user (no forks have been dismissed for quantum overflow or the system has 48K of memory) the system will wait until one of the forks on SWQ has all of its pages core resident. There can be a maximum of two forks on SWQ.

When the swapper activates a fork, all of the pages belonging to the fork are marked "read-only". If the fork executes any instruction that alters memory, a trap routine will change the status of the page to "not read only". The amount of RAD I/O required is significantly reduced in this way. The swapper does not have to output a "read-only" page since a copy already exists on the RAD. The only pages in the system that are truly read-only, however, are the reentrant Executive or subsystem pages. The trap routine produces an error condition if a read-only trap occurs in one of these reentrant pages.

THE EXECUTIVE

The executive is the intermediary between the Monitor and the user. It is concerned with

- the command language through which the user controls the system from his teletype
- identification of the various users
- the specification of the limits of each user's access to the system
- control of the directory of symbolic file names, and backup storage for these files
- requests for a subsystem.

The Executive is a reentrant program that uses a Temporary Storage (TS) page, 2048 words, to accomplish reentrance. Each user is assigned a TS page when he dials onto the system. Therefore, each user actually has a maximum of 30K available memory (32K minus the TS page).

The TS page contains the I/O buffers, various constants pertaining to the user, and reserved storage for the B and X registers. Requests by the user for file I/O (other than teletype) involve a transfer of data from the device to a TS page buffer. Various I/O SYSPOPs exist that accommodate a transfer of information from the TS page buffer to the user's program area.

The access a user has to the system is defined by his status. The user's status determines what Executive commands are available to him and what SYSPOPS he may execute. Four levels of status are available:

1. Operator
2. System
3. Subsystem
4. User (the Executive commands available to the user are explained in the "Terminal Users Guide" 90-11-18A)

Any of the above may be granted peripheral status or subsystem class status, or both. With peripheral status the user may access certain peripheral devices through the medium of the Executive command structure. Subsystem class status permits the user the use of TAP and DDT.

SUBSYSTEMS

Subsystems are major processors such as FORTRAN II, CAL, QED, etc. that perform specialized functions. These subsystems are programs that are permanently connected to the main system. Each subsystem is called by name through the Executive, and the Executive then establishes a lower fork for the subsystem.

The processors implemented as 940 subsystems have such a high rate or usage that they have been written as reentrant programs, enabling many users to share the same processor simultaneously.

Programming reference information on the major subsystems is contained in individual manuals listed under related publications in front of this manual.

2. SCHEDULER

PROGRAM ACTIVATION TABLE

Since a time-sharing environment involves the dynamic swapping in and out of user forks, tables must be maintained that enable the system to establish the program the fork had before it was dismissed. Each fork has a PAC table (PACT) associated with it. The format of the PACT is shown in Figure 3.

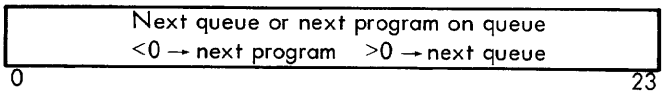
Note that the PACT contains locations for saving the program counter, P, and the contents of the A register. The B and X registers are saved in the TS block. The PACT also contains two of the three pseudo-relabeling registers for the fork. The third, which specifies the TS block, is kept in one of the job tables (RL3). Pseudo-relabeling is discussed in detail later in this chapter. The word PTEST determines the conditions under which the fork should be reactivated if it is not currently running. The panic table address in PTAB and the two pointers called PFORK and PDOWN are discussed under "Panic Conditions".

Once the fork has become active, it can be in one of five states

1. currently running;
2. on the scheduled queues awaiting allocation of a time-slice;
3. on the swap wait queue
4. dismissed (in limbo); the fork executed a BRS 9 and dismissed itself in order to activate a lower fork. The fork can be reactivated because the linking is intact in the PFORK and PDOWN entries in PACT.
5. terminated; e.g., the fork ran to completion, executed an illegal instruction, the user logged-off, etc. Once terminated, the PACT is returned to the free PACT list.

The function of the various PAC table entries is shown in Figure 3. Detailed explanations for each word of the table entries are as follows

PNEXT
(Word 0)



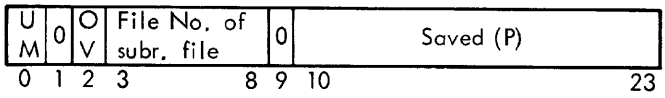
Used in the queue chaining scheme. Three possibilities exist:

a. If the fork defined by this PAC table is on a queue and PNEXT is >0, this fork is the last on a particular queue, and PNEXT points to the next queue.

b. If the fork is on a queue and PNEXT is < 0, it is a pointer to the next PAC table on the queue.

c. If the fork defined by this PAC table is not on a scheduled queue, or if this PAC table is not in use, PNEXT is not meaningful.

PL
(Word 1)



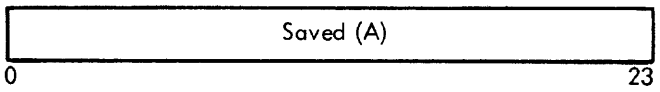
UM User Mode. This bit is set if the fork executes in user mode. This is the case for all forks except the phantom user.

OV stored overflow – status of the hardware overflow indicator is stored into this bit when the fork is dismissed. The overflow status can then be restored when the fork is activated.

Saved(P) the virtual (see chapter on "Mapping" in 940 Reference Manual) address of the cell to which control must be transferred when the fork is reactivated. The following coding is executed when a fork is activated.

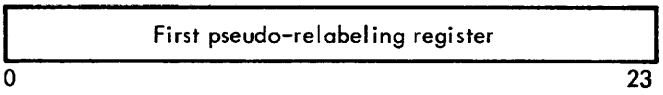
```
        .
        .
        .
LDA     PL, 2
STA     0
        .
        .
        .
BRU*    0       will transfer to user mode if
                UM=1
```

PA
(Word 2)

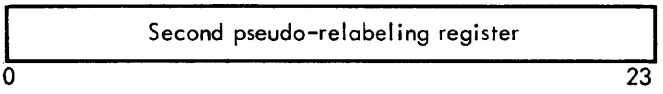


Saved (A) the contents of this cell will be placed in A when the fork is reactivated. The cell is initialized from the A register if the fork was dismissed or from the panic table if this is a new fork. "PB" and "PX" are stored in the user TS page, and are indexed by fork number (bits 18 through 20 of PIM).

RL1
(Word 3)

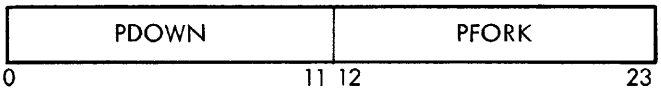


RL2
(Word 4)



RL1 and RL2 these are the pseudo-relabeling registers for this fork. Each six-bit byte points to an entry in the SMT or user PMT which in turns points to a real page or RAD address.

PPTR
(Word 5)



This word can have two configurations

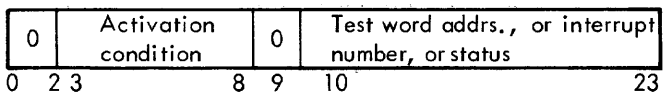
a. If PAC table is not is use, bits 10 through 23 contain a pointer to the next free PAC table.

b. If PAC table is in use, PPTR links this fork within the hierarchy of this user's forking structure.

   1. PDOWN contains the PACT pointer to the next lower fork.

   2. PFORK contains the PACT pointer to the parent fork.

PTEST
(Word 6)



Activation conditions are

(Bits 3-8)

0 Word greater than 0

1 Word less than or equal to 0

(Bits 3-8)

2 Word greater than or equal to 0

3 Word less than or equal to teletype early warning

4 Special test. The address points to a special activation test routine. Applies to Phantom User.

5 Interrupt occurred. The address contains the number of the interrupt which occurred.

6 Word less than or equal to real-time clock.

7 Special address =

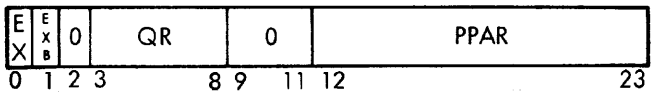
   0 dead
   1 running
   2 BRS 31 (see Chapter 6 re BRSs)
   3 BRS 106
   4 Executive BRS
   5 BRS 109
   6 BRS 9 (User Program)

10 Do not activate

11 Bit 1 of word = 0 (buffer ready)

12 Word less than 0

PQU
(Word 7)



Program Quantum Word:

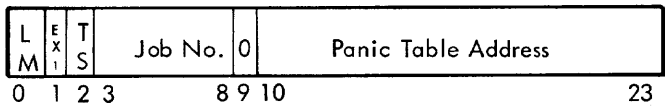
EX fork has system status.

EXB fork was created by an Executive BRS.

QR contains the long quantum remaining for the fork (measured in 60 HZ clock ticks).

PPAR PACPTR of a parallel fork. Parallel forks are not implemented.

PTAB
(Word 8)



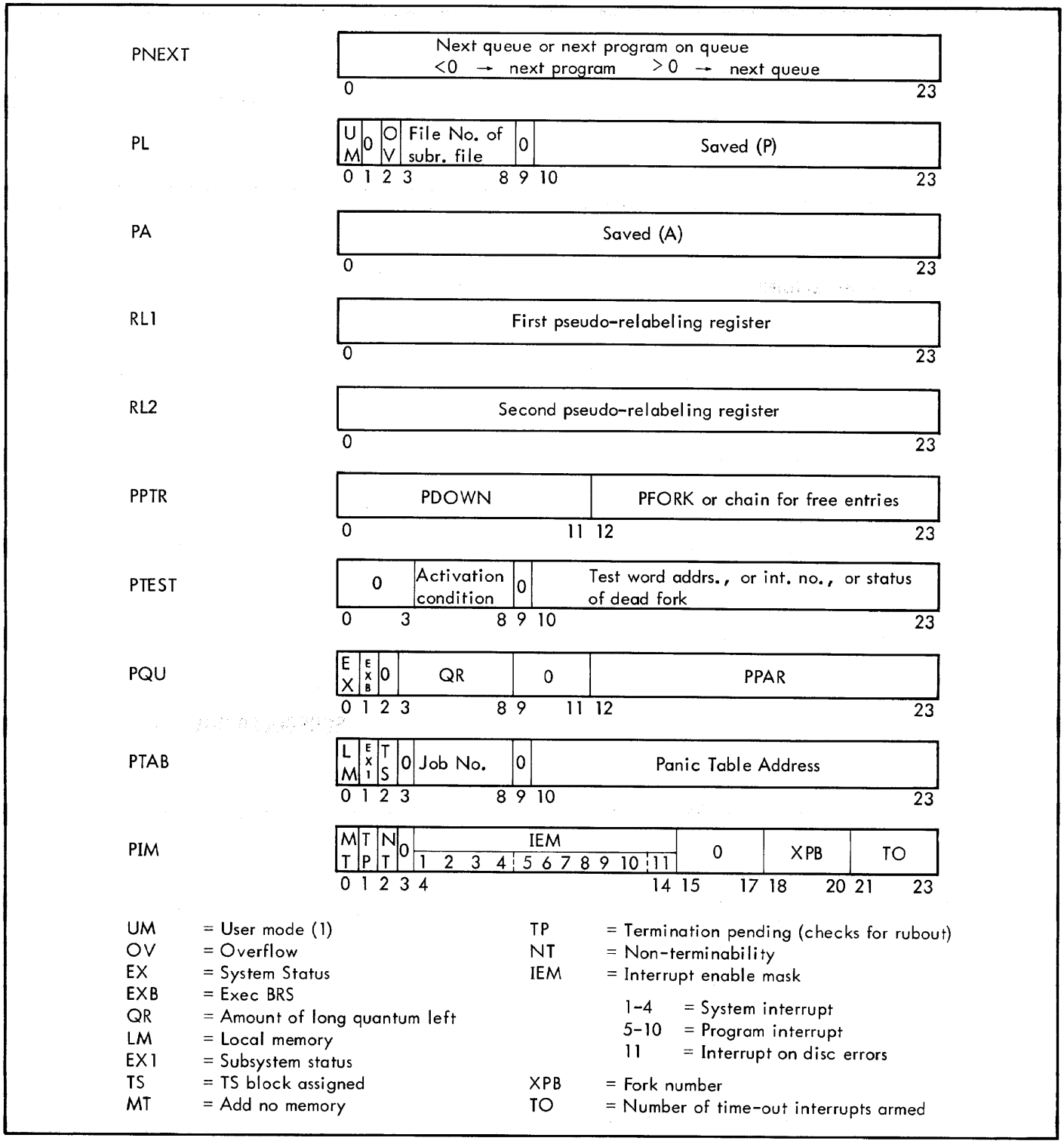
LM fork is local memory. This means, essentially, that this fork will obtain memory independent of its ancestors which means, in turn, that its ancesters are protected from this program. See "Memory Acquisition."

EX1 fork has subsystem status.

TS A TS page has been assigned to the user in whose forking structure this PAC table appears.

Job number has nothing to do with user or TT number but is assigned arbitrarily when user logs on.

Panic Table Address The virtual address of a 7-word panic table to be filled in when this fork terminates. It is usually within memory "owned" by the controlling fork.

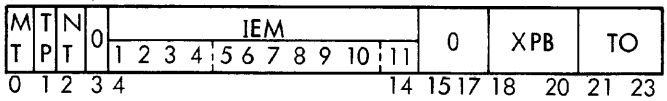


UM = User mode (1)
OV = Overflow
EX = System Status
EXB = Exec BRS
QR = Amount of long quantum left
LM = Local memory
EX1 = Subsystem status
TS = TS block assigned
MT = Add no memory

TP = Termination pending (checks for rubout)
NT = Non-terminability
IEM = Interrupt enable mask
1-4 = System interrupt
5-10 = Program interrupt
11 = Interrupt on disc errors
XPB = Fork number
TO = Number of time-out interrupts armed

Figure 3. PAC Table – One Per Fork

PIM
(Word 9)



Program Interrupt Mask:

MT fork may acquire no memory. Any attempt to do so will result in the fork being terminated on memory panic. See "Memory Acquisition".

TP termination pending; set by the Phantom User if an escape occurs while NT is set. If the fork executes a BRS 47 (turn escape on), both the TP and NT bits are reset. If the TP bit was set, the fork is dismissed onto QQE. The task for processing the escapes is still on the Phantom User and can now be processed. If TP bit was not set, then the fork continues execution.

NT non-terminability; set by the BRS 46 (turn escape off). The escape task is still placed on the

phantom user queue. However, phantom user takes no action regarding termination of this fork.

IEM software interrupt mask. See "Software Interrupts".

XPB the fork number; used to index PB and PX in the TS page. This number may range between 0 and 7.

TO count of the number of time-out interrupts that the fork is currently using. A fork may have a maximum of 3 time-out interrupts armed.

TIME-SLICING

In order to implement time-slicing, the system defines a long and short quantum. These parameters are defined in clock-ticks and can be easily modified at system-generation time. All times in the discussion are measured in periods of the 60-cycle computer clock.

Two dedicated cells in the system are used to contain the amount of time remaining for the currently active fork. These are TIME (short quantum) and TTIME (long quantum). These cells are initialized each time the scheduler activates a fork.

Once a fork is activated it will be allowed to execute until its short quantum has expired. The fork may take action that will cause itself to be dismissed (such as an I/O request) before the short quantum has elapsed. However, if the fork is compute bound, it will execute for at least one short quantum.

Once a fork has consumed a short quantum, it can be dismissed if any of the forks previously dismissed for I/O (i.e., any of the forks on QTI or QIO) are ready to be activated. If this occurs, the currently active fork is dismissed for short quantum overflow.

The clock interrupt routine determines if any I/O bound forks are ready for activation by checking a word called ACTR. ACTR is set to -1 when the scheduler determines that no I/O bound fork is ready for activation. ACTR is incremented by an I/O interrupt routine whenever any I/O bound fork is ready to be activated.

If the short quantum has expired, but no I/O bound forks are ready to be activated, the currently active fork is permitted to execute until it consumes a long quantum. Once a fork has expired its long quantum, it is unconditionally dismissed. This permits another computer bound fork to be assigned a time slice.

The long quantum is cummulative; that is, when a fork is dismissed, the long quantum remaining is recorded into the PAC table. Whenever this fork is activated, TTIME is set to the previously recorded value. If a fork is dismissed for long quantum overflow, it is assigned a new long quantum and this value is stored in the PAC table.

In order to activate a fork, the system must allocate a considerable amount of its resources. These resources are in the form of core memory, RAD operations, etc. In order to make this allocation worthwhile, the system always guarantees a fork a full short quantum each time it is activated. An exception to this occurs if a fork was activated from resident user status. Since this fork has already executed for at least a short quantum and since another fork is in the process of being swapped into core, the resident user is not guaranteed a short quantum of execution time. This fork will be dismissed for simulated long quantum overflow as soon as the swap in progress has completed. This prevents the resident user from executing for a maximum of a long quantum (in the event that no forks dismissed for I/O are ready to be activated) and enhances the probability that the system will have a resident user to activate when the next swap is initiated.

The clock interrupt routine forces forks to be dismissed for quantum time out by arming the Monitor-to-user transition trap. (See the SDS 940 Computer Reference Manual, Publication No. 90 06 40, for a discussion of the Monitor-to-user transition trap). If the fork was executing in user mode, this trap will occur when the clock interrupt is cleared. The trap routine then determines if the short quantum has expired and causes the fork to be dismissed to the appropriate queue. Since the Monitor is not reentrant, no attempt is made to dismiss a fork that is executing in Monitor mode. Therefore, no user fork is dismissed while executing a SYSPOP. The clock routine will have armed the transition trap when either quantum has expired, however, the trap will not occur until the Monitor exits from the SYSPOP. Note that this mechanism also prevents the system from dismissing itself or the phantom user for quantum overflow.

SCHEDULED QUEUES

When a fork has been dismissed or is awaiting activation, its PAC table will be on one of the four scheduled queues. The queues are listed in order of priority.

QTI Forks dismissed for teletype input/output.

QIO Forks dismissed for other I/O, forks that have just been initiated by a BRS 9, and forks activated by escapes, program panic, etc.

QSQ Forks dismissed for short-quantum overflow.

QQE Forks dismissed for long-quantum overflow.

The information in any PAC table can be retrieved by using the PAC pointer associated with the PAC table. The PAC pointer is a negative index that allows retrieval of any entry in a fork's PAC table. Assume that PACPTR contains the index to a particular PAC table in the array of PAC tables. Then

```
LDX     PACPTR
LDA     PA,2        Fetches the PA word
LDA     PQU,2       Fetches the PQU word
```

Location PACPTR always contains the PAC pointer of the fork the system is currently running. PA, PQU, PNEXT, etc., are defined in relation to the end of the PAC table array. Currently the system is dimensioned for 144 PAC tables.

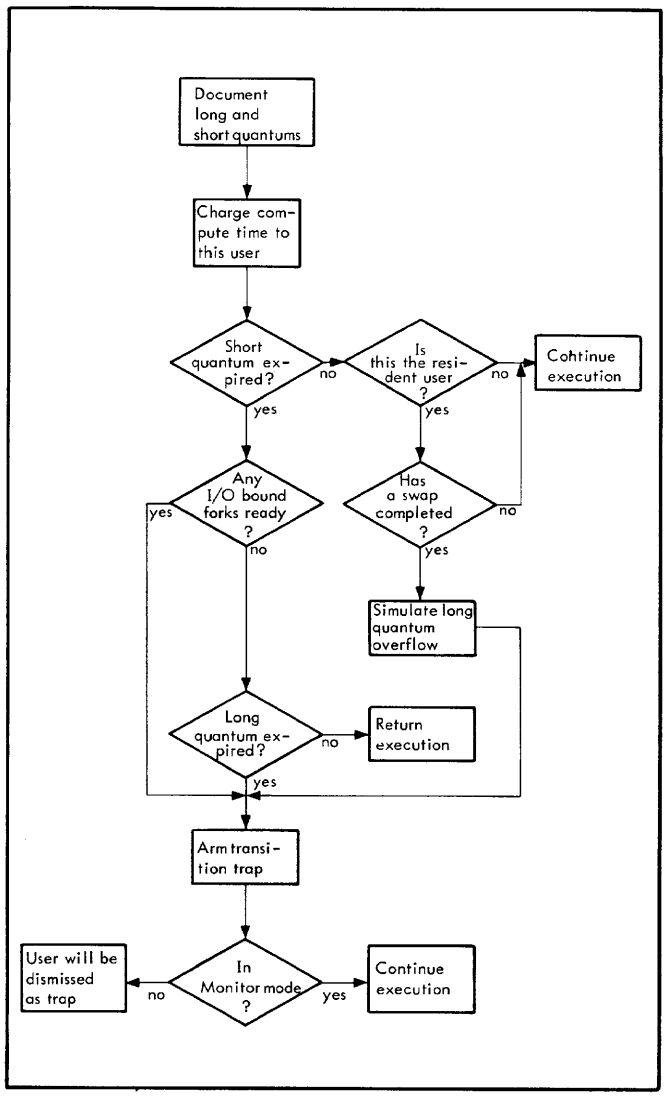


Figure 4. Simplified Clock Interrupt Routine

When a PAC table is on the scheduled queues, the PAC pointer for this fork in in the PNEXT word of the PAC table immediately before it on the queue. The scheduled queues are each 3 entries long and have the following format:

Word 0

PAC pointer to the first PAC table on this queue, or the address of the next queue
0 23

Word 2

PAC pointer to the last PAC table on this queue, or a pointer to this queue
0 23

Word 3

Address of the next queue
0 23

The scheduler begins searching a queue by fetching Word 0 of the queue. If this word contains a positive number, there are no entries on this queue since a PAC pointer is always negative. However, if Word 0 does contain a PAC pointer, it can easily obtain the PTEST word associated with the PAC table and determine if activation is possible. If activation is not possible, the PNEXT word can be retrieved (since each PAC table on a queue points to the next one on the queue) and the next PAC table can be checked for activation. If an entire queue has been searched and no activatible fork is found, the scheduler will then search the next lower priority queue.

For example, assume PAC tables numbered:

1, 3, 10	are on QTI
2, 8	are on QIO
15, 4	are on QSQ
None	are on QQE

The numbers are assigned for convenience in the following diagram (Figure 5). Note that if a queue has no entries, the first word contains a positive number which is the address of the next queue.

If there are no forks on the scheduled queues which are ready to be activated, the system searches the swap wait queue for a fork that has all of its pages core resident. If no fork on the swap wait queue is ready, the system activates the resident user, if any. If no activatable fork is found, the system returns to search the beginning of the scheduled queues and repeats the same process.

PHANTOM USER

There are certain operations that the Monitor must perform for the users on the system. Some of these tasks are

1. Processing of a teletype ON interrupt
2. Processing of escapes
3. Processing of the software time-out interrupts
4. Processing of teletype OFF interrupts
5. Typing of certain error messages
6. Testing of I/O devices for ready.

Consider the processing of a teletype ON interrupt. This involves the assignment of a job number, initialization of various tables, assignment and initialization of the PAC table for the Executive fork, etc. It is not feasible to perform all of these functions in the teletype ON interrupt

routine. Therefore, the interrupt routine should simply honor the interrupt and notify the Monitor to finish processing the task.

Since many of the tasks are initiated by an interrupt, the Monitor must have a task queue where a function that it is to perform can be added. As the system performs a task it can remove it from the queue. However, in order to process these tasks, the Monitor must be allotted a time slice.

It is the scheduler that decides which fork on the system to activate. What whould be given priority - the processing of a teletype OFF interrupt or the activation of a fork on QTI that was dismissed for teletype input and is now ready? The problem of priority assignment for all users on the system is handled by establishing the scheduled queues. Therefore, the Monitor can conveniently be assigned a time slice if it has fork on the scheduled queues. When this fork is activated it could check the task queue and perform the various functions. This fork is referred to as the phantom user. The phantom user runs in Monitor mode and requires no memory of its own. Its memory and relabeling are those of the Monitor. The Monitor task queue is called the PUCT table.

The PAC table for the phantom user is set up and put on QTI when the system is first initialized. The PTEST word contains an immediate activation condition. When the scheduler activates the phantom user, if first checks a word called PUCTR (Phantom User Counter). PUCTR is incremented by every routine that adds a task to the PUCT queue (the Phantom User Task Table). If PUCTR=0 (no tasks), the phantom user dismisses itself onto QTI.

If PUCTR is greater than 0, the phantom user begins to search the PUCT table for a task to perform. Each task the phantom user can perform is given a code number. (See bits 3-8 of the second word of a PUCT entry.) Each time an attempt is made to process a task the following occurs:

1. A check is made to determine if the task is ready to be processed. For instance, the phantom user can do nothing about a time-out interrupt until the appropriate amount of time has elapsed. If the task cannot be processed, the phantom user continues to scan PUCT until it reaches the end of the queue.

2. If the task can be processed, PUCTR is decremented. The phantom user then branches to the appropriate routine for performing the task. All of the routines return to where the phantom user can begin to scan the PUCT table.

The phantom user will execute until either all of the tasks have been performed (PUCTR=0), or until the tasks that remain on the queue cannot be processed at this time. When this occurs, the phantom user dismisses itself onto QTI with an activation code of 4.

Whenever the phantom user is dismissed, location ACTPU is set negative to indicate that all of the tasks that were ready to be processed have been completed. Any routine that causes a task on PUCT to be ready for processing will increment ACTPU. With a phantom user activation code of 4, the scheduler will activate the phantom user if ACTPU is non-negative. The phantom user will be unconditionally activated when at least 3 seconds have elapsed. Figure 6 shows the format for the PUCT table entries. A routine named EPU adds an entry to the PUCT list.

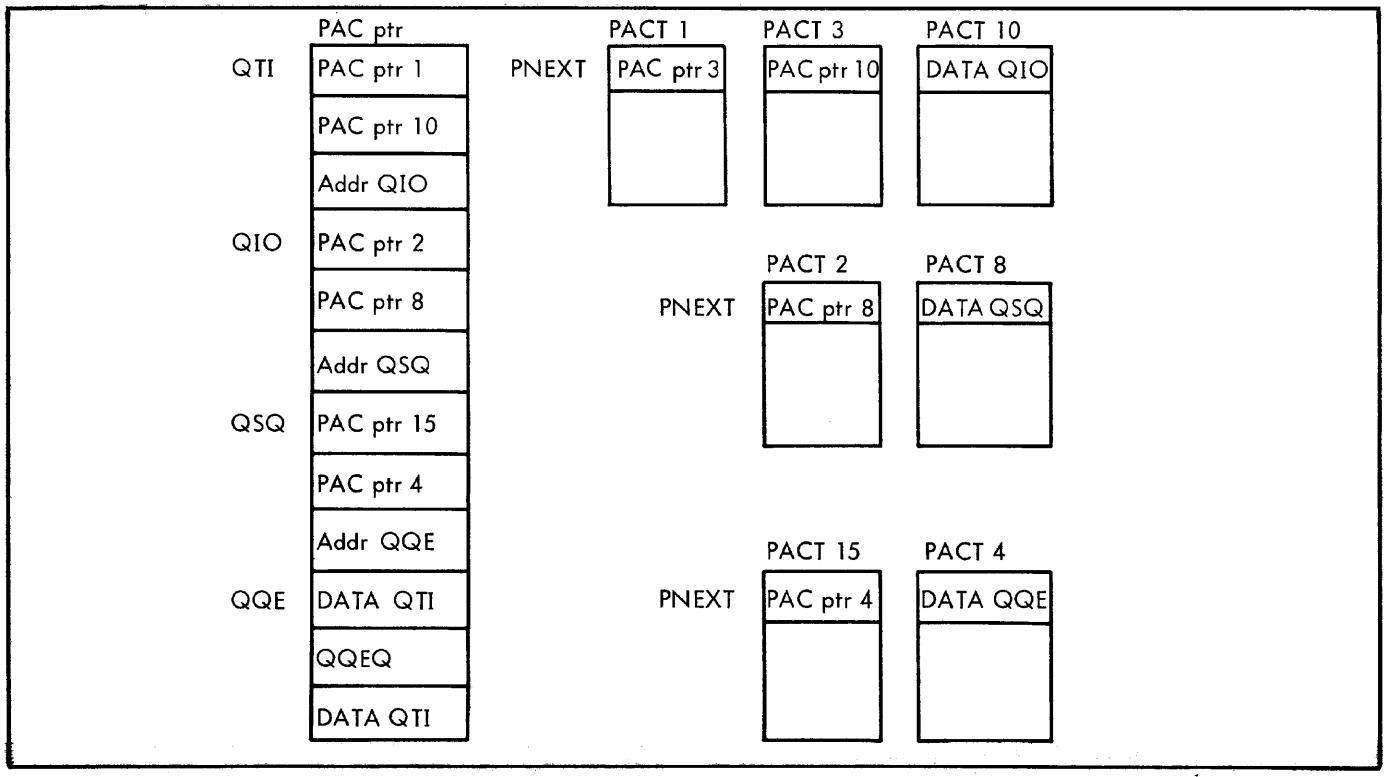


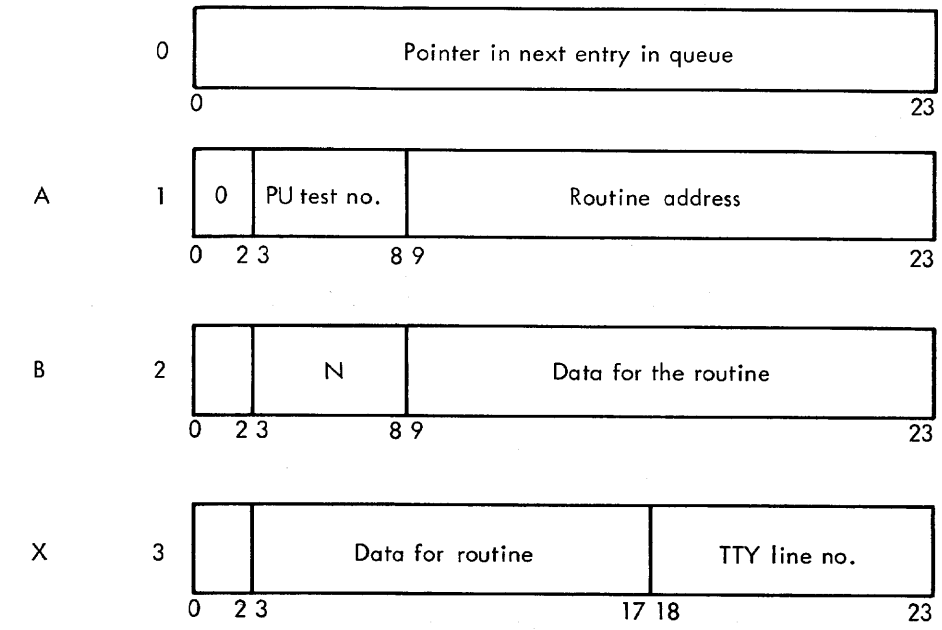
Figure 5. Fork Searching Scheme

EPU accepts its input parameters through the A, B, and X registers.

SUMMARY OF SCHEDULER FUNCTIONS

Figure 7 is a flow diagram showing the functions of the scheduler. The scheduler searches the scheduled queues for an activatable fork. When it finds one it calls the swapper. The input to the swapper is the pseudo-relabeling registers and a flag called MGTS5. MGTS5 is set to 0 to indicate that the swapper is to build up an appropriate list of RAD commands and initiate the RAD I/O, but not wait until the RAD I/O is completed. The swapper returns to the scheduler the real (hardware) relabeling registers. Since MGTS5=0, the swapper does not at this time set the hardware relabeling. If the swapper is unable to allocate real pages to this fork because sufficient memory is not available at this time (i.e., a sufficient number of pages are not unlocked), it takes an abnormal exit. The scheduler will then attempt to find another fork that the swapper is able to allocate.

The scheduler then puts the fork onto the swap queue (SWQ). SWQ consists of five tables. These contain the PAC pointer of the fork (SWQPAC), the number of pages (negative count) that still remain to be read from the RAD (SWQPGC), and the three real relabeling registers, (SWQRL1, SWQRL2, and SWQRL3). The RAD logic will increment the page count each time a page is brought in. The schedule determines if a fork on SWQ is ready to be activated by checking the page count. When the count is zero, the fork is in core and ready to be run.



PU Test No.:

0 Continue disc I/O

1 Escape or teletype off interrupt

2 Magnetic tape ready

3 BLK31=0

4 Card reader ready

5 Start PU fork

6 Data set time-out ("teletype on interrupt")

7 Program interrupt time-out

8 Line printer ready

9 Card punch ready

10 Dead entry - the phantom user will ignore this task and remove it from the PUCT queue.

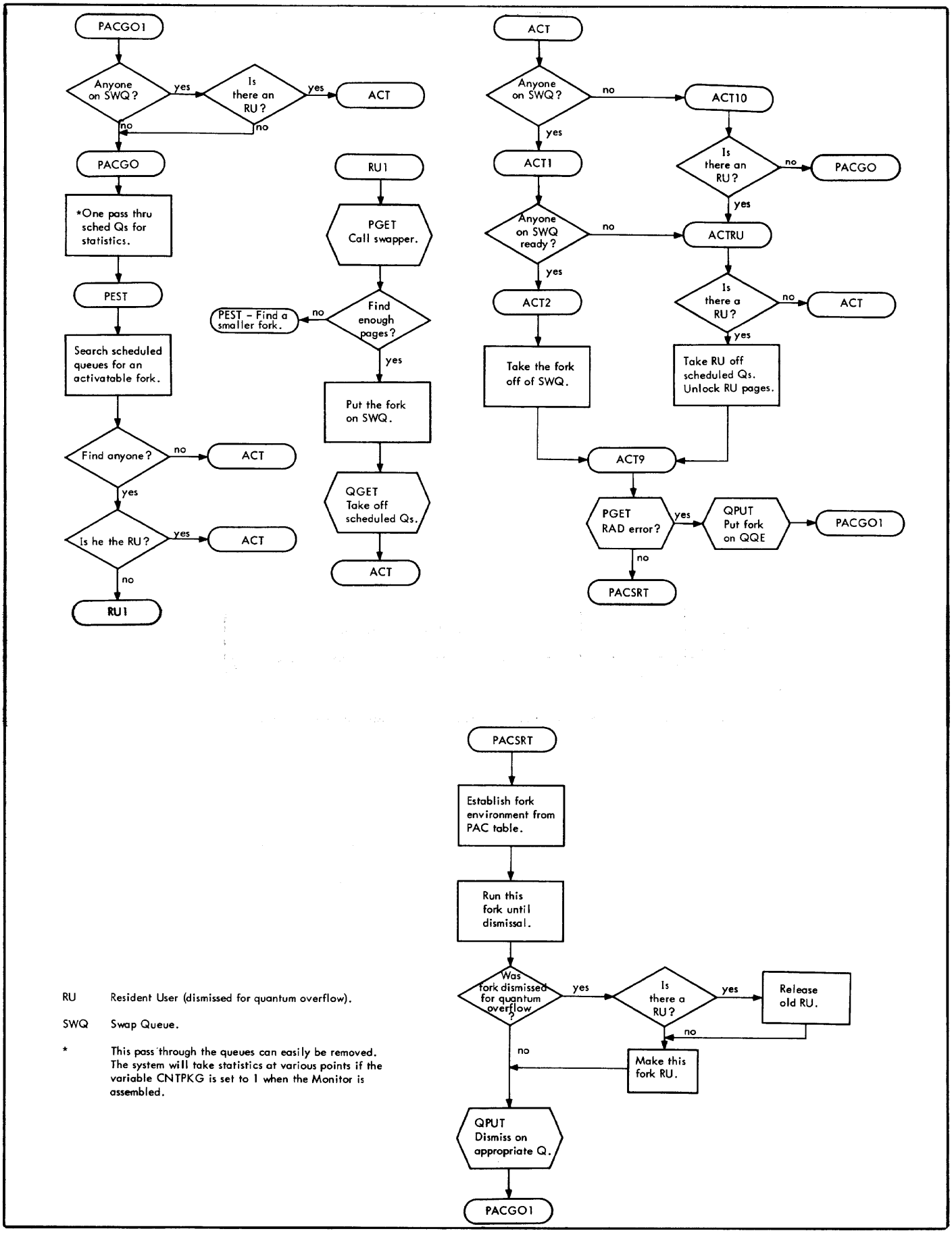
N is meaningful for the following tasks:

N = 1 on interrupt

2 off interrupt

3 escape

Figure 6. Phantom User Queue Entry



RU Resident User (dismissed for quantum overflow).

SWQ Swap Queue.

* This pass through the queues can easily be removed. The system will take statistics at various points if the variable CNTPKG is set to 1 when the Monitor is assembled.

Figure 7. Overview of Scheduler

The SWQ is scanned at ACT for forks that are ready to run. If none is found, then the resident user (a user dismissed for quantum overflow) is run. If there is no resident user and no fork has yet completed its swap, then the system continues to scan SWQ. If a fork has all the required pages, the entry is removed from SWQ and the swapper is again called. This time MGTS5=-1. The swapper verifies that all of the pages are in, i.e., no RAD errors have occurred, and sets the hardware relabeling. The fork is then activated.

If a RAD error did occur, the fork is dismissed on QQE with an immediate activation condition. The system will then search the scheduled queues for another activatable fork.

At PACQE, any fork which is dismissed for quantum overflow will have its memory locked in core and its real relabeling saved in RURRL1, RURRL2, and RURRL3. The memory of the previous resident user, if any, is released at this time. Whenever there are no forks on SWQ that are ready to run, the resident user is activated, thus using CPU time which would otherwise be wasted. Whenever the resident user is activated, he loses his identity as the resident user and will be the resident user again only if he is dismissed for quantum overflow.

There can be a maximum of two forks on SWQ. This is accomplished by the coding at PACGO1. A fork will be added to SWQ only if there is no resident user. If there is no resident user, the coding at ACT will continue to search SWQ until one of the forks can be activated and thus removed from SWQ. If the fork that was placed on SWQ first is still not finished swapping, and the second fork has all of its memory (often the case with the phantom user), then the second fork will be activated. The system does not attempt to put more than two forks onto the swap wait queue due to memory limitations. The two forks could account for a maximum of 36K of core (2K for the TS page and 16K for the fork). Some of the memory may be locked into core due to prior I/O requests. The system should have pages available to accomodate requests by the running fork for relabeling changes, memory acquisitions, and the processing of panics.

3. FORKING STRUCTURE

JOBS

The system refers to a currently active user as a "job". A job number is associated with every forking structure. If the system is dimensioned to handle 32 users, the job numbers range from 0 to 32, with the phantom user always assigned job number 0. The job numbers are assigned in a somewhat random fashion. As a user logs on the system, he is assigned a job number that is currently not in use. When he logs off, his job number is returned to the free job list. The available job numbers are chained in the TTNO array.

The job number is used to index several tables. These tables contain information that pertains to the job, rather than an individual fork, such as the teletype number associated with the user, CPU time, etc. The job number is stored into the PAC table for every fork in a user's structure. The job tables are shown in Figure 9.

PMT AND SMT TABLES

The PMT (Private Memory Table) preserves the environment of the user's memory. The table provides a real page number and a RAD address and indicates whether the page is in core or on the RAD. Each user is assigned a PMT. The table is 20B entries in length which represents the 16 pages or 30K and one TS page that a user can acquire. The PMT table that a user is assigned is a function of his job number.

There is one SMT (Shared Memory Table) in the system. The SMT is similar in format to the PMT. The SMT provides information about the Monitor, Executive, and reentrant subsystem pages. This table contains 60B entries. Each entry in the table is unique to a particular subsystem. The entries are assigned when the system is assembled. (See Figure 8).

PSEUDO-RELABELING

When a fork is dismissed, it would be meaingless to save the contents of the hardware relabeling registers, since memory is being changed dynamically. Therefore, each fork has a pair of "pseudo-relabeling" registers associated with it. Each pseudo-relabeling register consists of four bytes. Each byte points to a PMT/SMT entry. Using the PMT/SMT tables, any necessary swapping can be initiated and the hardware relabeling can be constructed.

Psuedo-relabeling bytes with a value of 0 through 57B point to SMT entries. Bytes having a value of 60B through 77B point to a PMT entry.

The Executive always uses PMT entry 60B for the TS page. All of the reentrant subsystems use at least one page of the user's memory for scratch storage.

As a fork acquires new memory (e.g., by executing a "store A" instruction referencing a page that the fork does not have) a page is acquired and a RAD address is supplied. This process can continue until the user has acquired all 32K.

In this way, the PMT reflects all the memory that a user has acquired. The pseudo-relabeling registers indicate which PMT/SMT entries (i.e., what memory) are necessary in order to activate a particular fork.

MEMORY ACQUISITION

A fork may have a maximum of 16K. When the fork is activated it may have less than 16K and then acquire more memory as needed while it is executing. The following is a partial list of how not to acquire more memory).

- By falling through a page (to a page which is not in the pseudo-relabeling) to get the next instruction.
- By going indirect via some address which is out of bounds (i. e., LDA *100 where 100 is out of bounds).
- By doing an EXU to an address which is out of bounds.
- By doing a POP if page 0 is not in the fork's relabeling.
- By an unconditional branch to an out of bounds address.
- By doing a BRS 44 and requesting a byte that points to a PMT entry that has not been acquired.
- By doing a BRS 9 and requesting pseudo-relabeling bytes that are not meaningful.

The correct way to acquire more memory is to execute any instruction (such as LDA, STA, ADD, MIN, etc.) that directly references a location in a page that has not been acquired. This includes the initial loading of a program or an I/O request into a page which has not been acquired.

If the fork addresses a block of memory that is not assigned to it, a check is made to determine whether the machine size specified by the user has been exceeded. If so, a memory panic is generated. If the fork is fixed memory, a memory panic is also generated. Otherwise, a new block is assigned to the fork so that the illegal address becomes legal. For a local memory fork, a new block is always assigned. Otherwise, the following algorithm is used:

The number, n, of the relabeling byte for the block addressed by the instruction causing the memory trap is determined. A scan is made upwards through the fork structure to (and including) the first local memory fork. If all the forks encountered during this scan have Rn (the nth relabeling byte) equal to 0, a new entry is created in PMT for a new block of user memory. The address of this entry is put into Rn for all the forks encountered during the scan.

If a fork with nonzero Rn is encountered its Rn is transmitted down to all the forks between it and the fork causing the trap. If any fixed memory fork is encountered before a nonzero Rn is found, a memory panic occurs.

This arrangement permits a fork to be started with less memory than its controlling fork in order to minimize the amount of swapping required during its execution. If the fork later proves to require more memory, it can be reassigned the memory of the controlling fork in a natural way. It is, of course, possible to use this machinery in other ways, for instance, to permit the user to acquire more than 16K of memory and to run different forks with nonoverlapping or almost nonoverlapping memory.

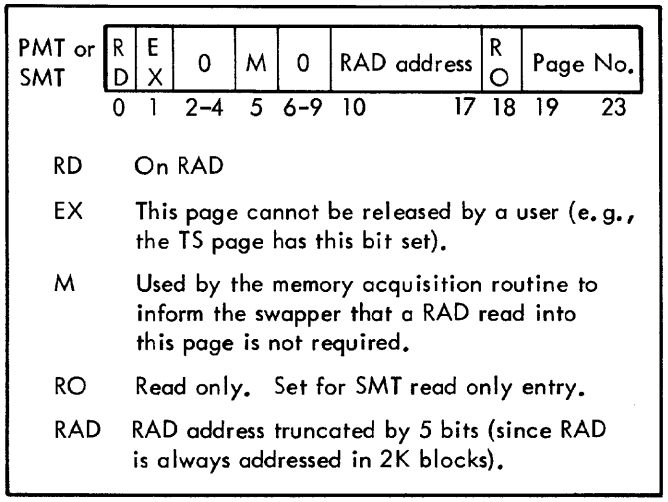


Figure 8. Private Memory Table Entry or Shared Memory Table Entry

The acquisition of memory is performed in a routine named MGET. The RAD bit map is checked and an available page is found. The RAD address is stored into the next available PMT entry and the swapper is called. The swapper then finds an available core page. MGET calls the swapper with MGTS5=-1. The swapper waits until any necessary RAD I/O has been completed and sets the new hardware relabeling. See Chapter 4 for more information on memory allocation.

CHANGING RELABELING

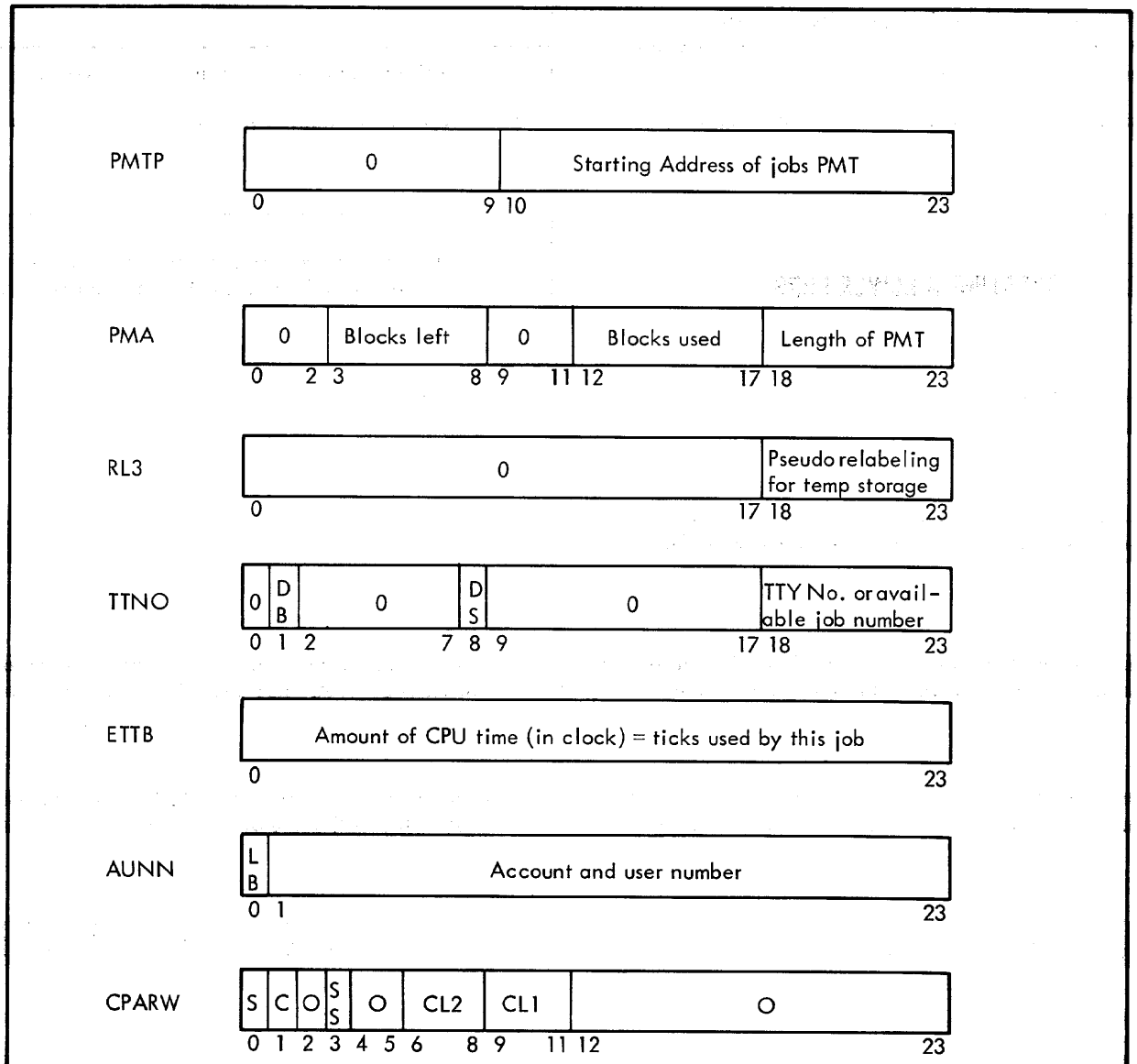
Several BRSs are available to the user to allow him to manipulate his pseudo-relabeling.

BRS 43 Returns the pseudo-relabeling of the calling fork in A and B.

BRS 44 Sets the pseudo-relabeling with the contents of A and B. There are several restrictions associated with this BRS:

1. The user cannot relabel over a system page unless he has the proper status.
2. The user cannot specify a pseudo-relabeling byte that points to a PMT entry that he has not acquired.

When a fork is activated, all of its pseudo-relabeling bytes are satisfied. Depending on the execution path, the fork may not actually need all of the pages. The fork can release a page by replacing the desired byte with 0. This will reduce the amount of swapping necessary each time the fork is activated. The BRS 44 does not remove the entry from the PMT. Therefore, the page can be retrieved by executing the BRS 44 and specifying the byte.



PMT	Private Memory Table
TTNO	Contains teletype associated with job if the job number is assigned; otherwise, contains a chained list of the free job numbers
DB	Disc busy for BRS BE+1 or BE+2
DS	Disc error
LB	User Receiving Broadcast Letter

CPARW	Status of user
S	System
C	Control
O	Operator
SS	Subsystem
CL2	Class 2 subsystem status (not used)
CL1	Class 1 subsystem status: bit 9 = TAP, bit 10 = DDT

Figure 9. Job Tables

For example, assume a fork consists of five pages. All of the error routines are in one page. This page can be relabeled out when the fork begins execution. When an error occurs, the page can be relabeled in and a branch made to the appropriate error routine.

It is possible for one fork to acquire all 30K of memory. However, it must use the BRS 44 to relabel some of the pages out, since a fork can have only 16K of memory active at any one time.

CREATING A LOWER FORK

The forking structure consists of up to 8 forks including the Executive fork. The forking structure provides the following advantages:

- Swapping time can be significantly reduced by segmenting a program.
- It permits centralized control. The user can activate one of the subsystems, return to the Executive to have various functions performed, return again to the subsystem, call another subsystem, etc.

BRS 9 will initiate and activate a lower fork, taking its argument from the A register. The first six bits of the A register specify various attributes the lower fork should have while the remaining bits specify the address of a 7-word panic table.

The panic table allows the parent fork to transfer various parameters to the lower fork. When the lower fork terminates, information is returned to the parent via this table. A panic table must not overlap a page boundary or be used for more than one dependent fork.

Table 1. Panic Table

Word	Contents
0	Program counter
1	A register
2	B register
3	X register
4	First relabeling register
5	Second relabeling register
6	Status

The status word is set by the system and may be:

-2 Dismissed for input/output
-1 Running
0 Dismissed on escape or BRS 10
1 Dismissed on illegal instruction panic
3 Dismissed on memory panic

Table 2. Significance of Bits in A Register

Bit	Significance
0	Make fork system if current fork is system
1	Set fork relabeling from panic table. Otherwise, use current relabeling.
2	Propagate escape assignment to fork (see BRS90).
3	Make fork fixed memory. It is not allowed to obtain any more memory than it is started with.
4	Make fork local memory. New memory will be assigned to it independently of the controlling fork.
5	Make fork subsystem status if current fork is subsystem.

BRS 9 causes the following to occur:

1. A PACT is obtained and initialized. The PA and PL words are set from the panic table entries. Various other bits in the PAC table are set from the information supplied in A. RL1 and RL2 are set up from the contents of the panic table or from the relabeling registers of the currently running fork. Linkage to the parent fork is established by setting the PFORK and panic table address entries. The PDOWN parameter is set in the PAC table of the parent fork.
2. A fork number is obtained. A job may have a maximum of 8 forks, including the Executive fork. The values for B and X that are supplied in the panic table are stored into the TS page indexed by fork number.
3. The supplied pseudo-relabeling is checked for validity. The bytes must point to PMT entries that have been acquired, a user fork cannot relabel in system pages, etc.
4. A word called TTYASG (indexed by teletype number) contains the PAC pointer of the fork that is to be terminated when an escape occurs. The fork pointed to by TTYASG and all lower forks will then be terminated. BRS 9 will set TTYASG to the PACPTR of the fork it is creating unless bit 2 of A is 0.
5. The lower fork (the one being created by BRS 9) is put on QIO with an immediate activation condition.
6. The parent fork is now dismissed (i. e., placed "in limbo" which implies that the fork is not on a scheduled queue) with an activation condition of 7 @ 6. The parent fork will be reactivated when the lower fork panics. If the parent fork has subsystem status, it is not dismissed and continues execution at the instruction following the BRS 9.

The parent fork and lower forks may interact in the following three ways:

1. If the parent fork is not dismissed by BRS 9:

   BRS 30 reads the current status of a lower fork into the panic table. It does not influence the operation of the fork in any way.

   BRS 31 causes the controlling fork to be dismissed until the lower fork causes a panic. When it does, the controlling fork is reactivated at the instruction following BRS 31, and the panic table contains the status of the fork on its dismissal. The status is also put in X.

   BRS 32 causes a lower fork to be unconditionally terminated and its status to be read into the panic table.

   BRS 106 causes the controlling fork to be dismissed until any subsidiary fork causes a panic. When it does, the controlling fork is reactivated at the following instruction with the panic table address in A, and the panic table contains the status of the fork at its dismissal.

   BRS 107 causes BRS 30 to be executed for all subsidiary forks.

   BRS 108 causes BRS 32 to be executed for all subsidiary forks.

2. If interrupt 3 is armed in the controlling fork, the termination of any subsidiary fork will cause that interrupt to occur. The interrupt takes precedence over a BRS 31. If the interrupt occurs and control is returned to BRS 31 after processing the interrupt, the fork will be dismissed until the subsidiary fork specified by the restored (A) terminates.

3. The forks can share memory. The creating fork, can, as already indicated, set the memory of the subsidiary fork when the latter is started.

PANIC CONDITIONS

The three kinds of panic conditions that may cause a fork to be terminated are listed in the description of the status word. If the panic was caused by an escape, the following occurs to the fork being pointed to by TTYASG, and to all lower forks.

1. The page in the parent fork that contains the panic table is brought into core if necessary. Data is inserted into the panic table.
2. The B and X registers are stored into the TS page.
3. The PAC table is returned to the free PACT list. The only exception is that the PAC table of the Executive is not released by any panic condition.

If the panic was not caused by an escape, the above three steps will affect the fork causing the panic.

The PAC table of the controlling fork (or the fork above the one being pointed to by TTYASG) is put onto QIO with an immediate activation condition. If TTYASG contains the Executive PACT pointer then the Executive fork is placed on QIO. When the controlling fork is activated, execution will begin at the location indicated by PL. For user forks this will be one instruction after BRS 9.

The panic that returns a status word of 0 is called a fork panic and may be caused by either of two conditions:

1. The escape button on the controlling teletype is pushed, or an off interrupt occurs. This terminates a fork with a fork panic. A fork may declare that it is the one to be terminated by executing BRS 90. If a user fork is terminated by escape, the teletype input buffer is cleared. If the controlling fork of the terminated fork is executive, the output buffer is also cleared.

   If a fork to be terminated by escape has armed software interrupt 1, the interrupt will occur instead of a termination. The teletype buffers will not be affected.

   If the Executive is activated, control goes to the location EXECP in the Executive. Executive programs can turn the escape button off with BRS 46 and turn it back on with BRS 47. An escape occuring in the meantime will be honored when BRS 47 is executed. A program which is running with escape turned off is said to be nonterminable. BRS 26 skips if there is an escape pending.

   If two escapes occur within approximately 0.12 seconds, the Executive fork will be activated. This has the same effect as having TTYASG contain the Executive PACT pointer. (This device permits a user trapped in malfunctioning lower forks to escape). Closely spaced escapes can be conveniently generated with the repeat button on the teletype.

2. A BRS 10 can be executed in the lower fork. This condition can be distinguished from a panic caused by the escape button by the fact that, in the former case, the program counter in the panic table points to a word containing BRS 10.

An extension of this system provides a way in which several forks may be terminated simultaneously by a lower fork. BRS 73 provides a count in the A register. A scan is made upward through the fork structure, decrementing this count by one each time a fork is passed. When the count goes to 0, the scan is terminated and all forks counted are terminated. If an executive program is reached before the count is 0, then all the user programs below it are terminated.

The panic which returns a status word of 1 is caused by the execution of an illegal instruction in the fork. There are two kinds of illegal instructions:

1. Privileged machine instructions.
2. SYSPOPs, either forbidden to the user, or provided with unacceptable arguments.

A status word of 2 is returned by a memory panic. This may be caused by an attempt to address more memory than is permitted by the machine size the user has set, or by an attempt to store into a read-only page. If interrupt 2 is armed, it will occur instead of the memory panic.

Note that no type of panic releases memory. The PMT entries remain intact. This allows pages of memory that are acquired by one fork to be shared with the other forks in the hierarchy. Also, after the panic occurs, the parent fork can again issue the BRS 9 and resume execution in the lower fork.

4. SWAPPER, MEMORY ALLOCATION AND RAD ORGANIZATION

SWAPPER

The swapper accomplishes the allocation of memory. It is called to activate a fork, change relabeling, or acquire a page of memory. The input to the swapper is the pseudo-relabeling. To determine the exact location of each of the pages the fork requires, the pseudo-relabeling is decoded and the SMT and PMT tables are consulted. The swapper can then determine how many pages need to be read in from the RAD. This count in then compared with the Memory Availability Count (MAC). MAC contains the number of unlocked pages (minus one).

The system keeps tables that define the status of real memory. These tables, both indexed by real page number, are the Real Memory Table (RMT) and Real Memory lock Count (RMC). The RMC entry indicates whether a page is locked or unlocked. If RMC = -1, the page is unlocked and available for swapping. An RMC entry may be made non-negative (the page can be locked) for any of the following reasons:

1. Part of the Monitor is in the page.

2. The resident user occupies the page.

3. The page is I/O bound. This implies that the page contains an I/O buffer that is currently active. Pages which are being swapped are also I/O bound. Any routine that initiates an I/O operation will increment the appropriate RMC entry. The I/O interrupt routine will decrement the RMC entry when the operation has been completed.

4. The scheduler locks the pages of a fork that is on SWQ so that the memory will not be assigned to the second fork that could be placed on SWQ. The memory is unlocked when the fork is activated.

If the number of pages a fork requires is greater than MAC, the swapper is unable to allocate memory at this time and exits with an abnormal return.

If sufficient memory is available, the pages to be swapped are selected. The RMT table is scanned to determine the optimum pages to be swapped. The format of RMT is shown in Figure 10

At most, three passes are made through the RMT tables to select the required number of pages. The following method is used to determine which pages are to be released:

1. No locked page (see RMC) is released.

2. Pages that are not locked are selected in the following manner:

   a. User pages (PMT pages) marked as read-only (RMT bit 1 = 1, bit 2 = 0). No RAD write is required in this cast since there is a valid copy of the page on the RAD.

   b. User pages that are not read-only (RMT bit 1 = 0, bit 2 = 0). Requires RAD write-out.

   c. SMT pages (RMT bit 2 = 1).

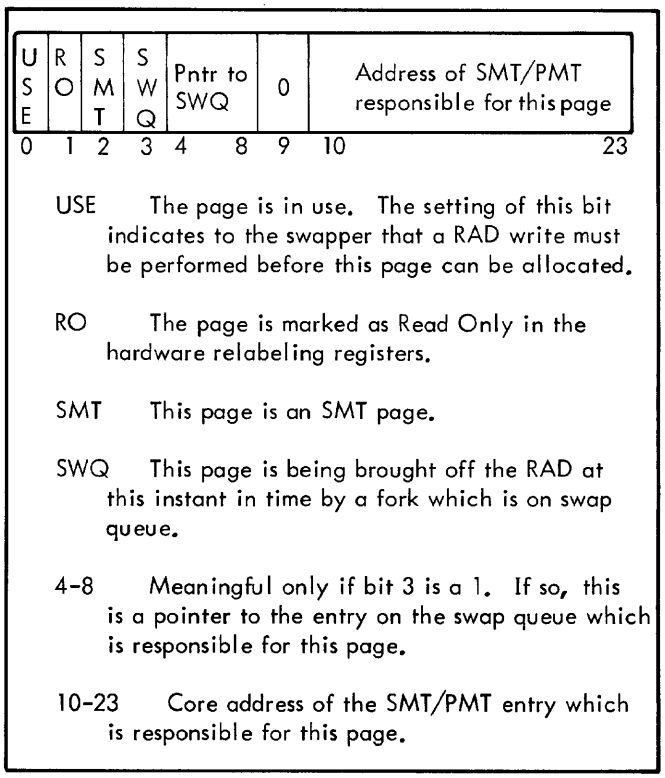
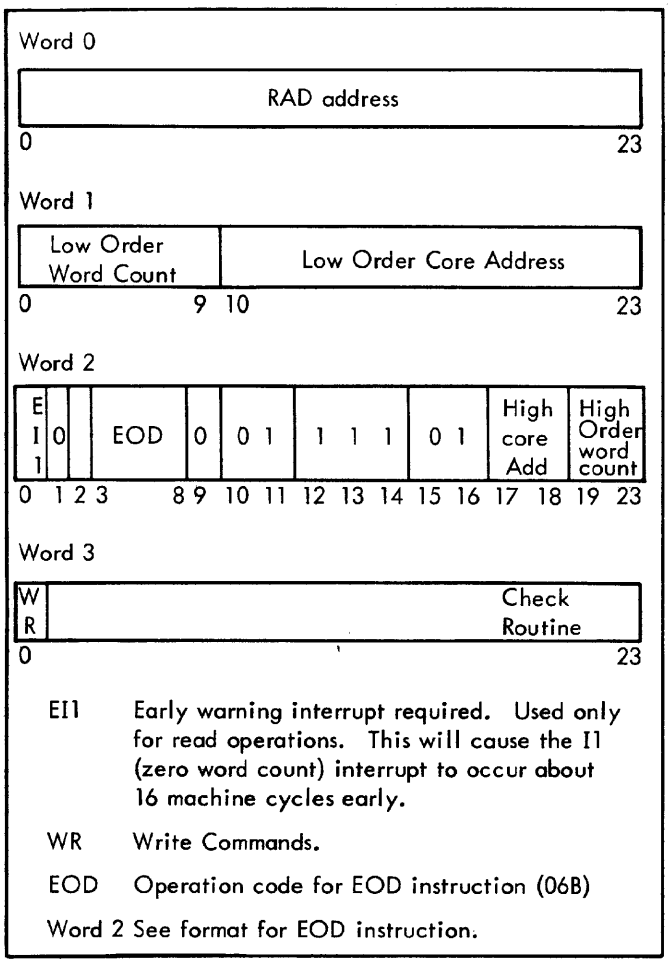


Figure 10. Format of Real Memory Table (RMT)

This scheme provides the SMT pages, which are the most likely to be used within the next few activations, with the best chance of remaining in core.

Memory is allocated and a list of RAD operations is constructed. After the pages which are to be written out to the RAD have been selected, all the read and write commands are placed on the RAD queue. The commands are placed on the queue in a sequence that guarantees minimum rotational latency. When a RAD read is put on the list, the actual (real) memory address of the page is placed into the low order five bits of the PMT or SMT entry. Pages that have been selected to be released are marked in the SMT/PMT entry as being on the RAD. When all the commands have been placed on the RAD queue, the RAD driver is called. Figure 11 shows the format of the RAD queue.

If the swapper was called in the process of activating a fork (MGTS5 = 0; see Function of the Scheduler), certain parameters must be set to indicate that the fork is being placed on swap queue. As each RAD read command is



EI1 Early warning interrupt required. Used only for read operations. This will cause the I1 (zero word count) interrupt to occur about 16 machine cycles early.

WR Write Commands.

EOD Operation code for EOD instruction (06B)

Word 2 See format for EOD instruction.

Figure 11. RAD Queue Entry

added to the queue of RAD commands, a pointer to the swap queue is inserted in bits 3 through 8 of the appropriate RMT entry. Since MGTS5 = 0, the swapper does not wait for the completion of the RAD I/O. Now, all the SMT or PMT entries in the current pseudo-relabeling are examined and a set of real relabeling registers is constructed. The real relabeling registers are saved in the current input position of SWQRL1, SWQRL2, and SWQRL3 and are used to lock the fork's memory in core until it is activated.

Bit 1 of RMT indicates a read-only page. This bit is set for all pages being read from the RAD. The bit is set in the OMR routine which sets up the read commands. When the user's real relabeling is constructed in PKRL, all pages are marked as read-only. As the relabeling is being prepared for output in LABEL, the TS page is marked as not-read-only to facilitate the handling of the read-only trap. As the user runs, any attempt to store into a user page results in a read-only trap and the code at TRAPR determines whether the page referenced is a true read-only page or not. If the page is not a true read-only page, then bit 1 of the RMT entry is cleared, and the user's relabeling is changed so that the page is no longer read-only.

If a running fork calls the swapper to change its relabeling (or acquire more memory), a similar path is taken with the following exceptions: The swapper waits until the RAD I/O is completed; and the real relabeling registers are constructed and stored in the fixed Monitor locations RRL1, RRL2, and RRL3 and output to the hardware relabeling registers.

When the scheduler attempts to activate a fork that is on the swap queue, it calls the swapper with MGTS5=-1. When the swapper examines the pseduo-relabeling, it should find all of the required pages in core and set the hardware relabeling. If the pages are not core resident, a RAD error has occurred. The RAD interrupt routine marks a page (in the SMT/PMT entry) as being on the RAD and changes the RMT entry to indicate that the page is not in use whenever a read error has been detected. The swapper will again attempt to bring this fork into core and since MGTS5=-1, the system will wait for RAD I/O completion. If a RAD error again occurs, the swapper will take an abnormal return. The scheduler will then dismiss this fork on QQE with an immediate activation condition.

MEMORY ALLOCATION

The method of memory acquisition is described in Chapter 3. Pages can be completely released by use of BRS 4 or BRS 121. BRS 4 requires an address (virtual) in A: BRS 121 requires a relabeling byte. When the BRS is executed, the PMT entry for the page is cleared, the pseduo-relabeling bytes are zeroed for all forks in the structures, and the RAD map is adjusted to indicate the availability of the page. A page released in this manner is unrecoverable. A user cannot use these BRSs to release an SMT page or his TS page. (See EX bit in PMT/SMT).

BRSs that are restricted to Executive forks include:

116	Read relabeling from user's TS page
117	Set relabeling in user's TS page
120	Obtain a page
56	Make page Executive (see EX bit in the PMT)
104	Read a page from the RAD
105	Write a page on the RAD.

RAD ORGANIZATION

RAD space is allocated at the rate of one page (2K words) at a time when requested. A bit map, DRAT, with one bit for each page on the RAD, is used to determine which pages are available. When a user requests a page of memory, the code at PMTA assigns a space on the RAD for the new page so that the user's pages

will be rotationally consecutive in their order of occurrence in his PMT. This means that, although two pages which are consecutive in the PMT may be quite far apart on the RAD, they may be read in with no rotational delay between them. The job number of the user determines whether the first page of the user's memory is assigned an even or an odd position on the RAD. When a user releases a page of memory, the code at MPUT3 returns the appropriate bit to bit map. The first 64 pages of the RAD contain the subsystems and Executive.

5. SOFTWARE INTERRUPTS

A facility is provided in the Monitor to simulate hardware interrupts. There are eleven possible interrupts: five are reserved for special purposes and six are available to the programmer for general use. A fork may arm the interrupts by executing a BRS 78 with an 11-bit mask in the A register. This causes the appropriate bits in PIM to be set or cleared to correspond to the bits in the mask. Bit 4 of A corresponds to interrupt number 1, etc. No other action is taken at this time. When an interrupt occurs the execution of an +SBRM* to location 200 plus the interrupt number is simulated in the fork that armed the interrupt.

Whenever any interrupt occurs, the corresponding bit in the interrupt mask is cleared and must be set explicitly if it is desired to keep the interrupt on. Note that there is no restriction on the number of forks which may have an interrupt on.

To read the interrupt mask into A, the program may execute a BRS 49.

INTERRUPTS 6 THROUGH 10

A fork may generate an interrupt by executing a BRS79 with the number of the desired interrupt in the A register. This number may not be one, two, three, four, five, or eleven. The fork that arms the interrupt should not be the one that triggers it using the BRS 79 (i.e., a fork should not interrupt itself using the BRS79). The interrupt causes the fork structure to be scanned upward. The first fork with the appropriate interrupt mask bit set is interrupted. The interrupted fork is put on QIO with an activation condition of 5 @ interrupt number. Execution of the program in the fork causing the interrupt continues without disturbance. If no interruptable fork is found, the interrupt instruction is treated as a NOP. If there is an interruptable fork, it skips on return.

SYSTEM INTERRUPTS

Interrupts 1, 2, 3, 4, 5 and 11 are the system interrupts. They can be caused by the same fork which has the interrupt armed.

If the fork which is being pointed bo by TTYASG also has interrupt 1 armed, a program panic (BRS 10 or escape key) that would normally terminate the forking structure, will instead cause interrupt 1 to occur. The fork will be placed on QIO and begins execution at the location indicated by the contents of location 201B. This permits the programmer to control the action taken when the escape key is pushed, without establishing a fork specifically for this purpose. If depressing the escape key causes an interrupt to occur rather than terminating a fork, the input buffer will not be cleared.

If a memory panic occurs in a fork that has armed interrupt 2, it will cause interrupt 2 to occur rather than terminating the fork. If an illegal instruction panic occurs in an executive fork that has armed interrupt 2, if will cause interrupt 2 to occur rather than terminating the fork.

Interrupt 3 is caused, if armed, when any lower fork terminates. Interrupt 4 is caused, if armed, when any input/output condition occurs that sets a flag bit (e.g., end of record, end of file and error conditions).

Interrupt 5 is caused, if armed, when overflow occurs in the Floating Point Arithmetic Unit (FPAU). This interrupt is unique in that it may be armed by the user for use as a time out interrupt, but may not be triggered by use of the BRS 79.

Interrupt 11 is caused, if armed, if a disc error is encountered during a BRS BE + 1 or BRS BE + 2. These BRSs require system status. Consequently, interrupt 11 has no meaning for user or subsystem forks.

TIME-OUT INTERRUPTS

A fork may be interrupted after a specified period of time by issuing BRS BE + 12. It takes the interrupt mask in A, the time (in msec) in B, and the interrupt number in X. If the specified interrupt is armed when the time runs out, the fork will be interrupted.

The interrupt number which is specified in X may be any of the user interrupts (6 through 10). A fork may have a maximum of 3 time-out interrupts pending. The number of time-out interrupts that are pending is noted in the TO entry of the PAC table.

+SBRM* When a fork is being activated because of a software interrupt, the scheduler simulates the execution of a BRM* 200B+N where N is the interrupt number. See description of BRS 78.

6. BRS LOGIC

The BRSs are divided into classes 1, 2, and 3. The class of each BRS is listed in Appendix B. See Appendix M for flow chart of BRS logic. See the 940 Computer Reference Manual for a discussion of SYSPOP logic.

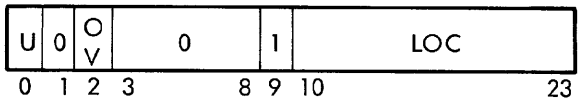
The majority of BRSs are class 1. This class must not call any other BRS since the contents of location zero would be altered.

The string processing system is implemented as class 2 BRSs. The system will save the contents of location zero when any of these BRSs is executed. Therefore, a class 2 BRS may call a class 1 BRS.

Both class 1 and class 2 BRSs execute entirely in Monitor mode. Thus, the calling fork will not be dismissed until execution of the BRS has been completed. The coding for these BRSs is entirely within the resident Monitor.

Any BRS which requires a considerable time to execute or deals with file manipulation is implemented as a class 3 BRS. The class 3 BRSs (also called Executive BRSs since much of the coding is within the Executive) all declare a lower fork to execute. This fork runs in user mode and therefore may be dismissed during execution. A class 3 BRS may call both class 1 and class 2 BRSs.

When a BRS is executed, flow enters the BRS file at location BS. Absolute location zero contains:



U	BRS executed in user mode
OV	Status of overflow indicator
LOC	Location of the BRS instruction. This will be a virtual address if BRS was executed in the user mode.

The contents of the central registers are stored into SS01, SS02, and SS03.

The number of the BRS can be obtained by referencing location 0. The transfer vectors for the BRSs are stored at locations BST through BSTU. Location BSX is set to contain the transfer vector. The transfer vector will be:

BSX =			
	BRU	ROUT1	for a Class 1 BRS
	EAX	ROUT2	for a Class 2 BRS
	NOP	N	for a Class 3 BRS
	BRM	TRAPB	for an unimplemented or nonexistent BRS

Flow is transferred to a class 1 BRS when the system executes an EXU BSX. Real location 0 will contain its initial setting while the BRS is executing. A class 1 BRS returns to the calling program by branching to the POPX routine. POPX will restore the control registers and execute a BRR 0. If it is not necessary to have the central registers restored (the BRS may return data in the registers), the BRS may exit by only executing a BRR on location zero. If the BRS gives a skipping (exception) return it will increment location 0 before branching to POPX.

While a class 2 or 3 BRS is executing, location SBRSRT contains the initial setting of location 0. Flow is transferred to a class 2 BRS when the system executes a BRU* BSX.

This BRS returns to the calling program by branching to the EPOPX routine. EPOPX executes a BRR SBRSRT.

The value N in the transfer vector of a class 3 BRS indicates what pseudo-relabeling the BRS fork should have and provides information for setting the fork's PL word. A PAC table is obtained and initialized. The pseudo-relabeling includes the TS page, the COMPG file, and either the GSBR or the FLTIO file. The PL word is initialized to begin execution at a jump table in either GSBR or FLTIO (see Executive files in Appendix E). The contents of location zero and the central registers are saved in locations UPL, UPA, UPB, and UPX in the user's TS page. The BRS fork is put on QIO. The parent fork (the fork which executed the class 3 BRS) is dismissed with an activation condition of 7 @ 4.

The BRS fork will execute a BRS 111 when it has finished executing. The BRS 111 will delete the BRS fork's PAC table. The contents of location UPL are stored into location 0. The swapper is called to relabel in the parent fork. A branch is then made to POPX.

7. INITIALIZATION AND TERMINATION OF A USER

INITIALIZATION OF A USER

When the user dials on the system, the "teletype on" interrupt is generated. The interrupt routine places a task on the phantom user. The phantom user will process the task when four seconds (from the time the interrupt was sensed) have elapsed. This d elay in processing allows time for the teletype carrier signal to become stable.

The major part of the coding for this phantom user task is performed in a routine named TSON. TSON assigns a job number to the user and acquires a PAC table for the Executive fork provided there is a job number and a PAC table available. The PAC table is initialized to contain the pseudo-relabeling and status parameters applicable to an Executive fork. The PL word is initialized to begin execution in the TSONI routine. (TSONI is a monitor routine.) The PMT table that corresponds to this job and the WERIS entry for the teletype are zeroed, and TTYASG is initialized to contain the PAC pointer of the Executive fork. The PAC table is then added to the QTI queue.

When this fork is allotted a time slice, execution begins at TSONI. A TS page is acquired and relabeled into logical page zero of the Executive relabeling and logical page seven of the monitor relabeling. The Executive subroutine transfer vectors and other constants in the TS page are initialized. A branch is then taken to a location in the Executive, causing a transfer to the user mode. The Executive then attempts to log the user onto the system. If the user does not successfully log on within 90 seconds, his teletype is deactivated and the Executive fork is terminated.

The system will handle a maximum of 40 users (i.e., there are 40 job numbers available). However, more than 40 teletypes can be handled by the communications equipment. Therefore, it is possible to receive a "teletype on" interrupt and have no job numbers or PAC tables available. In this case the teletype number of the user is placed onto a dial-in-and-wait queue. A task is placed on the phantom user to type a message to the user indicating that he should wait until the system can accept him. A second task is placed on the phantom user that causes the system to check at certain intervals for the availablity of a job number and a PAC table. When it is possible to accommodate this user, he is initialized to the system as described above.

TERMINATION OF A USER

A user can indicate the termination of his job by giving the LOGOUT or EXIT command, or by hanging up the teletype. The LOGOUT command releases all the program memory. (clears PMT entries 61B through 77B), writes the user's file directory, outputs accounting information the system requires for billing, prints an elapsed time message to the user, closes all files, and releases the TS page. The Executive then executes a BRS 112 which resets such tables as WERIS, TTYASG, LCW (used for linking), releases the PAC table for the Executive fork, removes any tasks from the phantom user that apply to this teletype, and deactivates the teletype if the operator has issued the SHUT DOWN command. The BRS 112 then causes the same task as that which results from a "teletype on" interrupt to be added to the phantom user's task queue. This allows another user on the same teletype to log onto the system provided that the first user did not hang up.

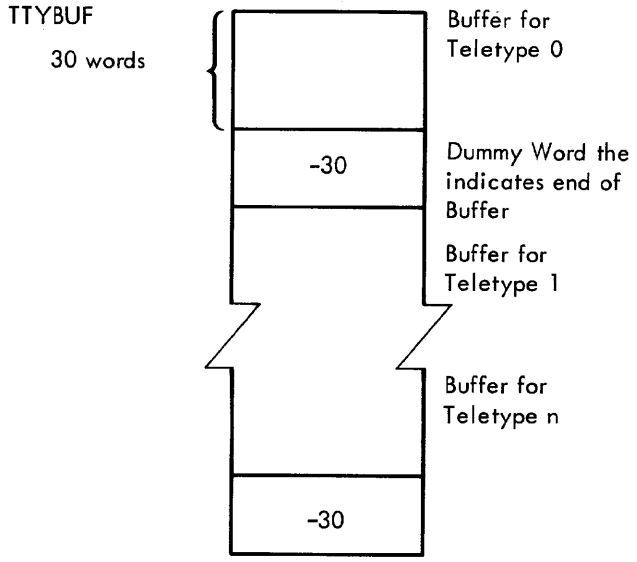
The EXIT command is similar to LOGOUT except that the user's file directory is not written. Therefore, any user files established during this session are not entered into his directory.

The "teletype off" interrupt places on the phantom user a task that is similar to the processing of a high speed escape. With the exception of the Executive fork, the entire forking structure is terminated. The teletype input and output buffers are cleared and the Executive fork is placed on QIO with the PL word containing OFFINT (OFFINT is a location in the Executive). At OFFINT a dump file of the user's memory is taken if the user has established a /$/ file and has not logged off the system. The path taken now is the same as if the user had given the LOGOUT command. That is, the memory is released, file directory written, etc., and finally the BRS 112 is executed.

8. TELETYPE INPUT/OUTPUT

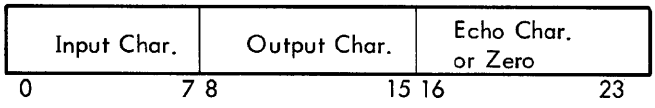
TELETYPE BUFFER

Monitor file TTY contains the teletype buffers, pointers, and tables. (See Figure 12.) Each teletype has one buffer which is 30 words in length. The label on this buffer is TTYBUF. TTYBUF is initialized in TTYSET.



The teletype buffers are "ring buffers". Pointers indicate where the next character is to be read into (or taken out of) the buffer. Assume a character for TTY 0 has just been put into TTYBUF + 29. When the next character comes in, the dummy word will be detected, the input pointer will be adjusted by the value of the dummy word (-30), and the character will be placed into location TTYBUF.

The format for each word in TTYBUF is:



When a character is typed on a teletype, it is converted to 940 internal code and added to the input buffer. The echo character is in trimmed ASCII. The output character is formed by adding 240B to the internal code.

Although the input and output buffers share the same locations, separate pointers allow the buffers to be manipulated independently of each other. Teletype associated variables are shown in Figure 12.

OUTPUT PATH

To output a character from location M, the SYSPOP

TCO M (teletype character output)

is used. This instruction outputs a character from the rightmost eight bits of location M. Normally, the character is in internal format.

If the user executes a TCO instruction, TOS5 is incremented and the character is placed into the location indicated by TOS5. The character count in TOS2 is also incremented. If this is the first character that is being sent out (TOS2 is negative) the character out interrupt must be initiated. This is accomplished by "potting" 001400CN where CN is the teletype channel number. The setting of bit 9 will cause the character out interrupt to be generated as soon as the buffer within the CTE (Communications Teletype Equipment) is empty. Note that this interrupt will be sent each time the CTE has finished processing an output character.

The interrupt routine will increment TOS4, decrement TOS2, and output a character from the location pointed to by TOS4.

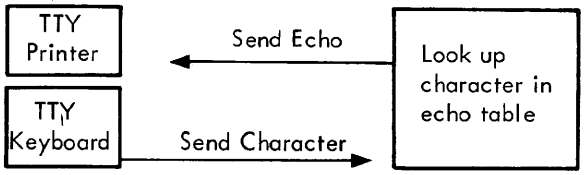
A fork will not be dismissed for teletype output unless the buffer is full (i.e., TOS2 = 30). When the buffer is full, the fork is dismissed with an activation condition of 3 @ TOS2 + CN. The fork will be reactivated when TOS2 is less than or equal to the teletype early warning (TTYEWM) — a system parameter with the value 6.

If a fork wishes to send out a multiple number of spaces, a 135B (the multiple blank character) is sent. The next character that is sent represents the number of blanks to be output. The TOS3 indicator keeps track of the detection of the 135B and the outputting of the blanks.

The output interrupt will respond with both a carriage return and a line feed if either character is detected. An exception to this rule occurs if the user outputs a carriage return (or a sequence of carriage returns) when the system determines the carriage is already at the left margin. In this case, only line feeds will be output.

ECHO TABLES

The teletypes used are fully duplexed to allow simultaneous keyboard transmission and reception. That is, when the user types a character, no printing occurs. As illustrated below, the character is transmitted to the 940 and the software echoes it back (prints the character). Four echo tables are available. Presently the system may choose to print or not to print a character (echo or not).



The way the echo table logic is implemented, it would be hypothetically possible to modify an echo table so that when an A is typed a Z is echoed. The input character serves only as a pointer into the echo table. The corresponding table entry is then output to the teletype.

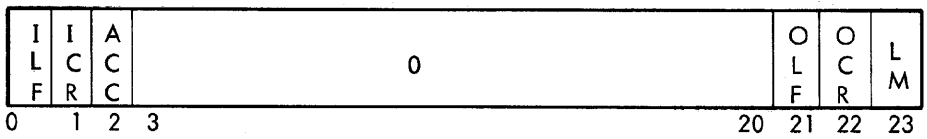
TOS2 Number of characters in output buffer; has the value -1 when output buffer is inactive.

TOS4 Output readout pointer. The output interrupt routine outputs a character from the location indicated by TOS4.

TOS5 Output write-in pointer. Pointer to next available space in the output buffer, e.g. the TCO routing will place a character in the location indicated by TOS5.

TOS3 0 – not in multiple blank mode; 20000000B – just received 135 (multiple blank character); other – number of blanks to be output. This is used to simulate the tab settings on a typewriter.

TIIS5



ILF Last input character was a line feed

ICR Last input character was a carriage return

ACC See description of BRS BE + 11

OLF Last output character was a line feed

OCR Last output character was a carriage return

LM Carriage is at left margin

ATIS2 Number of "pinned" words in ATTBUF. This location is incremented by the input interrupt routine and decremented by the 205 interrupt routine.

ATIS5 Write-in pointer. The input interrupt routine will put a word into the location pointed to by ATIS5.

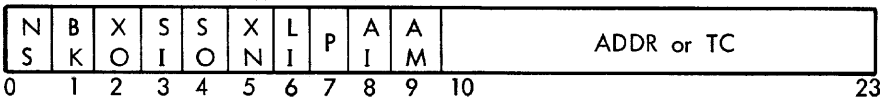
ATIS4 Read-out pointer used by the 205 routine. A word is pulled out of ATTBUF from the location pointed to by ATIS4.

TIS2 Number of characters in the input buffer. This word is incremented by the 205 routine and decremented by any routine requesting teletype input (such as TCI).

TIS4 Write-in pointer used by the 205 routine. A character is pulled out of ATTBUF and put into TTYBUF in the location pointed to by TIS4.

TIS5 Read-out pointer used by any routine requesting teletype input (such as TCI). A character is read out of TTYBUF from the location pointed to by TIS5.

TTYTBL



NS Not 8-level mode

BK Waiting for a break character

XO Paper tape reader is to be turned off because input buffer is becoming full.

SI 8-level input

SO 8-level output

XN Paper tape reader has been turned off and should be turned on again when there is room in the buffer.

LI Input buffer full. "Don't listen for input bit"

P Output routine is in the process of turning on the paper tape reader.

AI Accept linked input bit. Currently not used.

AM Accept linked output. Teletype is willing to be linked.

ADDR Address of echo-table

TC Terminal character for 8-level output

NOTE: These tables are indexed by teletype channel number.

Figure 12. Teletype Tables

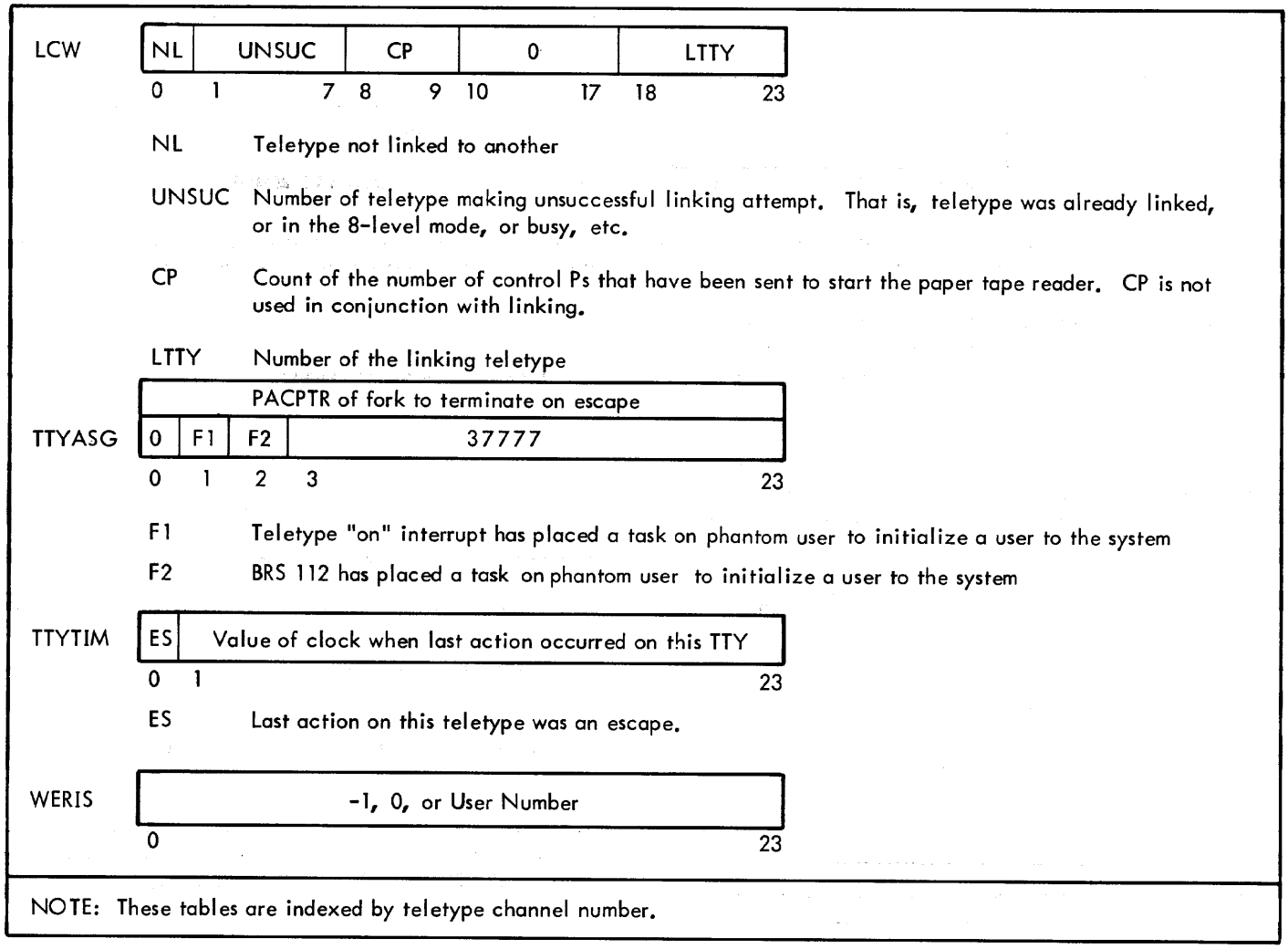


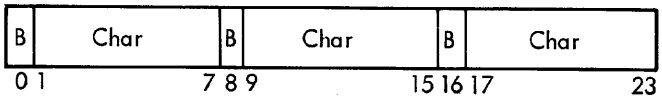
Figure 12. Teletype Tables (cont.)

In addition to specifying the echo character, the echo tables also define the "break" characters. A fork will not be dismissed for teletype input unless the input buffer is empty. However, once the fork is dismissed, it will be reactivated again when a break character has been input or the buffer is almost full. Suppose a program requires an entire line of text before it can do any processing. It could then define the control characters as break characters. Once the program has been dismissed, it will not be reactivated until the entire line is available.

There are four standard echo tables in the system, referred to by the numbers 0, 1, 2, and 3. Zero is a table in which the echo for each character is the character itself, and all characters are break characters. Table 1 has the same echoes, but all characters except letters, digits and space are break characters. Table 2 also has the same echoes, but the only break characters are control characters (including carriage return and line feed) and exclamation mark. Table 3 specifies no echo for any character, and all characters are break characters. This table is useful for a program that can either compute the echo itself or direct that no echo be sent. The Executive uses echo table 3 to suppress password printing.

Each echo table is 32 words (3 characters per word) in length. The 8-bit characters are stored in trimmed ASCII code. The total character set includes 96 characters. The input character is converted to trimmed ASCII code. Therefore, the table look-up is done with a character that has a value between 0 and 137B.

The format of the echo table is:



where

B is 1 if this is a break character.

Char is character to be echoed in trimmed ASCII code. If the character is not to be echoed, character is set to 1.

The echoes for characters having a trimmed ASCII value of 0, 1, or 2 are in word 0 of the echo table; 3, 4, or 5 are in word 1; etc.

Consider echo table 0 which will echo everything and break on everything. The echo for a B, C, or D (ASCII 102, 103, 104) will be in word ET0+26B and have the value

6 0 5 4 1 7 0 4B =

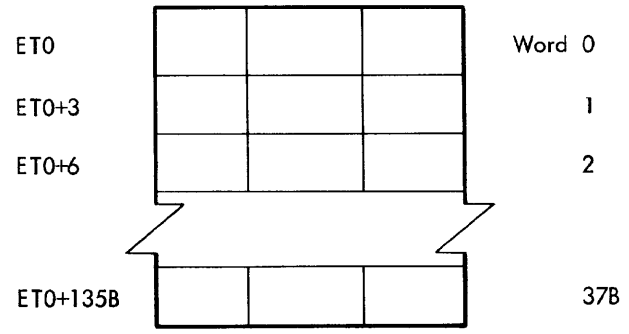
11000010	11000011	11000100
302B	303B	304B

= 102B, 103B, 104B plus break character bit set

The echo table a fork wishes to use can be set by executing a BRS 12 which causes TTYTBL (indexed by the teletype number) to contain the address of the chosen echo table.

940 BYTE ADDRESSING

An echo table can be thought of as a table of 96 consecutive bytes. For echo table 0, the byte addresses of the entries in the table would have values between ET0 and ET0+137B.



Suppose an ASCII character has been input and the trimmed 7 bits have been stored in CHAR in bits 14-23.

```
LDA   CHAR
MUL   =12525253B     Divided by three
```

A byte address can be converted to a word address by dividing it by three. The remainder from the division represents the byte position that the character has in the word. The multiplication by 12525253B (this constant effects a divide by three) will leave the word address in A, and bits 0 and 1 of B will contain 00, 01, or 10. We then

```
ADD   TTYTBL,2   Get address of Echo Table
CAX
LDX   0,2        Load X with echo word
LCY   5          Get 5 bits of B
ETR   =30B       Extract 00, 10B or 20B
COPY  XB, AX     Echo word in B, Shift count in X
LCY   8,2
```

The proper character will now be in the last 8 bits of the A register. For example, in trimmed ASCII:

```
B = 102B    102B/3 = 26B    (Remainder = 0)
C = 103B    103B/3 = 26B    (Remainder = 1)
D = 104B    104B/3 = 26B    (Remainder = 2)
```

Therefore, the echoes for B, C, and D are in word 26B of the echo table in positions 0, 1, and 2, respectively.

Note that this technique of byte addressing is also used by the string processing system.

8-LEVEL MODE

Special provision is made for reading 8-bit codes from the teletype without sensing escape or doing the conversion from ASCII to internal. To switch a teletype into this mode, execute:

```
LDX   teletype number
LDA   terminal character + 40000000B
BRS 12
```

This will cause each 8-bit character read from the teletype to be transmitted unchanged to the user's program. The teletype can be returned to normal operation by

1. Reading the terminal character specified in A.
2. Setting the echo table with BRS 12

No echoes are generated while the teletype is in 8-level input mode. Teletype output is not affected.

A parallel operation, BRS 85, is provided for 8-level output. While in 8-level output mode, the system does no special processing of line feed and carriage return characters. The BRS 86 will reset 8-level output mode, as does any setting of the echo table.

INPUT PATH

To input a character from the controlling teletype (the teletype on which the user of the program is entered) into location M in memory, the SYSPOP

```
TCI     M     (teletype character input)
```

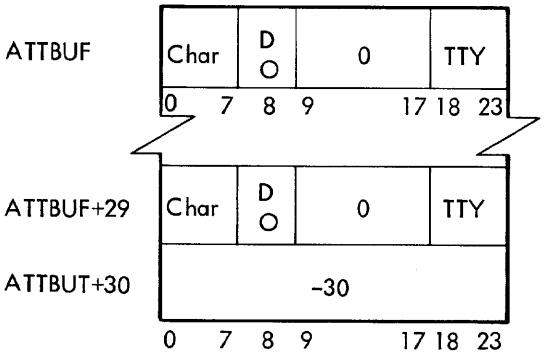
is used. This SYSPOP reads the character from the teletype input buffer and places it into the 8 rightmost bits of location M. The remainder of location M is cleared. The character is also placed in the A register, which destroys the former contents.

Two interrupts and an additional buffer are associated with the input path. When the input interrupt occurs, the word is "pinned" into a buffer called ATTBUF.

The input interrupt routine then arms the 205B interrupt so that a lower priority routine can complete the processing of character. Since the 205B interrupt is "hard wired" (always triggered), it will go to the waiting state as soon as it is armed.

The 205 interrupt routine takes a word from ATTBUF, gets the appropriate echo character, attempts to process the echo, puts the input character into TTYBUF, and decides if a break character has just been received.

ATTBUF (one per system) has the following format:



where

Char is ASCII character.

DO is Data Overrun.

TTY is Teletype Number

Suppose a fork exexutes a TCI. As long as TIS2 is greater than 0 a character is extracted from TTYBUF using the TIS5 pointer. When the 205 routine places this character into TTYBUF, it sends the echo if one is required, and none of the the following conditions prevail: (1) output active, i.e., TOS2 is not negative, (2) no previous echoes were deferred, (3) teletype is not linked. If these conditions are absent, the echo is sent and bits 16 through 23 of the word pointed to by TIS5 are reset. The echo is sent by placing it into the output position of TTYBUF.

If the echo is not sent immediately (205 routine), it will be sent later by the TCI routine. The echo transmission is accomplished by moving the echo character to the output position of the TTYBUG word pointed to by TOS5. If the teletypes are linked, the echo will be placed into the output buffer of the linked teletype as well.

IF TIS2 is zero, the input buffer is empty, and the fork is dismissed with an activation condition of 11 @ TTYTBL. Bit 1 of TTYTBL is set to indicate that the fork is waiting for a break character.

The 205 routine resets bit 1 of TTYTBL when a break character is detected and also increments ACTR so that the scheduler knows that a fork on QTI is ready. The routine will take the same action whether a break character has been detected or not, if the buffer is within 10 characters (TTYEEW) of becoming full.

The buffer becoming full presents a problem when the input is from paper tape. If the buffer is within 6 characters of becoming full, bit 2 (XOFF) of TTYTBL is set and the dummy output interrupt is sent. When the output interrupt routine is entered and XOFF is set, the paper tape reader is turned off. XOFF is then reset and XON (Bit 5 of TTYTBL) is set to indicate that the reader must be turned back on. When the input buffer becomes empty and XON is set, the dummy output interrupt is generated. XON is reset and bit 7 of TTYTBL is set to indicate that the output interrupt routine should send out two interrupts and the XON ASCII character to reactivate the paper tape reader. Bits 8 and 9 of LCW are set to keep track of the number of interrupts that that have been sent. The two interrupts are sent for timing purposes. This guarantees that the teletype has finished processing the XOFF request before turning it back on.

MISCELLANEOUS TABLES

Any time an action occurs at a teletype (either input or output) the value of REAL is stored into TTYTIM. If a user hits two high-speed escapes, this indicates he wishes to return to the Executive fork. When an escape is processed, the value in TTYTIM is compared with REAL to determine if this was a high-speed escape. TTYTIM is set to -1 if a high-speed escape was received. TTYTIM is also set to -1 in TSOFF when an interrupt has been received.

TTYASG is initialized to 37777B when the teletype becomes inactive. An array named TTYBIT and the high-order bits of the TTYASG array are used to allow the system to ignore duplicate "on" and "off" interrupts. TTYBIT is a two-word array whose bit positions correspond to a teletype number (i.e., bit 0 is associated with teletype 0, bit 1 with teletype 1, etc.) When the teletype line is not in use, the corresponding bit in TTYBIT is set and TTYASG contains the value 37777B.

When the teletype "on" interrupt is received, the appropriate bit in TTYBIT is reset (to indicate that an "on" interrupt has been received) and a one is merged into bit position one of TTYASG (to indicate that the task to initialize a user to the system has been placed on the phantom user queue). Before the phantom user processes this task, checks are made to determine if the carrier is stable and the data set is ready. If the data set is not ready, TTYBIT and TTYASG are initialized to indicate that the teletype is not in use, the teletype line is activated to allow another user to access the line, and the remainder of the task is ignored. If the carrier and the data set are ready, the Executive fork is initialized and TTYASG now contains the PAC pointer of the Executive fork. If the corresponding TTYBIT is reset when the "on" interrupt is received, the interrupt is ignored and the above processing does not occur.

Throughout a user's session, TTYASG is modified to contain the PAC pointer of the fork to be terminated (or interrupted if software interrupt 1 is armed) when an escape occurs.

User execution of either the LOGOUT or EXIT commands causes eventual execution of the BRS 112 (see Chapter 7). The BRS 112 will place the initialization of a user task on the phantom user queue so that another user at the same teletype may log onto the system. TTYASG is now set to contain the value 1037777B. The initialization task is processed in the manner described above. If the user disconnects after issuing the LOGOUT or EXIT command, the task will not be processed since the data set will not be ready.

When a teletype "off" interrupt is received, TTYBIT must be reset or the "off" interrupt processing will not occur (essentially, the interrupt is ignored). If bit one of TTYASG is set, the "off" interrupt is also ignored, since an "on"

interrupt has been received but has not been processed. The "off" interrupt can be ignored since the "on" interrupt processing will not occur if the data set is not ready.

If the "off" interrupt is legitimate, TTYBIT should be reset. TTYASG should contain a negative PAC pointer or bit two of the word should be set if the user executed to LOGOUT or EXIT command before disconnecting. If TTYASG is negative, the "off" interrupt processing task is placed onto the phantom user queue. This task forces termination of the forking structure and then functions similarly to the LOGOUT command (i.e., the file directory and accounting data are written and eventually the BRS 112 is executed). If bit two of TTYASG is set, the "off" interrupt processing task is not placed on the phantom user queue, since the termination of this user was already performed by the LOGOUT or EXIT commands.

WERIS = -1 when teletype is inactive, 0 when the user is in the process of logging on, and user number after the user has logged on.

LINKING OF TELETYPE

It is possible for one teletype to accept linkage to another, break the linkage, or refuse to be linked. When the user logs on the system, bit 9 of TTYTBL, the accept message bit, is set.

LCW is initialized to 400000FT. Bit 0 indicates that the teletype is not linked and FT is a fictitious teletype number.

Once a teletype is linked, the associated teletype numbers are inserted into bits 18 through 23 of the LCW word of both teletypes. The teletype output routine checks bit 0 to see if NL is reset. If it is, the output character is placed into the TTYBUF for both teletypes.

In order for one teletype to link to another, bit 9 of TTYTBL (the accept message bit) must be set. Also, the teletype must not be in the 8-level mode, be already linked, or have just turned off the paper tape reader or be in the process of turning it back on. All these conditions will cause the link to be unsuccessful and the number of the teletype that attempted to link will be placed in bits 1 through 7 of LCW.

9. DEVICES AND TS PAGE BUFFERS

FILE STORAGE ON DISC

The physical records for the storage of files are divided into blocks of 256 words. The files use the disc in groups of 4 sectors of 64 words each.

The disc files used by this system consist of 8 to 32 physical discs, with each disc having a movable arm. The arms have 64 positions numbered 0 to 63 and each arm position on a disc can access 8, 192 words. Each arm position contains four pages (a page is 1/4 of an arm position) and one page contains 2, 048 words. It is possible to access four pages without moving an arm position.

For example, if the total number of arm positions is multiplied by the number of words per arm position, the total number of words per disc can be calculated (i.e., 8, 192 words x 64 arm positions equals 524, 288 words per disc).

The disc is divided into two major sections: system data and file storage. The disc map in Figure 13 illustrates the disc sections. Octal addresses 0, 40, 100, 140 are the beginning addresses for the four pages in a specified arm position. In this addressing scheme, each increment of one represents a sector of 64 words. Therefore, four addresses such as 0, 1, 2, and 3 would represent a physical record containing 256 words.

"User 400 FD" in arm position 0 at disc 0 represents the file directory of the individual's user number. "Acct @ 1 UAD" (arm position 1 at disc 4) is the user's account directory for for account @ 1.

The format for the disc address word is shown in Figure 14.

Figure 15 shows the flow necessary to retrieve a disc file. When a user logs on the system, the account number is used to calculate the disc address of the User Account Directory (UAD). The UAD contains a list of the user names associated with this account. Associated with each user name is the status of the user and his user number. The user number is assigned to a user by the operator. It can be used to calculate the disc address of the file directory associated with this user name.

A file directory (FD) is 128 words long and contains the ASCII name of a file and four control words that specify parameters peculiar to the file for all the files that pertain to this user. The number of files that can be represented in the file directory is a function of the length of the file names. If a user gives all of his files 3 character names, there would be room for about 24 files. One of the parameters that is associated with each file is the disc address of the index block.

Every file is written on the disc in data blocks of 255 words. The index block contains the disc address of the data blocks for the file. Currently the index block contains pointers (disc addresses mod 4) for 76 data blocks. Therefore, a file may be 255 x 76 or 19, 380 words in length.

When a file is written, the system collects 255 words in a buffer, searches a disc bit map for an available area, and writes the contents of the buffer on the disc. The disc address is stored into the index block. When the file has been completely written, the index block will then be written on the disc.

The formats for the UAD and the FD are shown in Appendix C. The layouts of the file buffer and of the index block buffer are shown in Figure 16.

Available storage in the file area of the disc is recorded in a bit table. A bit indicates that the corresponding block on the disc is free. The bit map is set every time the system is updated to agree with the files in the file directories. To set the bit map, BRS BE+5 is used, requiring index block pointer (mod 4) in A. When all files have been checked, the BRS is called with A set to -1, the new overflow pointer in B, and the accounting area address in X.

FILE BUFFERS

Every open file in the system with the exception of purely character-oriented files, such as the teletype, has a file buffer associated with it. The form of this buffer is shown in Figure 16. The index block is used only by disc files but is present in all cases. Each user has three buffers in his TS page. Therefore, any user can have a maximum of three files open. The Monitor always relabels the TS page into logical page 7.

Note that the amount of buffer space actually used is a function of the device attached to the file. In all cases, the two pointer words at the head of the buffer indicate the location of the data. The first word points to the beginning of the relevant data and is incremented as data is read from or inserted into the buffer. The second word points to the end of the data. On the output path, the second pointer is set to the physical record size that pertains to the device.

On the input path, this pointer is set by the routine that drives the device once it determines the number of words read. When the buffer pointers are equal, the buffer is either empty (input) or full (output).

The size of a TS page buffer is:

255	Data Words
2	Buffer pointers
6	Index Block parameters
128	Size of Index Block (only 76 words are used)
391	Words per buffer

Although only the disc files require an index block, every device that requires a TS page buffer is assigned a 391 word buffer.

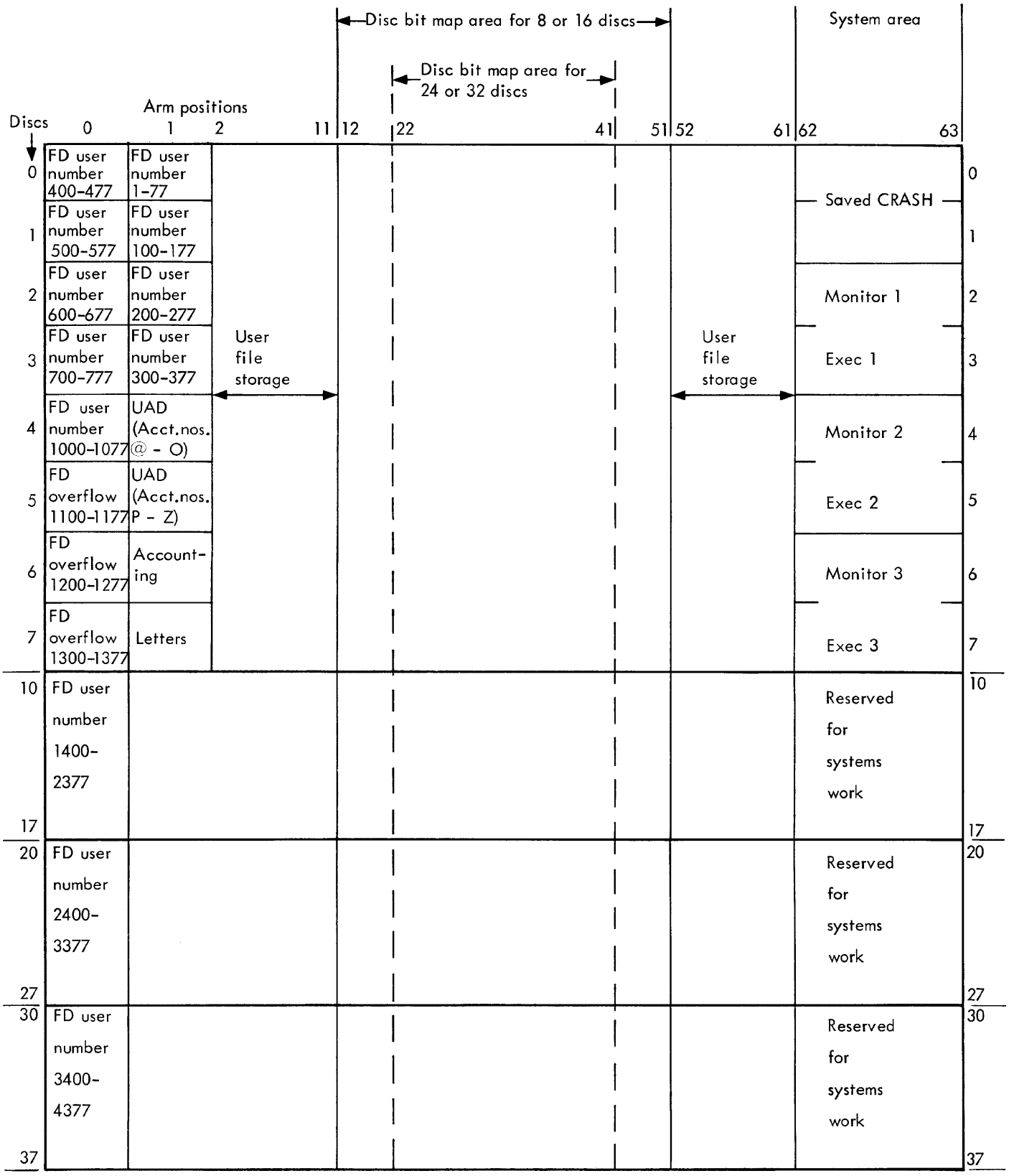
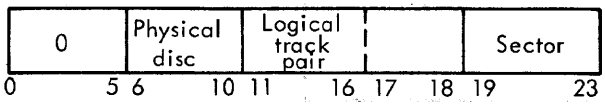


Figure 13. Typical Disc Layout

Sector in one position



where

Physical disc Bits 6-10 specify one of the 32 possible discs in the file unit.

Logical track pair Bits 11-18 specify one of the 256 track pairs on the disc. A track pair consists of one outer and one inner track.

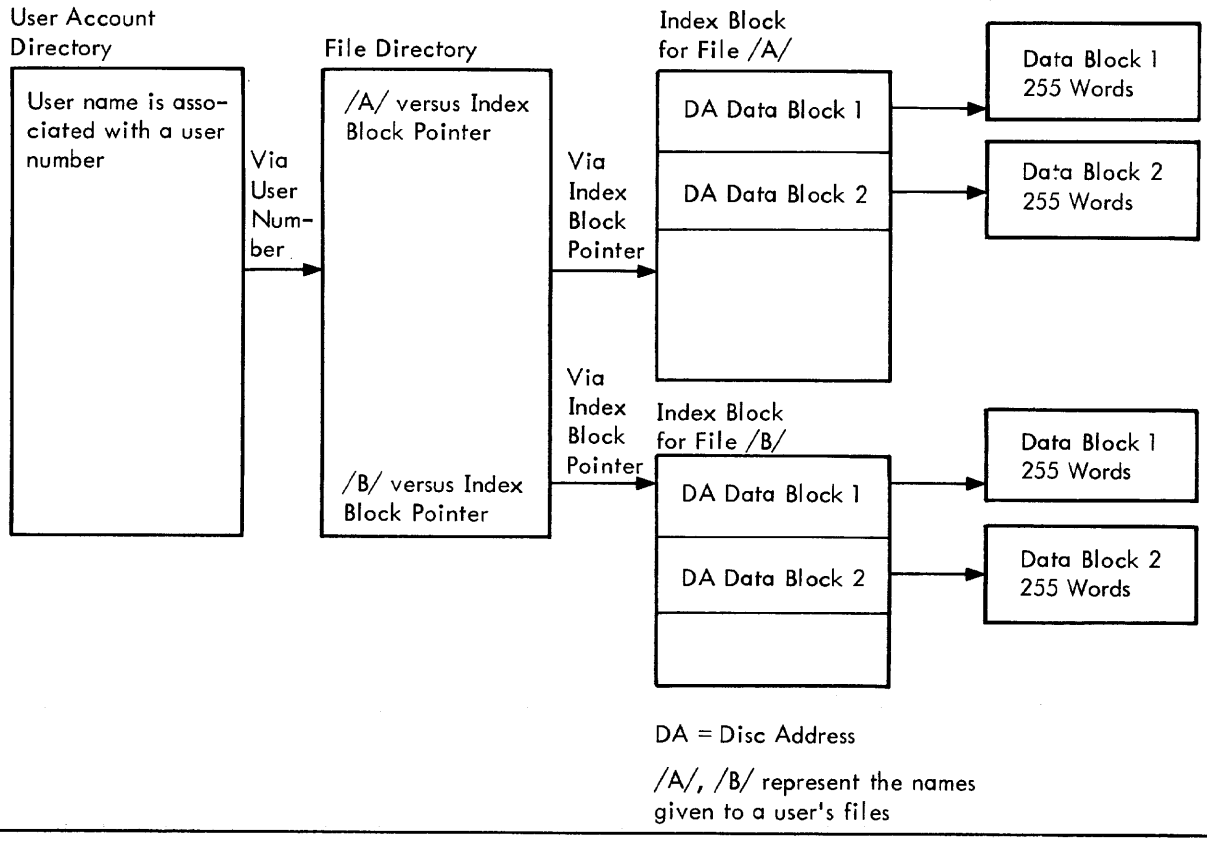
Bits 11-16 specify one of the 64 positions of the access arm.

Bits 17-18 specify one of four logical pairs that can be accessed without moving the arm.

Sector Bits 19-23 specify one of the 32 sectors in each logical track pair. Two disc revolutions are required to access the 32 sectors on one logical track pair.

Bits 17-23 specify the 128 sectors that can be accessed without moving the arm. Eight disc revolutions are required to access the entire sector string from one arm position.

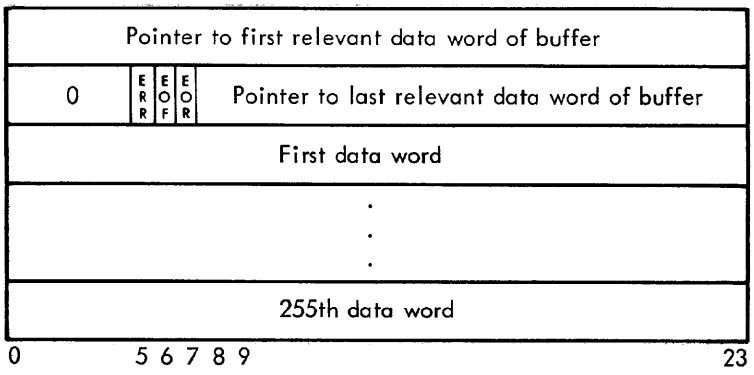
Figure 14. Disc Address Word



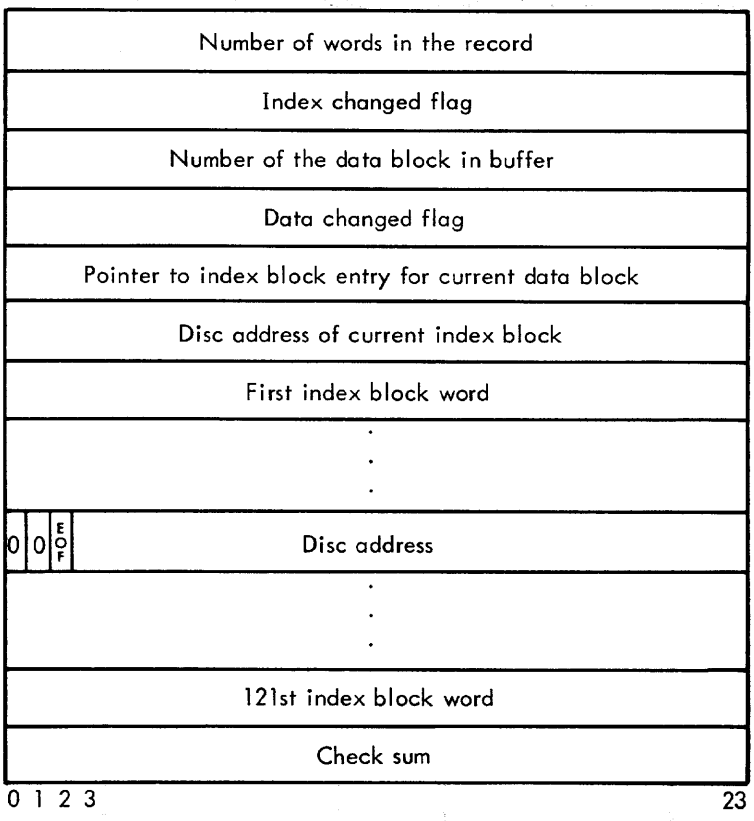
Note: See Appendix C for format of UAD and FD. See Figure 16 for Index Block format.

Figure 15. Flow Required to Access a Disc File

Layout of Data Block and Pointers



Layout of Index Block Buffer and Associated Pointers for a Disc File



where

Check sum "Exclusive or" of the first 121 words in the index block.

BIN When a disc data block is written, 256 words are output. The 256th word (contents of BIN) specify the number of words in the record.

BIC Initialized to -1. Incremented by an output operation to indicate that the index block must be output when the file is closed.

BDN Initialized to -1. Currently not used.

BDC Initialized to -1. Incremented by an output operation to indicate that the data block must be output to the disc.

BIP Contains BUFF + N where BUFF is the TS buffer address and N points to the entry in the index block that is being used.

BIA Storage for the disc address of the index block that is currently being used.

ERR, EOF, EOR Flags that are set by the device drivers, indicating error, end-of-file, and end-of-record, respectively. These flags are checked by the GPW (Get/Put Word) routine.

BXO Beginning of Index Block. All disc addresses are Mod 4 (truncated by 2 bits).

Figure 16. File Buffer

DEVICES

Every input/output device attached to the system has a device number. The numbers assigned to specific devices are given in Table 3. The various tables indexed by device number are described in this section. See Figure 17. The entries in these tables are specified by assembly parameters.

The major parameters of a device are:

1. the opening routine, which is responsible for the operation needed to attach it to a file.
2. the GPW routine, which performs character and word I/O.
3. the BIO routine, which performs block I/O.
4. the SEL routines, which perform the physical device I/O.

The minor parameters are:

1. maximum legal unit number.
2. physical record size (determining the proper setting of buffer pointers and interface control words for the channel), and the expected time for an operation.

Table 3. Device Numbers

Device	Number
Paper Tape Input	1
Paper Tape Output	2
BCD Tape Input	3
Magnetic Tape Input	4
Magnetic Tape Output	5
Hollerith Card Output	6
Binary Card Output	7
Disc File Input	8
Disc File Output	9
BCD Tape Output	10
High Speed Printer Output	11
Hollerith Card Input	12
Binary Card Input	13
Binary Magnetic Tape	14

When a file is opened, the device number is specified. The device number is used to index into the device tables. The device-dependent parameters are abstracted from the device tables and stored in the file control block (see Figure 18). Every open file in the system has a file control block associated with it.

Figure 17 shows tables indexed by device number. The DEV table specifies various characteristics of the device and the address of the entry point into the GPW (Get/Put Word) routine. This routine is used by the CIO, WIO, and BIO SYSPOPs. On the input path the GPW routine will take a word from the buffer in the TS page and place it in the A register. When the buffer becomes empty it will call the device driver (whose address is specified in SEL) and read the record size (specified in BUFS) into the TS page buffer. The opposite flow is taken for the output path. Word 0 of the DEV table contains the FD word (see Figure 18) of the currently active file.

The DIU table will contain an entry of -1 if the device is not in use. If the device is in use it will contain the file number of the file using the device. The disc will never have a meaningful entry in this table since it can be accessed by more than one user at one time. For magnetic tape it is not sufficient to indicate whether the "device" is busy since there may be several magnetic tape units. Therefore, the DIU entry for magnetic tape points to another array named ADIU. ADIU is indexed by tape unit number and contains the same information (-1 or file number) for each tape unit.

The address portion of the OPNDEV table contains the addresses of the routines that are called by BRS 1 to open a particular device. Bits 3-8 of this table contain the maximum amount of time (in 60 HZ clock ticks) that should occur once this device has started. Any routine that initiates action at a device will extract these bits and store them right justified into FTIME. The clock interrupt will decrement FTIME; if it becomes negative, action is taken to try to correct the fault. Once an interrupt from an I/O device is received, FTIME is set positive (37777777B).

SYSTEM DATA ON OUTER ARM POSITION OF DISC

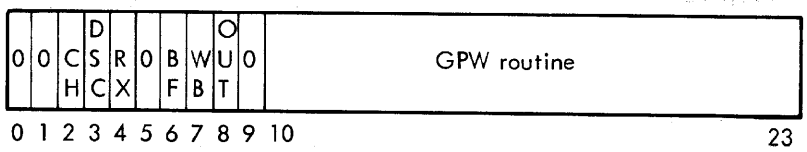
Arm positions 62 and 63 contain systems which are loaded by a utility routine named DSWAP. This routine dumps the first 32K of core on discs 0 and 1, then reads a new system into the first 16K of core. The disc from which the new system is read is determined by console switch settings. The Executive commands SYSDP and SYSLD can be conveniently used to access arm positions 62 and 63 of any disc.

Arm positions 0 and 1 contain the file directories, accounting information and data.

BRSs FOR DIRECT DISC ACCESS

There are four BRSs available to system level forks to read and write the system data on the disc. These are: BRS BE+1, BRS BE+2, BRS BE+9, and BRS BE+10. They require the core address in A and the disc address in B. In addition, BRS BE+1 and BRS BE+2 require the word count in X. BRS BE+9 and BRS BE+10 always read or write a page (2K) from or to the disc.

DEV word



CH a character oriented device. The WIO and BIO POPs cannot be used to access this device.

DSC indicates the device is the disc. This bit determines whether to relabel in the WPAGE or DISC files into page 6 of the Monitor's relabeling.

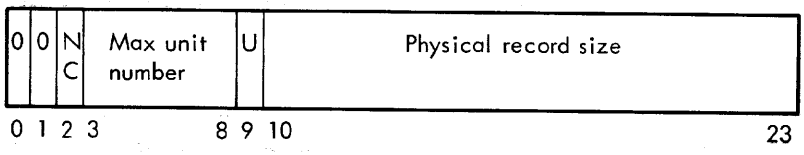
RX Random file. Currently not used

BF device uses one of the buffers in the TS page. Presently all devices have this bit set.

WD W buffer device. Presently this bit is not used.

OUT Output device.

BUFS
Buffer size

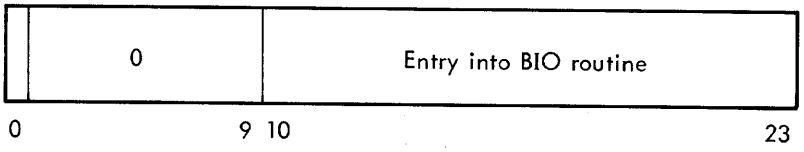


NC not common. Most devices can be accessed by only one file at a time. This is not true of the disc. This bit is set for the disc.

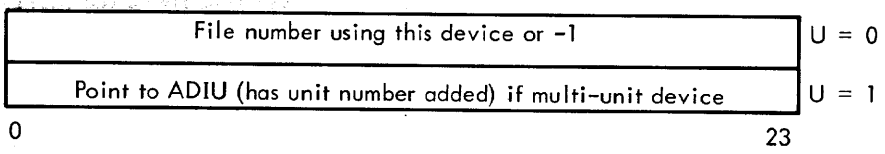
Unit the maximum unit number. Applies to multi-unit devices (such as magnetic tape). Presently magnetic tape units 0 and 1 are used.

U indicates this is a multi-unit device such as magnetic tape.

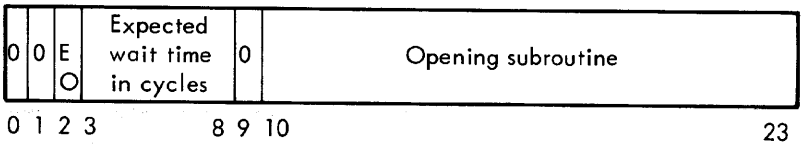
BDEV
Block I/O
Routine



DIU
device in
use

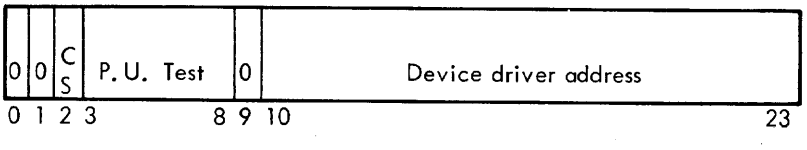


OPNDEV
Opening
Routine



EO Executive only allowed to open

SEL



CS check user's status

Figure 17. Tables Indexed by Device Number

10. SEQUENTIAL FILES

FILE NUMBERS

The term "file" refers to:

1. A disc file (i.e., a collection of data that has been named and output to the disc in blocks of 255 words).
2. A magnetic tape file (i.e., a collection of sequential blocked records that have been output to magnetic tape).
3. A physical device.

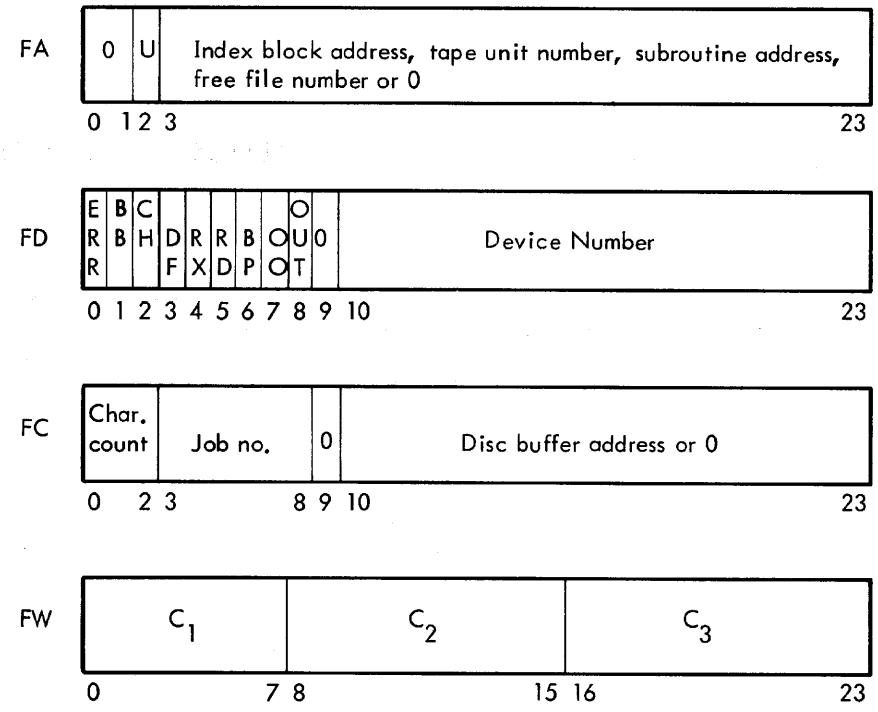
When a file is opened, the system will return a file number. The system may have up to 40 files opened. The user may have a maximum of 3 files that require a buffer open, as he is restricted by the number of available TS page buffers.

The file numbers range between 0 and 39 and are assigned in a somewhat random manner. As a user opens a file, he is assigned a free file number. When the file is closed, the number is returned to the free file number list which is kept in the FA table (see file control block).

Once the file number has been assigned, the user references the file by that number. Note that the I/O SYSPOPS (CIO, BIO, WIO) require the file number as an argument.

FILE CONTROL BLOCKS

Every open file in the system has a file control block associated with it. This block consists of four words shown in Figure 18.



U FA contains a free file number.

ERR Error occurred at the device.

BB When action at the device is initiated this bit is set. When the interrupt occurs this bit is reset. If a fork is dismissed for I/O, the PTEST word points to FD. This bit is examined to determine when to reactivate this fork.

CH Character oriented. See CH in DEV table.

DF Disc file. See DSC in DEV table.

RD Read only.

BP The file is using one of the buffers in the TS page.

OUT Output file.

Char. count -1 to 2.

C_n Word being packed or unpacked.

RX Random File - currently not used

OO Old output file. A previously existing file open for output.

Figure 18. File Control Block

This block is initialized when the file is opened. The file control block tables are indexed by file number. Note that the FD entry contains the device number of the physical device that is attached to the file. The system can use this number to reference the physical device tables. The entries in these tables provide the information that is necessary to completely define the file.

The FA array contains free file numbers. The word FFLST contains the next free file number. The corresponding entry in FA will contain the next available file number or 0 if there are no free files. The free file list is initialized in TTYSET. When a file is in use, FA will contain the disc address of the index block for a disc file or the magnetic tape logical unit number for a magnetic tape file. FA is not used for any other device.

The FC array contains the address of the TS page buffer that is being used by this file and the job number of the user. The job number is set to 77B for a permanently opened file. The character count has significance if the CIO is being used to access a word-oriented device. On the input path, CIO will fetch a word from the buffer, store it into the FW array, and return a character to the calling program. Subsequent CIO calls will retrieve a character from FW. When all of the characters in FW have been sent, the next CIO call will again fetch a word from the buffer. The opposite flow occurs on the output path. The character count in the FC word has the following significance:

Bit Configuration	Input
111	There are no characters in FW. Call GPW to get a word.
001	There are 2 characters in FW.
000	There is 1 character in FW.
	Output
010	There are no characters in FW.
001	There is 1 character in FW.
000	There are 2 characters in FW.
111	There are 3 characters in FW. Call GPW and place the FW word into the buffer and place this character into FW.

OPENING AND CLOSING FILES

In order to manipulate a file, it must first be opened. A user can open a file by executing one of the opening BRSs (see BRS 15, 16, 18, 19, 48, 60). These BRSs obtain all of the necessary parameters from the user's file directory and then execute the BRS 1.

Opening a file accomplishes the following:

1. A file number is assigned.
2. An available TS page buffer is dedicated to this file.
3. The buffer pointers are initialized.
4. Where applicable, a check is made to determine if the device is already in use.
5. A call is made to the opening routine that is associated with this device. See OPNDEV table.
6. The file control block is initialized.

The operations performed by the device opening routines depend on the complexity of the device. For many devices, the routine simply issues a device ready test. The disc opening routine must obtain and initialize the index block for the file. All of the BRSs which open files will return the file number. Once the file number is available, the user can execute any of the I/O SYSPOPS to input/output data to the file.

A file must be closed when its processing has been completed. This is accomplished by executing a BRS 2 with the file number in the A register. BRS 2 is available to both user and Executive programs. To close all his open files, the user may execute a BRS 17. Closing the file releases it for other uses. The file number and the buffer are also released.

ACCESSING THE TELETYPE AS A FILE

The teletype can be a accessed by using the CIO SYSPOP. However, the teletypes do not require a TS page buffer. Each teletype on the system has a dedicated buffer which is core resident. Chapter 8 describes the teletype buffers. When the teletype is accessed as a file, it does not have to be opened since the teletype is a permanently opened file.

PERMANENTLY OPEN FILES

The system has the following built-in sequential files with fixed file numbers:

0	controlling teletype input
1	controlling teletype output
2	nothing (discard all output)
1000+n	input from teletype n
2000+n	output to teletype n

SEQUENTIAL DISC FILES

A sequential file has a structure very similar to that of an ordinary magnetic-tape file. It consists of a sequence of logical records of arbitrary length and number. Disc sequential files, are however, considerably more flexible than corresponding files on tape, because logical records may be inserted and deleted in arbitrary positions and increased or decreased in length.

The system opens a disc file by the following sequence of instructions:

```
LDX     device number, 8 (input) or 9 (output)
LDA     Address of the index block (mod 4)†
BRS     1
```

If a new output file is being opened, the A register should be 0 since an index block does not exist.

If BRS fails to skip, it returns in A the following:

-1 Device already in use. For the disc, produced by an attempt to open a file for output twice.

-2 too many files open – no file control blocks or no buffers available.

-3 no disc space left. This inhibits opening of output files only.

BRS 1 returns the file number in the A register and the disc accress (mod 4) of the index block in the X register.

A file that is open for output cannot be opened again for either input or output and a file that is open for input cannot be opened for output. However, a file may be opened for input any number of times.

The disc opening routine will read the index block into the buffer and initialize the index block pointers. If this is a new file, the disc bit map will be checked and an available space for the index block is obtained and the index block area (BXO and following) in the buffer will be reset to 0.

When the first I/O SYSPOP is encountered on the input path, the index block will be referenced and the data block read. The 256th word read will cause the next data block to be read. This process is continued until an index block entry of 0 is obtained (i.e., all of the data blocks have been read). The EOF or EOR flags will be set in the second word of the buffer.

When a file that already exists is used for output, the first attempt to write the data block will cause all of the old data blocks, as specified in the index block, to be released to the bit map. An available disc block will be obtained, the data block written, and the address of the data block will be stored into the first index block entry. Another disc block is also obtained in preparation for the next data block write. This disc address is stored into the next position (obtained by incrementing BIP) of the index block.

Subsequent requests to write the data block will use the disc address that is pointed to by BIP, increment BIP, and obtain another available disc block. This process will continue until the index block is full or the file is closed.

†mod 4 means the lowest 2 bits are truncated.

If a new output file is being written the path is similar to the one described above except that there is no need to release the old data blocks.

When an output file is closed, the remaining words in the buffer are written on the disc. The EOF flag is set in the index block entry that points to the last data block. The index block is then sent to the disc. The disc address of the index block was stored in BIA when the file was opened.

I/O SYSPOPS

Three kinds of input/output may be done with sequential files. They are: character input/output (CIO), word input/output (WIO) and block input/output (BIO). Each of these SYSPOPS can perform input or output since the file must be specified as an input or an output file when it is opened.

To input a single character to the A register or output it from the A register, the instruction

```
CIO     file number
```

is executed. During input, an end of record will set bits 0 and 8; an end of file condition will set bits 0 and 7 in the file number. These are called flag bits. An end of record will return a 134B character; an end of file, a 137B character. If interrupt 4 is armed, it will occur. The end of record condition occurs on the next input operation after the last character has been input. The end of file condition occurs on the next operation after the end of file, which signals the last record of the file. The user may generate an end of record while writing a file using the control operation to be described. An error condition sets bits 0 and 6 in the file number.

To input a word to the A register or output it from the A register,

```
WIO     file number
```

is executed. An end of file condition returns a word of three 137 characters. Mixing word and character operations is not recommended.

To input a block of words to memory or output them from memory, the instructions

```
LDX     first word address
LDA     number of words
BIO     file number
```

should be executed. The contents of A, B and X will be destroyed. The A register at the end of the operation contains the first memory location not read into or out of.

If the operation causes any of the flag bits to be set, it is terminated at that point and the instruction fails to skip. If the operation is completed successfully it skips. Note that a BIO cannot set both the EOR and the EOF bits.

The flag bits of the file number are set by the system whenever end-of-record (0 and 8) or end-of-file (0 and 7) is encountered and cleared on any input/output operation in which neither of these conditions occurs. Bit 0 is set on any unusual condition. In the case of a BIO the A register at the end of the operation indicates the first memory location not read into or out of. For any input operation, the end of record bit (bit 8) of the file number may be set. An output operation never sets either of these bits. Bits 0 and 6 of the file number may be set on an error condition. Whenever any flag bit is set as a result of an input/output operation in a fork, interrupt 4 will occur if armed.

A program may delete all the information in a disc file by executing the instructions:

```
LDA     file number
BRS     66
```

Putting the file number of a sequential file in A and executing BRS 113 will cause the file to be scanned to find the total number of data words. The number of data words is added to X.

A new disc file with a new index block can be created by BRS 1 with an index block number of 0 in A. The file number is returned in A and the index block number in X. The read-only bit may be set (bit 0 of A) and

```
BRS 67
```

returns the index block with address to available storage in A. An executive fork may read an index block into core with

```
BRS 87
```

which obtains the address of the block from A, and X will contain the address of the first word in core into which the block is to be read.

OTHER SEQUENTIAL FILES

In addition to disc sequential files, the user has other kinds of sequential files available to him. The system opens these files by the following sequence of instructions:

```
LDX     device number
LDB     RECL (BCD tape output only)
LDA     unit number
BRS     1
```

RECL is positive for 80 characters and negative for 132 characters.

The device number is put into X. The unit number, if any, is put into A. The file number for the resulting open file is returned in A. If BRS 1 fails, it returns an error condition in A. Three error conditions apply to magnetic tape only:

0 Tape not ready

1 Tape file protected (output only)

2 Tape reserved

BRS 1 is inverted by BRS 110, which takes a file number in A and returns the corresponding device number in X and unit number in A.

These files may also be closed and read or written in the same manner as sequential disc files.

CTRL = 1 (end of record)

is available for physical sequential files 2, 5, 10, and 14 (paper tape and magnetic tape output). Other controls available for magnetic tape files only are listed in Table 4.

Table 4. File Control for Magnetic Tape

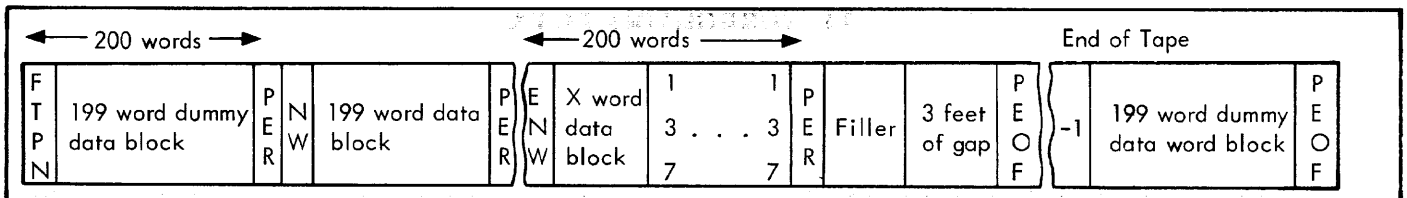
Operational Control No.	Magnetic Tape File Control
1	Write end of record
2	Backspace block
3	Forward space file
4	Backspace file
5	Write three inches blank tape
6	Rewind
7	Write end of file
8	Erase long gap

These controls may be executed only by Executive tape programs.

An Executive program may allocate a tape unit to itself by putting the unit number in A and executing BRS 118, which skips if the tape is not attached to some other job. BRS 119 releases such a tape.

The format for magnetic tape (devices 4 and 5) is shown in Figure 19. Note that the records are 200 words in length. For compatability with earlier versions of the system, magnetic tapes (devices 4, 5) have three dummy records after the load point. The records are placed on the tape by the operator's NEWTAPE program.

The format for BCDTAPES (devices 3 and 10) is shown in Figure 20.



FTPN Tape file position number (e.g., the fifth physical file on the tape will have FTPN=5). Note that the first record of every file contains FTPN as the first word and 199 dummy words.

PER Physical end of record.

NW First word of a record contains the number of words in the record.

ENW Same format as NW only bit 0 is set to indicate this is the last record of the file. The last record contains X words of data. The remainder of the record is padded with 137B characters.

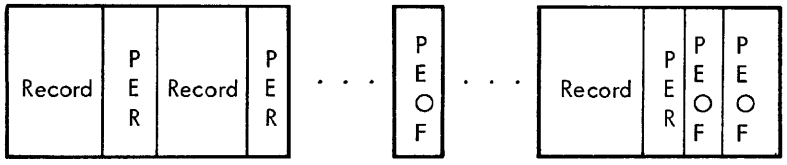
PEOF Physical end of file.

Filler A record (0.5% of the original file length) of 137B characters.

3 Feet Gap Allowance for expansion of the file.

End of Tape The last file on the tape is followed by a dummy record with NW = -1.

Figure 19. Format for Magnetic Tape Files



PER Physical end of record.

PEOF Physical end of file. The last file on the tape is followed by two PEOFs.

Record Length 3 to 132 characters for input and 80 or 132 characters for output.

NOTE: 79, 80, or 81 character lengths will be assumed to be 80 character transformations (card images).

Figure 20. Format of BCD Magnetic Tape

It is possible for magnetic tape and card reader files to set the error bit in the file number. The first I/O instruction after an error condition will read the first word of the next record; the remainder of the record causing the error is ignored. The magnetic tape routines take the usual corrective procedures (i.e., reread or rewrite) when they see hardware error flags, and the routines signal errors to the program only as a last resort.

In order to make the card reader and BCDTAPE look more like other files in the system, the following transformations are made by the system on card input:

1. More than two blanks are converted to a 135 character followed by a character giving the number of blanks. The teletype output routines will decode this sequence correctly.
2. Trailing blanks are not transmitted to the program.
3. The character 155 (carriage return) is added to the end of each transformation.

The result of this configuration is that the string of characters obtained by reading in a card deck or a BCDTAPE file may be output without change to a teletype and will result in a correct listing of the deck.

Whenever a card reader error (feed check or validity check) occurs, the program is dismissed until the reader is ready.

The EOF light is sensed as an end of file at all times.

Because of critical timing requirements, the card punch should be operated when there is but one user in the system, i.e., the operator.

Because of the interactive nature of the system peripherals, device speeds will decrease as the number of users in the system increases.

11. SUBROUTINE FILES

In addition to the previously mentioned operations for performing input-output through physical files, a facility is provided within the system for making a subroutine call appear to be an input-output request. This facility makes it possible to write a program which does input-output from a file which causes further processing to be performed before the actual input-output is done. This is accomplished by simply changing the file from a physical to a subroutine file. A subroutine file is opened by executing the instructions:

```
LDX     parameter word
BRS     1
```

The instruction never skips. The operation code field of the parameter word indicates the characteristics of the file. It may be one of the following:

110 00000 (octal)	Character input subroutine
111 00000 (octal)	Character output subroutine
010 00000 (octal)	Word input subroutine
011 00000 (octal)	Word output subroutine

I/O to the file may be done with CIO or WIO, regardless of whether it is a word-oriented or a character-oriented subroutine. The system will take care of necessary packing and unpacking of characters. BIO is also acceptable.

The opening of a subroutine file simply creates a file control block and returns a file number in the A register. When an I/O operation on the file is performed, the subroutine is called. This is done by simulating an SBRM to the location given in the word following the BRS 1 which opened the file. The contents of the B and X registers are transmitted from the I/O SYSPOP to the subroutine unchanged. The contents of the A register may be changed by the packing and unpacking operations necessary to convert from character-oriented to word-oriented operations or vice versa. The I/O subroutine may do an arbitrary amount of computation and may call on any number of other I/O devices or other I/O subroutines. A subroutine file should not call itself recursively.

When the subroutine is ready to return, it executes BRS 41. This operation replaces the SBRR which would normally be used to return from a subroutine call. The contents of B and X when the BRS 41 is executed are transmitted unchanged back to the calling program. The contents of A may be altered by packing and unpacking operations. A subroutine file is closed with a BRS 2.

In order to implement BRS 41, it is necessary to know which I/O subroutine is open. This information is kept in 6 bits of the PAC table. These 6 bits are transferred into the operation code field of the return address when an I/O subroutine is called, and are retrieved when the BRS 41 is executed.

12. EXECUTIVE TREATMENT OF FILES

GENERAL DESCRIPTION

The user's sole access to files is through the Executive. The Executive provides a connection between a symbolic name for a file created by the user, and the file numbers the user must have to execute input/output operations. This connection is established through the file directory. Supplementary to this function is the need to prevent the user from damaging or destroying other users' files.

The first part of this section describes the file naming system as it appears to the user; the second part describes the Executive tables that implement various features.

A user may give his files arbitrary names containing any characters other than ' or /, because the names of disc files must be surrounded by /, and the names of tape files by '.

When a user types a file name not enclosed within slashes or quotes, he need only type enough characters of the name to uniquely define it. If the user starts an output file name with a quote or slash, he must type the entire name. If it is an output file name and not already in his file directory, a new file will be created. In any other context, a name not in the file directory is in error.

When an output file name is being typed, the system, after determining the name, will type out either OLD FILE or NEW FILE and await a confirmation that the name has been given correctly. If the user types either a line feed or a carriage return, the name will be regarded as correct. Any other character will be regarded as an indication that the name was incorrect. This procedure is designed to make it more difficult for the user to destroy old files or create new ones inadvertently.

When a user gives a new (slashed) output file name to the system, this creates a new entry in the file directory and a new index block on the disc.

The user is allowed to reference files belonging to other users if the file name to be referenced contains at least one control character or an @. He does this by typing that user's account number and name, enclosed in parentheses, before the file name. Thus, to get at file/@PROGRAM/belonging to user JONES, he types

(A1JONES)/@PROGRAM/

In this way Jones can control the extent to which other users access his files.

Files in a public file directory may be accessed by typing the file name in quotes

"PROGRAM".

It is possible for a user to rename his files by typing for example

RENAME /PROGRAM/ AS /ROUTINE/

The rename logic protects the user against creating file names that conflict with existing file names or with the file type.

PHYSICAL DEVICES

Some of the physical devices can be accessed as files. There are six file names built into the system:

BDCTAPE CARDS PAPER TAPE PRINTER	the user must have peripheral status to use these files
TELETYPE NOTHING	Available to all users

These names may be used at any time. If the device referred to is not available because it is attached to some other user, a suitable error message will be generated. Paper tape output files opened by giving this name to the Executive will have the type of the file punched as the first word. Similarly, paper tape input files opened by giving this name to the Executive will read the first word as the file type.

There are four standard file types:

1. File written by Executive save command (sequential)
2. General binary file (sequential)
3. Symbolic file (sequential)
4. Dump file (sequential)

STRING POINTERS

Many of the BRSs that deal with file manipulation require string pointers as arguments. A string pointer is a character address found by multiplying the word address by three and adding 0, 1, or 2. The string pointer P1 points to the character before the beginning of the file name. The pointer P2 points to the last character of the name.

TAP assembles string pointers as follows for string pointers P1 and P2:

```
P1      DATA    (R) Z-1
P2      DATA    (R) Z+2
Z       ASC     '/T/'
```

Suppose that Z was at location 1000B. Then P1 would have the value 2777B and P2 would equal 3002B. See "Special Relocation" section of TAP reference manual for discussion of (R) in the DATA operators above.

THEORY OF HASHING

"Hashing" is a technique of having the system assign a "number" to a character string to avoid the need for a character-by-character search through an entire list of character strings. This number is calculated by adding the number of characters in the string and the ASCII code for the first and last three characters, and dividing the sum by a constant. This algorithm associates a number with a character string, and the information associated with the string can now be filed in a table indexed by the "hash number".

The number will not be unique. There are three words of data associated with each hash table entry. After a user's file directory has been read from the disc, the pertinent information can be abstracted from the file directory and stored into the hash table. If two strings produce the same hash number (HN), the second entry is placed in the first available space preceding the calculated hash number (lower numbered core location). All unused hash table locations contain 0.

Assume the following hash numbers for a user's files:

File Name	Hash Number
SMITHJ	M
SMITHM	N
FILANAM	P
FILBNAM	P

If FILANAM is inserted before FILBNAM, the hash table will have the format shown in Figure 21.

The first two of the three data words contain string pointers to where the character string for the file name is stored.

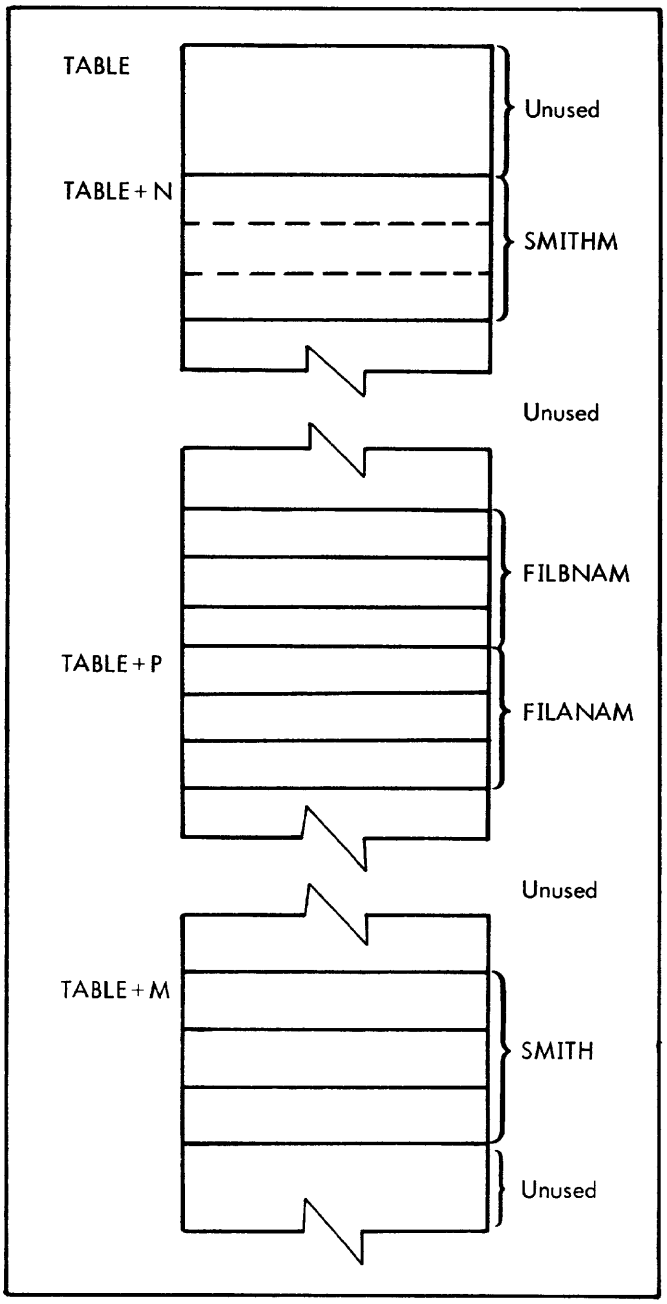


Figure 21. Hash Table

The third is the "hash value word" which is a data word pertaining to the particular entry. When the system wishes to retrieve file information, it must be able to verify that it has found the correct file.

The system can retrieve information about a specific file by:

1. calculating the hash number;
2. verifying that the string names match, or if not, searching the table backward and cycling around until all the entries have been searched; and
3. returning, if successful, the word address of TABLE+HN in the B register.

If the search is successful, the following coding will retrieve the three data words associated with the file:

```
CBX
LDA     0,2     First Data Word
LDA     1,2     Second Data Word
LDA     2,2     Third Data Word
```

940 HASHING ALGORITHM

W1 – $\boxed{c_1\ c_2\ c_c}$ – First 3 characters of the string

W2 – $\boxed{c_j\ c_k\ c_1}$ – Last 3 characters of the string

N Number of characters in the string

L Length (number of words) in the hash table

[] Integer divide (ignore remainder)

$$X = 8*N + W1_{0-11} + W1_{12-23} + W2_{0-11} + W2_{11-23}$$

$$Y = \left[\frac{X}{L}\right] = \text{Quotient} + R\ (\text{remainder})$$

$$HN = \frac{R}{L} * 3$$

R is some value between 0 and L-1. HN is an even multiple of 3 since the hash table is grouped by three.

If the character string contains 3 characters or less, the algorithm is slightly different. The missing characters in W1 and W2 are effectively reset to 0.

THE HASHING TABLE

There are three hashing tables used by the 940 software: the file directory, the commands, and the subsystem-names hash tables.

A hashing table consists of four distinct parts (see Figure 22, hashing tables for the file directory):

1. Control table – contains pointers into the other parts of the table.
2. Hash table – contains string pointer into the string storage area and other information pertinent to the entry.
3. Corresponding table – indexed parallel to the hash table. It contains three words of additional information pertaining to the entry. The commands hashing does not require a corresponding table.
4. String storage – stores the character string for each entry. The hash table entry contains pointers into this area.

The four parts of the table need not occupy contiguous core locations.

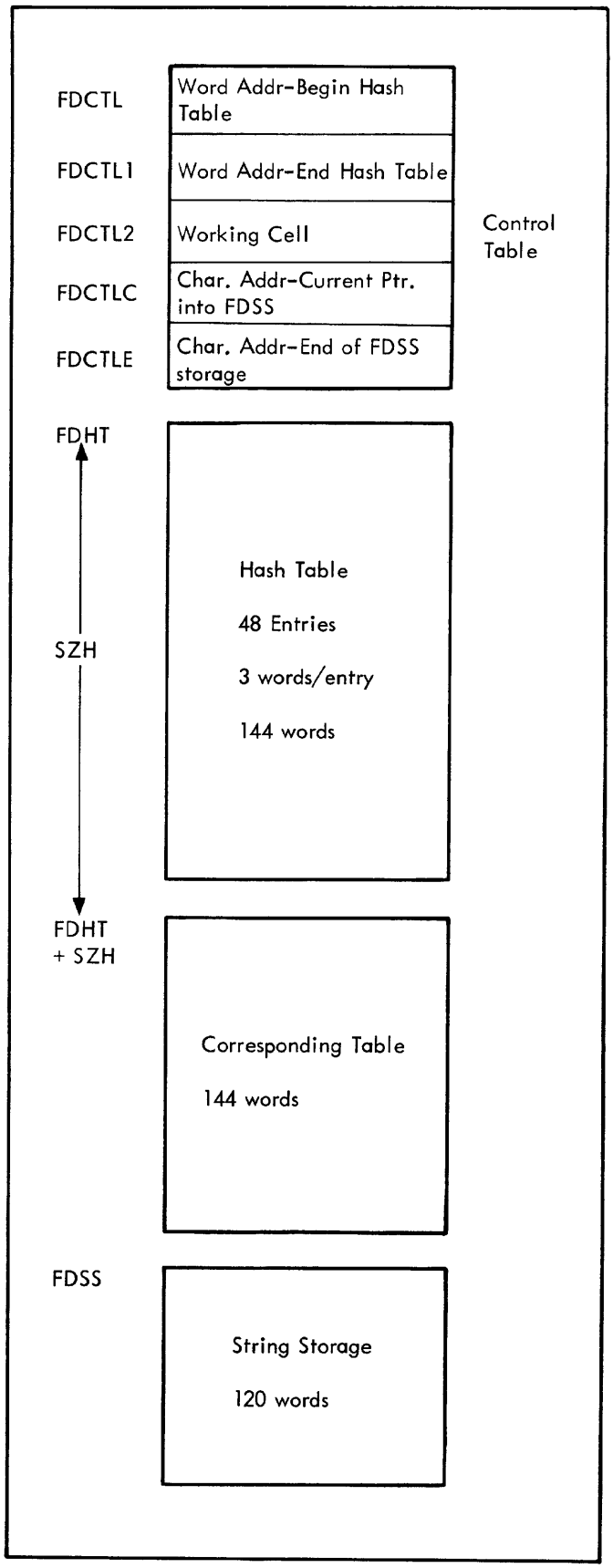


Figure 22. File Directory Hash Table

FILE DIRECTORY HASH TABLE

When a user logs on, his file directory is read in from the disc. This file directory is then stored in a hash table in the user's TS page. Figure 23 shows the format for the 3-word hash-table entries for magnetic tape, physical device and disc files. The three words consist of information abstracted from the file directory and pointers (character addresses) to the string storage table (FDSS). As an entry is inserted into the hash table (into FDHT+HN, where HN is the hash number), the character string (ASCII file name) is stored into FDSS in the character address pointed to by FDCTLC. Therefore, the file names are put into FDSS chronologically. FDCTLC must then be updated by the number of characters in the file name. Word 0 of the hash table entry contains the pointer (character address) into FDSS to the beginning of the character string. Word 1 contains the pointer to the end of the string. Word 2 of the hash table entry for any hash table is referred to as the "hash value word". Note that for disc file the hash value is the index block pointer.

When the user's file directory hash table is created, the names of the physical devices that can be accessed as files are always inserted into the table. Note the hash table entry for a physical device.

FILE DIRECTORY CORRESPONDING TABLE

Three more data words, called the corresponding table entry and also indexed by hash number, are associated with every file. These three words consist of data abstracted from the file directory for storage in FDHT+SZH+HN and following, where SZH is the difference between the beginning of the hash table and the beginning of the corresponding table.

The creation date of the file is set to the current data each time it is opened as an output file. The field "No. of Accesses" is incremented each time the file is opened for input or output.

BRS 5 AND 6

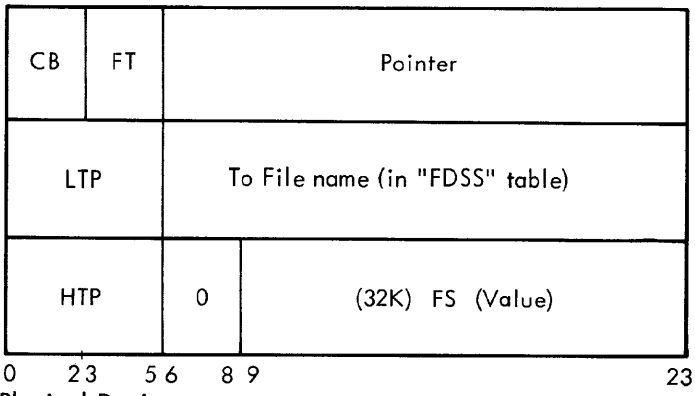
BRS 5 is used for two operations:

1. to look up an entry for a particular character string already in the hash table.

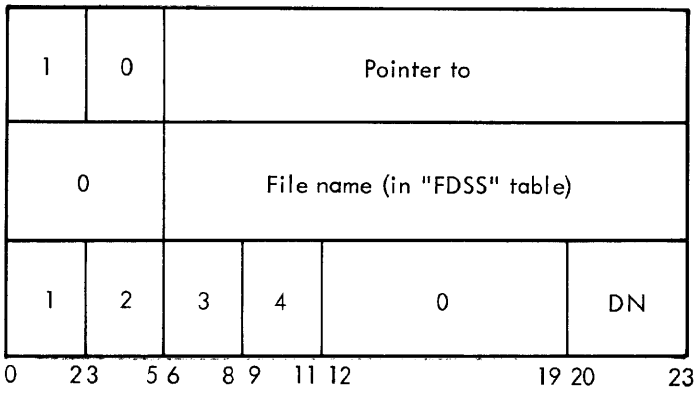
2. to find where in the hash table a new character string should be inserted.

The input for the first operation consists of pointers to the character string and to the address of the control table of the hash table that is to be searched. The output is the address of FDHT+HN (where FDHT is the address of the hash table) in B, and the value word (i.e., the 3rd word of the hash table entry) in A.

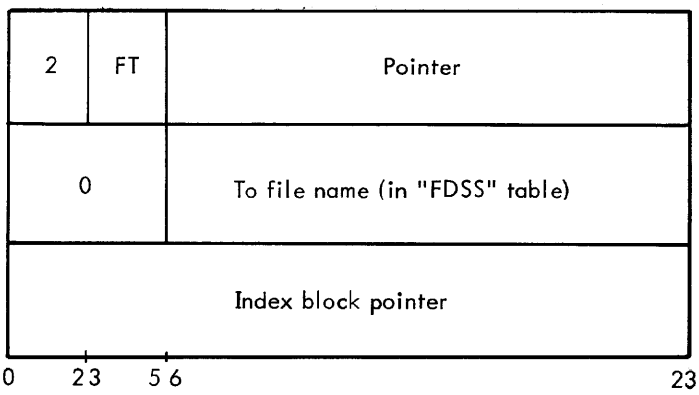
Magnetic Tape



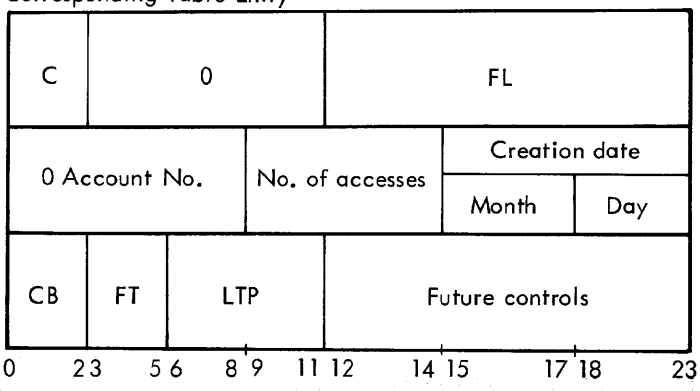
Physical Device



Disc File



Corresponding Table Entry



FT	File type
LTP	Low order tape position
HTP	High order tape position
FS	Tape file size
FL	File length for disc files
C	Change in file length
CB	File control bits 0 = Tape file 2 = Disc file
DN	Executive device number for the six built-in files 1 = Paper Tape 2 = Teletype 3 = Nothing 4 = Printer 5 = BCD tape 6 = Binary tape 7 = Cards

Figure 23. Hash Table Entry and Corresponding Table Entry for File Directory

Example:

Get the index block pointer for a disc file named /JOHN/. Also, increment the number of accesses which is stored in the 3rd word of the corresponding table.

```
                .
                .
NAME    ASC     '/JOHN/'
                .
PTRB    DATA    (R) NAME-1
PTRE    DATA    (R) NAME+5
                .
        LDA     PTRB
        LDB     PTRE
        LDX     =FDCTL      Address of control
                            table for File Direc-
                            tory Hash table
        BRS     5
        BRU     BAD         String not in table
        STA     IBP         Value word is index
                            block ptr.
        CBX
        LDA     SZH+1,2     Get 2nd word of
                            corresponding table.
        CLB
        LRSH    9
        ADD     =1
        LSH     9
        STA     SZH+1,2
                .
                .
```

For the second operation – to find where in the hash table a new character string should be inserted – the BRS 5 will perform the hashing algorithm on the string and get a hash number. If the entry is already in use, the hash table will be searched backward until an available slot is found. The value FDHT+HN (address for insertion into the hash table) will be stored into the third word (working cell) of the hash control table. If there are no available entries in the hash table, the working cell will contain -1. BRS 6 is used to insert the string pointers into words 0 and 1 of the hash table once BRS 5 has determined where the hash table entry should be placed. The BRS 6 does not move the string into FDSS.

Example:

Insert a new file name into a hash table. Refer to Figure 21.

```
                .
                .
NAME    ASC     '/NEW/'
PTRB    DATA    (R) NEW-1
PTRE    DATA    (R) NEW+4
                .
        LDP     PTRB
        LDX     =FDCTL
        BRS     5
        BRU     $+2
        BRU     BAD         Name should not be
                            in table.
        SKN     FDCTL2      Working cell = -1?
        BRU     $+2
        BRU     BAD         Hash table is full
        BRS     6           Make entry
        CBX
        BRM     XPHT        This routine inserts
                            words into the hash
                            and corresponding
                            tables.
        LDP     PTRB
        LDX     FDCTLC      FDCTLC points to
                            next available space
                            in FDSS
        BRM     INSERT      This subroutine will
                            insert the character
                            string into FDSS
        LDA     FDCTLC
        ADD     =5
        STA     FDCTLC      Update pointer since
                            5 characters in
                            /NEW/
```

COMMANDS HASH TABLE

The entry for the Commands Hash Table is shown in Figure 24.

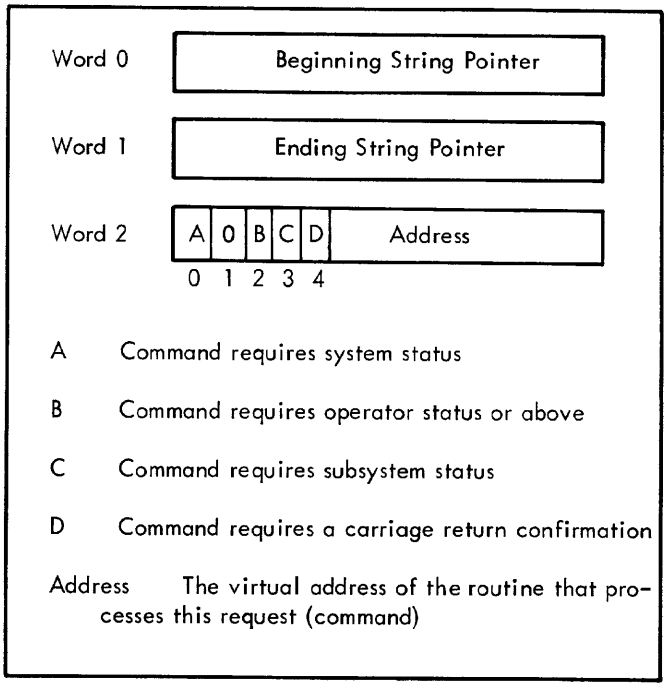


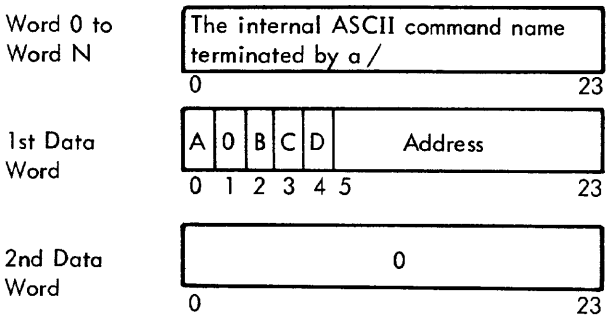
Figure 24. Commands Hash Table Entry

The commands and subsystem hashing tables are initialized after the Executive has been assembled. This is accomplished by loading the new version of the Executive and also loading a file called INTLE. The INTLE file includes a routine named SYSIN and two arrays named CIT and SIT which contain the information necessary to form the commands and subsystems hashing tables. The SYSIN routine uses the BRS 5 and 6 to insert the information provided by the CIT and SIT arrays into hashing tables which are in the Executive.

A macro named IT is used to generate the CIT array. A macro named SUBIT forms the SIT array.

IT forms the ASCII string for each command, followed by several data words. The exact number of data words formed depends on the number of parameters in the macro call. Presently, there are at most 2 arguments which will produce 2 data words.

The IT macro provides the following data words for each command:



See Figure 24 for description of 1st Data Word.

Note that bit 9 of the 2nd data word is 0. This indicates that there are no data words generated for the SYSIN routine to insert into the corresponding table.

SUBSYSTEM HASH AND CORRESPONDING TABLE

The Subsystem Hash Table and Corresponding Table entries are shown in Figure 25.

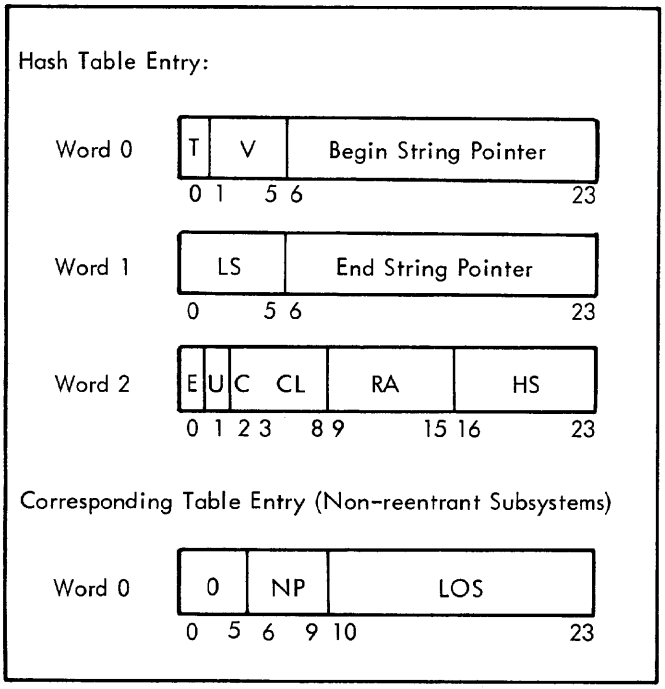
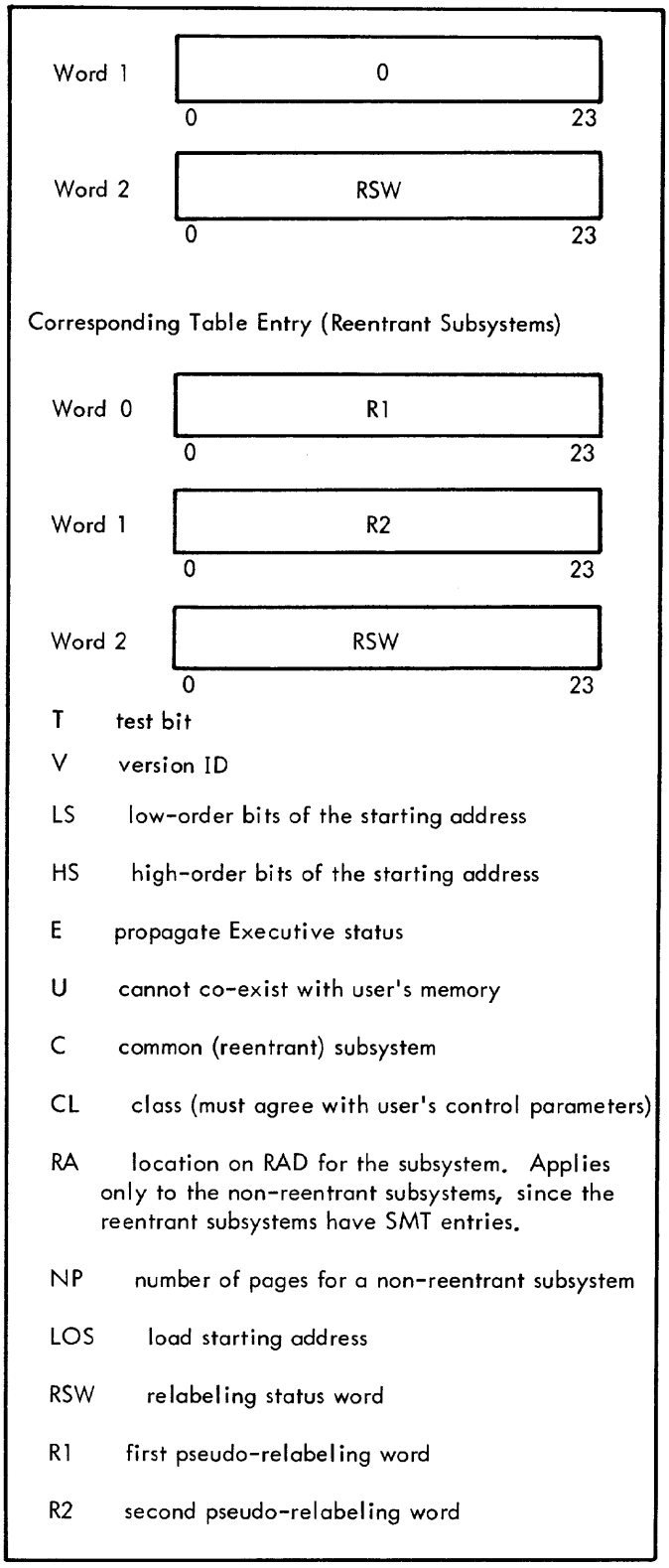


Figure 25. Subsystem Hash Table and Corresponding Table Entries



T test bit

V version ID

LS low-order bits of the starting address

HS high-order bits of the starting address

E propagate Executive status

U cannot co-exist with user's memory

C common (reentrant) subsystem

CL class (must agree with user's control parameters)

RA location on RAD for the subsystem. Applies only to the non-reentrant subsystems, since the reentrant subsystems have SMT entries.

NP number of pages for a non-reentrant subsystem

LOS load starting address

RSW relabeling status word

R1 first pseudo-relabeling word

R2 second pseudo-relabeling word

Figure 25. Subsystem Hash Table and Corresponding Table Entries (cont.)

The subsystem hash table is in page 1 of CMNDS and is labeled SYSHT. The corresponding table is labeled SYSCT.

The SUBIT macro forms the ASCII string for the name of the subsystem followed by five data words. Subroutine SYSIN

inserts the five data words into the hash and corresponding tables.

The SUBIT macro produces the data words shown in Figure 26 for each subsystem.

USER BRSs FOR FILE MANIPULATION

A program may open a disc file and obtain a file number by executing BRS 15 and BRS 16 (input) or BRS 18 and BRS 19 (output). BRS 15 and BRS 18 require the file name from the teletype. If the name is known to the program, BRS 15 and 18 may be replaced by BRS 48. These BRSs are used in the following way.

```
LDA     command file number
BRS 15 (or BRS 18)
EXCEPTION RETURN
NORMAL RETURN
```

The normal return leaves a file directory pointer, i.e., the location of the first word of the hash table entry (FDHT+HN) in A, and BRS 18 leaves the character typed after OLD FILE or NEW FILE in B. If no character was read, B contains a -1. The X register is modified.

```
LDA     file directory pointer
LDX     file type (BRS 19 only)
BRS 16 (or BRS 19)
EXCEPTION RETURN
NORMAL RETURN
```

The normal return leaves a file number in A, and BRS 16 leaves the file type in B. X is modified.

BRS 48 or 60 may be substituted for BRS 15 or 18. BRS 48 is used if the name is in the file directory and BRS 60 will create a new name if necessary.

```
LDP     string pointers
BRS 48 or 60
EXCEPTION RETURN
NORMAL RETURN
```

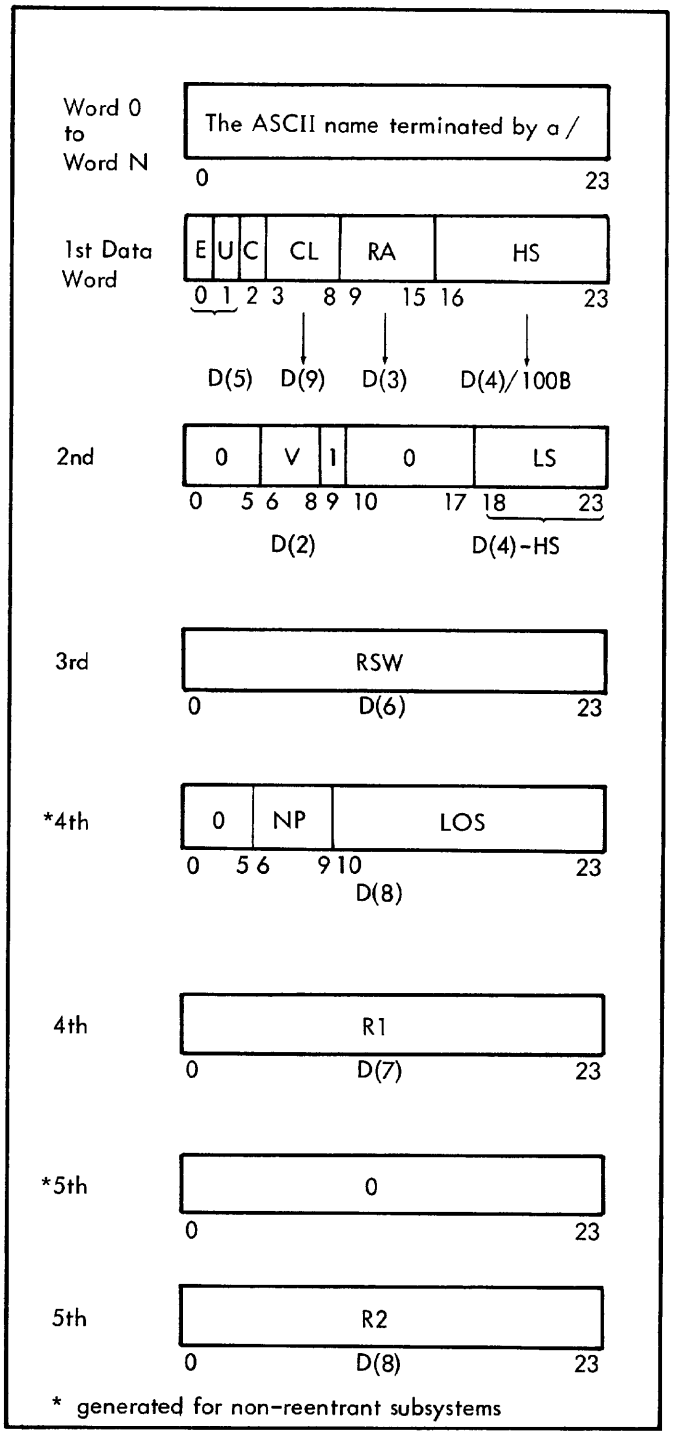


Figure 26. SUBIT Macro Data Words

13. EXECUTIVE COMMANDS RELATED TO FILES

When a user logs on the system, his complete file directory is read from the disc and placed in the file directory hash table along with the names of the physical devices. The "LOGIN" procedure is described in the SDS Terminal Users **Guide.**

The following executive commands are related to the user's file directory and are also described in the SDS Terminal Users Guide.

1. **FILES**
2. **WRITE FD**
3. **FD**
4. **DELETE**
5. RENAME

DELETE file is used to delete a file from the directory, and RENAME is used to change the name of a file in the directory.

FILES causes the complete directory to be typed while FB types only a single entry. Executive class users who have system status will receive the following special output:

p, dt, s name

Key	Tape Files	Disc Files
p	Tape position (octal)	Not used
d	Blank	2
t	File type (1 through 4)	File type (1 through 4)
s	File size	Index block pointer

A colon typed after FD or FILES in the above commands, will cause the length (in number of words) of a disc file to be typed out; the format is as follows where l is the length

p, dt, s, l name

Example:

-FD:/DEMO/CR

23, 512/DEMO/

Another feature of the system status typeout is that any control characters in the file name will be typed out in two characters; the first character is the ampersand "&". For example, if the name of the file was /(bell)PROGRAM/, it would type out the message

23, 12640/&GPROGRAM/

The command "DF" can only be used by users with a special system status since it can create new file names while bypassing all system protection. The complete file parameters must be typed as follows

DF file name AS p, dt, s

DF and AS are part of the command and are required for defining files. The disc file would be written in the following way.

DF /file name/ AS 23, 10240

An example of a tape file would be

DF 'file name' AS 7, 3, 10240

The command "WRITE FD" causes the current file directory (as it appears in the file directory hash table) to be written on the disc. A description of the disc format is given in Figure 13.

MAGNETIC TAPE

It is possible to read and write files on magnetic tape. The system will also read and write BCD tapes (with a defined format). Only users with peripheral status can cause tape commands to be executed. Normally only one user should be accessing tapes at any given time and such use should be restricted to periods of time in which there are not more than two or three users on the system.

All tape operations use an implied tape drive number of 0. This can be changed to 1 by typing the executive command:

-STN 1 (RET)

Under no circumstances should the same physical tape reel be used for recording both BCDTAPE files and standard 940 files.

BCD TAPE FILES

Each file is separated by an end-of-file (EOF) mark with two consecutive EOFs as the last recorded data on the reel. Input or output can begin at load point as file number 1.

For input the system will read multiples of 3 character/word up to 132 characters. If a record is 79, 80, or 81 characters long, the system assumes an 80 character record is desired (card image). If the record is not a multiple of 3 character/word, zeros are appended to the data.

For output, the records are written as either 80 or 132 character/word records.

The commands associated with BDCTAPE are:

-COPY BCDTAPE TO /FILE/ (RET)	BDCTAPE Input
START AT: N (RET)	N = file position The default for N is the current tape position
-COPY /FILE/ TO BCDTAPE (RET)	BCDTAPE output
80 CHAR. REC? (YES/NO) (RET)	Default = 132
START AT: N (RET)	N = number (Default is current position)
-BCDREW (RET)	Rewinds and positions to file 2. This command should be used only on tapes which have at least 2 EOFs already recorded.

File position is by EOF count only. For standard files, the software outputs a position word at the beginning of the file which indicates the position count. For more information on BCDTAPE format see Figure 20.

STANDARD MAGNETIC TAPE FILES

To write files on magnetic tape the tape must first be initialized. This is done with the operator's NEWTAPE program.

This program causes the tape on drive 0 to be rewound and 3 dummy files will be written on the tape. Therefore, the first meaningful file on the tape will have a file position number of 4.

A disc file can be placed onto tape by:

-COPY /FILE/ TO 'FILE' (RET)

Note that the disc file /FILE/ is not affected by this operation. There is no confusion between the name /FILE/ and 'FILE' because the slash or prime characters are stored as a part of the file name in the file directory. The copy operation will cause an entry to be made in the file directory for the file 'FILE'.

To copy files from tape to disc, the tape file directory entry must be in the file directory in use. If the tape file was not created by the user who wishes to copy it to the disc, (as is often the case) it will not be in the user's file directory. Therefore, the operator will type:

-DF 'FILE' AS 4, 3, 7640 (see p, t, s)
-COPY 'FILE' TO /FILE/
NEW FILE

The choice of names for tape files is completely arbitrary. No file name identification is carried with the file on the tape. See Chapter 9 for more information on the format of magnetic tape files.

14. EXECUTIVE COMMANDS

Table 5 gives a complete list of Executive Commands.

Table 5. Executive Commands

The following commands are accepted by the executive for all users.	
LOGOUT	Allows user to log out
WRITE FD	Write file director on disc
RENAME	Renames a file
DATE	Types date and time
KILL PROGRAM	Kills program relabeling only
RESET	Clears all of user's memory
COPY	Copies file to file
FILES	Types file directory
FD FOR	Types selected file directory entry

Table 5. Executive Commands (cont.)

GO TO	Goes to a "GO TO" (type 1) file
PLACE	Places a "SAVE" type program (type 1) in core
SAVE	Save program; creates GO TO or type 1 file
BRANCH	Branches into a program
DELETE	Delectes a file
TIME	Types real time used
STATUS	Types user's relabeling status
MEMORY	Types user's unused memory
" (Quote)	Causes typing to be ignored by EXEC Allows user to type comments
DUMP	Dumps all program, saves status

Table 5. Executive Commands (cont.)

RECOVER	Recovers from a Dump file (type 4)
CONTINUE	Returns to subsystem being used before the return to Executive
RELEASE	Releases a subsystem
EXIT	Allows a user to LOGOUT without writing file directory
ACCEPT MESSAGES	Indicates the user is willing to be linked
REFUSE	Indicates the user is not willing to be linked
LINK TO	Allows a user to link to the operator or a particular account and user name
Control D (Joint pressing of CONTROL and "D" keys of the keyboard)	Cancels the effect of " (Quote)
BREAK LINK	Allows a user to discontinue linking
PMT	Lists the PMT entries that a user has acquired
CREATION	Creation date of files
CFD	Creation date of a particular file
The following commands are recognized by the Executive for users with operator status.	
SHUT DOWN	Starts system shut down
UP	Cancels shut down
HANG UP	"Hangs up" selected teletype phone lines
ANSWER	Answers (or enables) data subset
WACCOUNTING	Controls accounting to paper tape
LETTER	Types broadcast letters
GFD	Gets another user's file directory
ENABLE	Enables a subsystem group
DISABLE	Disables a subsystem group
LOOK	Looks at real core locations
SYSLD	Allows load from disc directly into user's core
The following commands are recognized by the Executive for users with system status.	
USERS	Types number of users on system
WHERE IS	Give Teletype number for a user
WHO IS ON	Types users on system by account and name
REWIND	Rewinds tape, resets tape logic
RLT	Release tape
STN	Sets tape number

Table 5. Executive Commands (cont.)

PTN	Types tape number
SETEXEC	Sets user status
POSITION TAPE	Positions tape (not to be used on BCD tapes)
MTP	Types current tape position
DF	Allows a file directory entry to be set up (see Chapter 13)
REMOVE FILE	Removes file from directory (without deleting)
PSP	Types error counters, etc.
BCDREW	Rewinds a BCD tape
DBITS	Prints the number of 256 word data blocks remaining in the disc bit map
SMT	Lists the SMT table entries
SDATE	System start-up time
The following commands are recognized by the Executive for users with subsystem status.	
RSMT	Reads in from RAD a SMT Page
SYSDP	Allows core to be dumped directly on disc
SSMT	Sets the shared memory table

USER COMMANDS

LINK TO Account Number User Number or OPER

This command allows a user to link to another user that is currently on the system or to the operator. If (") is typed, the Executive will ignore (not regard as a command) anything that is input until a control D is typed. This allows a user to converse with the user that he is linked to. If the link cannot be made at this time, the Executive will respond with one of the following messages:

NOT ENTERED

BUSY

IN 8-LEVEL MODE

NOT ACCEPTING LINKS

Example: LINK TO F5103

LINK TO OPER

PMT

Types the user's current Program Memory Table in the following format:

aa DRMPOS: bb, cc (PAGE dd)

where

aa is the pseudo relabeling byte number.

bb is the RAD address (shifted right three places).

cc will type: RO for read only
EX for Executive page
DR for RAD page

dd (if typed) will be the real page number in memory.

CREATION ⏎ or :

This command can be used to output the creation date (the date that the file was last updated) and the access count (the number of times the file was accessed since the EDIT utility program was run) of a user's files. If the command is executed with user status (see SETEXEC command) the following is output:

dt, M/D NAME terminated by carriage return

dt, M/D, a NAME terminated by colon

where

d = 2 for disc file or blank for tape file

t = file type (1 through 4)

M = month

D = day

a = access count

NAME = file day

If the command is executed with system status (Executive flag has value of minus one), the following is output:

p, dt, IB, M/D NAME terminated by ⏎

p, dt, IB, M/D, a NAME terminated by colon

where

p tape position (octal)

IB index block address (MOD 4) for disc file or size of tape file.

CFD : (or blank) NAME ⏎

This command outputs the creation date and access count of the particular file specified by NAME. The command may be followed by a blank or a colon. The output format differs slightly depending on the setting of the Executive flag. See CREATION command for output format.

OPERATOR STATUS

SHUT DOWN (toggle switch 1 required)

After the operator toggles console switch 1, the command will set a flag that initiates system shutdown. All lines that are not currently being used will be made unavailable. As soon as the users that are currently on the system either log off or hang-up, their teletype lines will be made unavailable.

UP (toggle switch 1 required)

After the operator toggles console switch 1, the automatic shut down flag described under SHUT DOWN is reset so that teletype lines are now available. The operator must re-answer (by using the ANSWER command) all lines that have previously been made unavailable.

ANSWER (toggle switch 1 required)
k, m-n, . . .

This command enables selected teletype lines for users. The operator may specify single numbers, indicated by "k", separated by commas, or a range of numbers where the range is separated by a dash, or any combination. Spaces are ignored. The strings is terminated by a carriage return. If the line has already been enabled, the command will have no effect. Note that after the SHUT DOWN command has been issued, a line can be made available by this command but it will become unavailable after the user logs out.

HANG UP (toggle switch 1 required)
k, m-n, . . .

The command has two functions; it may be used to hang up a user while he is logged in (in this case the line will go ready again after the hangup operation has been completed provided the SHUT DOWN command has been used), or it may be used to make a line unavailable if no one is currently using the line. The format is the same as ANSWER.

LETTER ⏎

LETTER OFF/ON (Response from Executive)

LETTER n

This command has three functions and two formats. The second format, where a number n is typed after the command is used to type a broadcast letter, where n is the letter number from one to six. The operator can then inspect the text of any letter.

The first format is used to control the transmission of broadcast letters. If the letter switch if OFF no users will receive broadcast letters. If the letter switch is ON all users will receive the letters. The operator can add or delete letters using commands in /OPER/ program.

While the operator is inserting a letter, the letter switch must be OFF. If the operator uses the first format of this command, the status of the letter switch will be reversed and the new status will be printed.

Example: Assume the letter switch is on:

-LETTER ⏎	
LETTER OFF	The switch is now off
-LETTER ⏎	This will set switch on
-LETTER ON	

When the operator adds a new letter, each user will receive the letter when he logs on. The users that are on the system while the letter is being added will receive the letter when they return to the Executive (escape to the part of the Executive that types the - .

WACCOUNTING n ⏎ (Currently not implemented)

After the ⏎ is typed, the following message will type:

TOGGLE SW. 1 ⏎

The command will not be executed until console switch 1 is toggled.

If n = 0, accounting information being punched on paper tape when users log out will be stopped.

If n = -1, punching of the accounting information on paper tape when users log out will be started.

GFD aa nnnn ⏎

The command is used by the operator to get a file directory belonging to another user for special background or non-timesharing processing. The operator's own file directory and user number is replaced by those belonging to the account number "aa" and user name "nnnn" although the operator's account number and control parameters are retained. Subsequent to issuing a GFD command, the operator must not issue: RENAME, DELETE, WRITE FD, EXIT, or LOGOUT; otherwise, accounting misinformation will result.

ENABLE s ⏎

DISABLE s ⏎

where

s is the name of a subsystem in the group.

The subsystems are:

Group 1 - TAP, DDT

Group 2 - Currently not used.

LOOK

This command is typed in the following format:

LOOK a, n ⏎

a bbbbbbbb

a+1 bbbbbbbb

etc.

This command allows an operator or system class user to display real memory addresses where "a" is the first location to be displayed (in octal) and "n" is the number of locations (in decimal) to be displayed. The format of the typeout is as indicated in the example where "a" and "a+1" are the octal addresses and "b" represents the contents in octal.

SYSLD

The command is typed in the format:

SYSLD a ⏎

TO b ⏎

LOC c ⏎

This command allows a user to load his program memory from any location on the disc into any of his eight pages. "a" and "b" refer to his page numbers (0 to 7) and "c" is either a real disc address or a number from 0 to 7 referring to discs 0 through 7, with the load starting at arm position 64 of the given disc. Also, "c" may be formatted "n.m" where "n" is the disc number described above and "m" is a number from 0 to 7 referring to a relative page number of arm position 63.

Arm position	62	63
	n.0	n.5
Disc n	n.1	n.5
	n.2	n.6
	n.3	n.7

SYSTEM COMMANDS

USERS ⏎

nn

Types the number of users (nn) currently logged on the system.

WHERE IS aa nnnnnnn ⏎

xx

By typing the account number ("aa") and the user's name ("nnnnnnn . . ."), this command will type the current teletype number ("xx") of the user.

WHO IS ON ⏎

xx aa nnnnnnn . .

xx ss

This command outputs the date and time and then lists the users that are currently logged onto the system.

xx	teletype number
aa	user's account number
nnnn . .	user's name
ss	status of line (if user is logging on or off)

REWIND ⏎

This command releases and rewinds the tape regardless of its current status. It is applied to the current tape number (0 or 1).

RLT (RET)

This command releases the tape making it available to other users.

STN n (RET)

Allows a user to specify the tape unit number (0 or 1). BCD TAPE operations will normally use tape unit 0 unless 1 is specified by issuing this command.

PTN (RET)

n (RET)

Types a user's current tape number (0 or 1).

POSITION TAPE (RET)

This command will cause the current tape to move to the beginning of the next file.

MTP (RET)

Types the current tape position as far as is known to the Executive. This command does not check the actual position by reading tape.

PSP (RET)

This command outputs the date and time followed by a symbol which indicates a type of system error and the count of the number of such errors that have occured. If an error counter has the value zero, neither the symbol nor the count will be listed. The symbols are as follows:

TN	Tape noise errors
TU	Unrecoverable tape read errors
TR	Tape read errors
TW	Tape write errors
PF	Power Failures
DC	Disc channel errors
D	Disc controller errors
DU	Unrecoverable disc errors
DF	Disc failures
RC	RAD channel errors
R	RAD controller errors
RU	Unrecoverable RAD errors
I2	RAD I2 interrupt errors
CP	CPU memory parity count
IP	I/O parity errors
XB	Map index block error
FD	File directory errors
DO	Teletype data overrun
TD	Duplicate teletype on interrupts
IT	Illegal teletype off interrupts

SETEXEC nn (RET)

Sets or resets executive status if the user's status parameters allow the use of that status. The status set is then assigned to any fork started by the system executive with the "GO TO" command.

nn	
-1	Sets executive status
0	Cancels executive status

REMOVE FILE nn, (RET)

This command allows a user with system or operator status to remove an entry from a file directory without using the DELETE command. Since it may not be possible to delete a file if the name contains leading spaces or other spurious characters, it may be required to use this command as a last resort. The command removes a file from the "in-core" directory by referring to the file name's position "nn" in the printed file directory. The FILES command must be given just prior to using this command in order to find the current relative position of the name. The file directory is NOT rewritten on the disc by this command.

SMT (RET)

This command lists the SMT entries. The format is the same as the PMT command.

SUBSYSTEM COMMANDS

SDATE (RET)

The system responds: ENTER DATE AND TIME: M/D TTTT (underlined text is system response, M/D TTTT is operator input).

This command is used if the operator errs when providing the system start-up time. It is essential that the system start-up time be correct since the accounting records are affected by this parameter.

M/D = Month and day (as 1/22)

TTTT = Time (where 1300 means 1 PM)

SYSDP

The command is typed in the following format:

SYSDP a

TO b

LOC c

This command allows a user to dump his program memory onto any location on the disc from any selected pages of his eight pages of program relabeling. The format is the same as SYSLD.

RSMT ⏎ N (N-SMT pseudo-relabeling byte)

This command reads an SMT page from the RAD if the page is currently in core. The purpose of this command is to read in a new copy of an SMT page in case the copy that is currently core resident has been damaged or altered. The new copy will reside in the same core occupied by the old copy. This command is the same as the BRS BE+15.

SSMT ⏎

N ⏎

DRMPOS: aaa, E, R

N	SMT pseduo-relabeling byte
E	Make page Executive (set bit 1 of SMT entry)
R	Make page Read-Only (set bit 18 of SMT entry)
aaa	RAD address truncated by 3 bits (see output of SMT command)

SSMT is used to set an SMT entry. E and R are optional.

15. REENTRANT SUBROUTINE CALLS

Since the Executive is reentrant, any instruction it executes that will alter memory will always reference the TS page. The TS page is in page 0 of the Executive's relabeling. The BRM instruction must be used indirectly since it alters memory by storing the return linkage into the effective address of the instruction.

Assume the following is available to the Executive in the user's temporary storage page:

```
LVL1    ZRO
        BRR*    LVL1
LVL2    ZRO
        BRR*    LVL2
```

The Executive wishes to call subroutine SUB1 which in turn will call a subroutine SUB2.

```
        :
        :
EXC     BRM*    SUB1
        :
        :
SUB1    ZRO     LVL1
        :
        :
SUB1C   BRM*    SUB2
        :
        :
SUB1N   BRR     LVL1
        :
SUB2    ZRO     LVL2
        :
        BRR     LVL2
```

The instruction at EXC stores a markword at LVL1. LVL1+1 is executed as the next instruction. Since LVL1 now contains EXC (the markword), the BRR* transfers control to SUB1+1:

((LVL1))+1 = (EXC)+1 =SUB1 +1 → P

The BRR at SUB1N will transfer control to EXC+1:

(LVL1)+1 = EXC+1 → P

The call at SUB1C will store the markword (SUB1C) at LVL2 and the BRR* at LVL2+1 will transfer control to SUB2+1.

The word pairs (LVL1, LVL1+1, etc.) are in the TS page from the CFI1 through RLV4+1. They are initialized for each user when he is given his TS page. The coding which does the initialization is at TSONI.

16. MISCELLANEOUS FEATURES

A user may dismiss his fork for a specified length of real time by executing a BRS 81 with the dismissal time, in milliseconds, in A. At the first available opportunity after this time has been exhausted, his fork will be reactivated. The contents of A are changed.

A user can read the real-time clock into A and the system start-up date and time into B by executing a BRS 42. The number obtained increments by one every 1/60th of a second. An Executive fork can read the elapsed time counter for the user into A by executing a BRS 88. This number is set to 0 when entering the system and increments by 1 at every 1/60th second clock interrupt while the fork is running.

To obtain the date and time, the user can execute a BRS 91. This string pointers in the A and B registers. The string contains in order, the month/day, hour (0-23) and minute at which the instruction is executed.

A user may dismiss a fork until an interrupt occurs or the fork is terminated by executing a BRS 109.

A fork tests whether it is Executive or not by executing a BRS 71. The type of Executivity is returned in B. If B equals 1, the fork is subsystem. If B equals 0, the fork is user. If B equals -1, the fork is system and subsystem. If B equals -2, the fork is system. If B is negative, BRS skips on return.

An Executive fork can dismiss itself explicitly by executing a BRS 72 (see Chapter 2).

There are two operations designed for Executive BRSs which operate in the user mode with a map differing from the one they are called from. BRS 111 returns from one of these BRSs, transmitting A, B, and X to the calling fork as it finds them. BRS 122 simulates the addressing of memory at the location specified in A. If new memory is assigned, it is put into the relabeling of the calling fork. If memory panic occurs, it appears to the calling fork that it comes from the BRS instruction.

An Executive fork can cause an instruction to be executed in the system mode by addressing it with EXS.

There are switches in the Monitor that can be set by an Executive fork with a BRS BE+13. It takes the new switch value in A and the switch number in X. It returns the old switch value in A.

An absolute location in the Monitor relabeling can be read or changed by an Executive with a BRS BE+4. The absolute location is in X, the new value, if any, in A. The BRS reads if B is positive and changes the word if B is negative.

An Executive fork can also force a new page to be read from the RAD with a BRS BE+15. It requires an SMT pointer in A.

An Executive fork can test the state of any breakpoint switch with a BRS BE+7. The switch number is in X. The BRS skips if the switch is down.

An Executive fork can crash the system with BRS BE+8. A fork can set Executive status with a BRS BE+16 and 76543210B in the A register. System status is required.

17. UTILITY PROGRAMS

DSWAP

DSWAP is a self-loading utility program, usually stored on magnetic tape, that initializes the TSS system. It performs the following functions:

1. If breakpoint 1 is reset, the first 32K of core is written onto discs 0 and 1 in arm positions 62 and 63. If breakpoint 1 is set, the current contents of core are not written onto disc.

2. The first 14K of the Monitor is loaded into core from arm positions 62 and 63 of the disc selected by breakpoints 2 through 4.

   Copies of the monitor can be stored on discs 2, 4, and 6.

   The three breakpoint switches are regarded as representing a binary number between 0 and 7. Therefore, to load the system from disc 2, breakpoint 3 would be set.

After the monitor is loaded, DSWAP halts at location 24B. When the halt is cleared, a branch occurs to a portion of the Monitor which has just been loaded. The system executes a transfer from 930 to 940 time-sharing mode and the remaining 4K of the Monitor is read into core.

OPER PROGRAM

The general function of the OPER program is to provide the operator with information or control of the following permanently assigned areas of the disc:

- File directories.
- User/Account directory.
- Accounting data storage area.
- Broadcast letter area.

The specific programs provided are not used as often as the functions which are available through system Executive. Therefore, the program is initiated with a GO TO type statement and is normally the operator's file directory.

CONTROL COMMANDS

The program has a simple command dispatcher that indicates it is ready to receive a command by typing an asterisk. In order to reduce operator error, the commands must be typed completely. Each command is described in detail in this chapter and the commands are listed by category with a brief description in Table 6.

GENERAL OPERATING INSTRUCTIONS

The operator calls the program by typing

```
GO /OPER/ (RET)
*
```

where

/OPER/ is the name of the program.

* which is typed by the program, indicates that the program is ready to receive the first command. The typed command is then followed by a carriage return or a linefeed if appropriate.

(RET) Generally the carriage return confirmation indicates to the program that the complete output is desired. There is also a linefeed ((LF)) command which indicates that a selected output (for a particular user number in case of the file directories or for a particular account number in case of the user account directories) is desired and that a user number or account number will be supplied as appropriate to the command.

If an invalid command is typed, the program will respond with a question mark and type the asterisk, indicating that the program is ready for another command.

Table 6. Control Commands

COMMAND	DESCRIPTION
	File Directories
FILES	Outputs all or selected file directories.
CLEAR FILE	Clears a selected file directory.
TIME	Outputs the user's real and computer time as carried in the file directory.
RESET TIME	Same as time but also clears the time words to zero.
SET DAY	Validates all or selected users for 24 hour/day.
SET HOUR	Validates all or selected users for any selected time.
LENGTH	Computers length of all files by account number.
SIZE ACCOUNT	Uses length output to compute maximum storage used.
GARBAGE	Removes unused areas from the overflow file directory area.
POINTER	Indicates next available overflow storage area.
	User Account Directory
UAD	Outputs all or selected user/account directories.
ACCOUNT	Creates a new account or changes an account password.
NAME	Creates a new user name or changes a user name.
CANCEL ACCOUNT	Cancels an account directory.
CANCEL NAME	Cancels a user name out of a user/account directory.
	Accounting Storage
COPY RECORDS	Copies accounting records to a file.
CLEAR RECORDS	Copies accounting records to a file and then clears the accounting storage area.
	Broadcast Letter
COUNT LETTER	Counts the number of users who have not received each of the six broadcast letters.
REMOVE LETTER	Allows the operator to remove a broadcast letter.
LETTER	Allows the operator to create a broadcast letter.
	Miscellaneous
HELP	Lists all of the operator executive routine commands.

PROGRAM LOADING AND ASSEMBLY PROCEDURE

The program consists of two symbolic files, usually called /OP1/ and /OP2/. The first file is assembled by TAP in the usual manner while the second file is assembled using the CONTINUE command to the Executive since it uses constants contained in the first file. Both binary outputs are loaded using the DDT command ;T, and the program is then ready to run, starting at location 240B. Normally a program identifier is placed at location 237B so that the program is saved from 237B to the final address (as typed by DDT) with the starting address as 240B.

OPERATOR EXECUTIVE ROUTINE

The following paragraphs describe all the commands contained in the operator Executive routines program. The command is shown along with the appropriate terminator.

where

- (RET) only the carriage return is appropriate
- (LF) only the linefeed is appropriate
- (RET) / (LF) either carriage return or line feed is appropriate.

The function of the command is then described, followed by the operating instructions; if any messages are typed by the program, the messages are then shown along with the appropriate action to be taken by the operator. Actual example(s) of the use of the command is then shown along with a typical output, if any. In the examples, underscored copy represents copy produced by the system. Unless otherwise indicated, copy that is not underscored in an example must be typed by the user. Following the example an output description is supplied if appropriate.

Note: The outputs and inputs, if any, of all commands are symbolic files except for the COPY RECORDS and CLEAR RECORDS which supply binary (type 2) output files. This means that the comment OUTPUT FILE includes the physical devices such as the printer and teletype, except for the COPY RECORDS and CLEAR RECORDS commands.

ACCOUNT

COMMAND: ACCOUNT (RET)

FUNCTION: Creates a new account or changes an account password in the account user directory.

After giving the command the operator types the account number and the password, terminated by a carriage return. This will either create a new password or change an old one. The operator types the account parameter words, separating each parameter by a space, and terminating the list with a carriage return.

Examples:

*ACCOUNT (RET)
B1XYZ
0 0 (RET) / (LF)

where

"B1" is the account number

"XYZ" is the password,

0 0 set the account parameters to zero. The account parameters are currently not used by the system. However, they must be supplied for the command to work properly.

This command may be used to change more than one account. Note that after the operator types the new account parameters he may respond with either a line feed or carriage return. A carriage return indicates that the operator does not wish to change more accounts. The line feed allows another account to be changed.

*ACCOUNT (RET)
A1PASS (RET)
0 0 (LF)
A2PASS (RET)
0 0 (RET)
*

CANCEL ACCOUNT

COMMAND: CANCEL ACCOUNT (RET)

FUNCTION: Cancels account password and user names from an account directory.

After giving the command terminated by a carriage return, the operator types the account number followed by a carriage return. The program will then type the asterisk.

Examples:

*CANCEL ACCOUNT (RET)
B1 (RET) / (LF)

Several accounts may be canceled by terminating the account number with a line feed.

*CANCEL ACCOUNT
C5 (LF)
A8 (LF)
B7 (RET)
*

CANCEL NAME

COMMAND: CANCEL NAME (RET)

FUNCTION: Cancels a user name out of a user account directory.

After giving the command terminated by a carriage return, the operator types the account number and user name, followed by a carriage return. If the name is located, the program will type

OLD
*

completing the operation. If the name cannot be located, the program will type

NEW
INVALID USER
*

and the operator may then correct the name.

Examples:

*CANCEL NAME (RET)
B1JONES (RET)
OLD

The command can be continued if the operator wishes to cancel more than one name. Note that either a line feed or carriage return can follow the account and user name.

*CANCEL NAME (RET)
A2BET (LF)
OLD
A3CET (RET)
OLD
*

CLEAR FILE

COMMAND: CLEAR FILE (LF)

FUNCTION: Clears a selected file directory.

The operator types the user numbers for the file directories that are to be cleared. The command must be terminated by typing a user number that is greater than the last valid user number. Normally the operator will terminate by typing 7777 and the program will respond with the message

END OF JOB.

Examples:

*CLEAR FILE (LF)
234 (RET)
416 (RET)
7777 (RET)
END OF JOB

This command does not clear the bit map. The MAP program must be run in order to update the bit map.

CLEAR RECORDS

COMMAND: CLEAR RECORDS (RET)

FUNCTION: Copies accounting records to a file and then clears the accounting storage area.

After the operator has given the command, the program will ask for an output file; this file cannot be a physical device such as the PRINTER or TELETYPE since the output is binary (type 2 file). If a satisfactory file name is given, the program will write the accounting records to the file and return to the asterisk. If a bad file name is given, the program will ask for the output file again.

Examples:

*CLEAR RECORDS (RET)
OUTPUT FILE: /ACCT/ (RET)
NEW (OLD) FILE (RET)

COPY RECORDS

COMMAND: COPY RECORDS (RET)

FUNCTION: Copies accounting records to a file.

After the operator has given the command, the program will ask for an output file; this cannot be a physical device such as the PRINTER or TELETYPE since the output is binary (type 2 file). If a satisfactory file name is given, the program will write the accounting records to the file and return to the asterisk. If a bad file name is given, the program will ask for the output file again.

This command outputs the account number, user, name, connect time, and CPU time. It then calculates the maximum amount of disc storage used and prints it with the above information.

Examples:

*COPY RECORDS (RET)
OUTPUT FILE: /ACCT/ (RET)
NEW (OLD)FILE (RET)
ACCOUNTING DATE (M-D): 5-27

COUNT LETTER

COMMAND: COUNT LETTER (RET)

FUNCTION: Counts the number of users who have not received each of the six broadcast letters.

The operator merely gives the command terminated by a carriage return; the program will then give the count in the following format:

1	0
2	976
3	0
4	0
5	1024
6	0

where the number in the left column is the letter number and the number in the right column is the number of users who have not received the letter (0 indicates the letter is not being used and 1024, as for letter number 5, indicates the letter has not been released). The letter program is currently implemented for a maximum of 1024 users.

FILES

COMMAND: FILES ↵/⏎

FUNCTION: Provides to an OUTPUT FILE the complete or selected file directories.

If the command is followed by a carriage return (Figure 27) the program will ask for the output file by typing OUTPUT FILE:. The operator may then type any appropriate output file name. If a wrong file name is supplied the program will again type the message OUTPUT FILE:. The normal output file will be the printer since the output may exceed the capacity of disc files. The message END OF JOB will be typed when the last file directory has been printed. The output begins with user number 1 and continues for all the valid user numbers on the system. If a particular user number has an overflow file directory associated with it, the overflow directory will be listed immediately following the user's first directory.

When the file directory for the last user number has been printed, all of the overflow directories should have been printed. If any overflow directories remain (were not referenced by a user number), they will be supplied at the end of the listing under the title LOST OVERFLOWS.

If the command is followed by a line feed (Figure 28), the program assumes the output file will be the teletype. The operator must type the user numbers for the file directories desired. When a user number is typed that is greater than the last valid user number, the program will type END OF JOB and terminate.

OUTPUT DESCRIPTION

Typical output has the following form:

```
211  0:00.51   0:15   77777777

A1   5  9/26      2   23000000   31176/CONVERT/
```

where

211 user number

0:00.51 hours, minutes, and seconds of computer time.

0:15 hours and minutes of real time used (since reset time).

77777777 Valid on-time where each bit represents an hour of the day. The left-most bit represents 00:00 to 01L00.

A1 Account number

5 Number of times the file was accessed since last disc re-ordering. Reaches a maximum of 77B and stays there.

9/26 Creation date,

60000000 (would appear instead of the 2) Flag bits indicating file was written on.†

†These bits are used by the concurrent tape back-up routine and the disc file re-ordering routine.

```
-G0 /0PER/

*FILES

0UTPUT T0:  TELETYPE

2/3  14:31
   1  99:05.23 4059:37 77777777
  *1   2 01/22          12  21000000   116430 /F0R3/
  *1   0 11/27           2  23000000    17413 /$/
  *1   2 12/10          42  21000000    15303 /TAP/
  *1   2 01/22          22  21000000   115243 /F2C/
  *1   2 10/01           2  21000000    15301 WSD
  *1   3 08/13          32  21000000    11307 /XF2R/
  *1   2 01/22          12  21000000   136427 /F0R2/
  *1   3 02/01    60000000  21000000    61305 /FI0/
  *1   2 06/29          21  21000000     5326 /QED/
  *1   2 01/22          32  21000000   142401 /F0R1/
  *1   3 01/22          32  21000000   115265 /F2R/
  *1   2 08/21          22  21000000    55323 /DDT/
  *1   1 11/07    40000000  21000000   131373 /@0/
  *1   2 03/27          52  21000000    45317 /CAL/
  *1   0 10/10           2  21000000    31000 NEWTAPE
  *1   2 01/22          32  21000000   122402 /F0R4/
  *1   0 07/26          12  21000000   115126 /XF0R2/
  *1   2 08/07          32  21000000    51331 /BASIC/
  *1   0 01/22          12  21000000   164717 /MAP4/
  *1   2 08/09          22  21000000   101332 /XF2C/

   2  8:05.35 164:12 77777777
  *1   0 06/19          32  21000000    34153 /F0R10/
  *1   0 06/19          12  21000000    40145 /F0R20/
  *1   0 03/12          21  21000000   137604 /QED0/
  *1   0 06/19          12  21000000    40157 /F0R30/
  *1   0 03/12          12  21000000   137625 /CTRDP0/
  *1   0 06/10          22  21000000   110162 /DDT0/
  *1   0 12/17          12  21000000   163277 /FI00/
  *1   0 06/19          32  21000000    40171 /F0R40/
  *1   0 11/11          16  21000000    67272 /0PEX/
  *1   0 01/09          32  21000000    74501 /F2R0/
  *1   0 06/24          22  21000000    44174 /F2C0/
  *1   0 03/12          42  21000000   137637 /TAP0/
  *1   0 03/12          21  21000000   147601 /PL0P0/
  *1   0 03/12          32  21000000   147622 /BASIC0/
  *1   0 10/01           2  21000000   137533 /WSD0/

   4  4:53.95 192:44 77777777
  *1   0 10/09           6  23000000    53251 /G/
  *1   0 07/09           2  23000000   167541 /$/
  *1   0 01/28           2  23000000   112663 @8
  *1   0 01/28           3  23000000   112665 @3
  *1   0 01/28           3  23000000    71145 @2
  *1   0 10/09           5  23000000    53257 D
  *1   0 09/25           7  23000000    43260 BB
  *1   0 10/10           7  23000000    57255 L
  *1   0 01/28           2  23000000   112670 @7
  *1   0 04/11           7  23000000   130530 /CL0SED ACCTS/
  *1   0 10/10          10  23000000    57245 M
  *1   0 10/09           4  23000000    53264 E
  *1   0 09/25           4  23000000    43267 F
  *1   0 01/28           2  23000000   146735 @5
  *1   0 10/09           7  23000000    53270 H
  *1   0 01/28           4  23000000   162763 /A/
  *1   0 10/09           6  23000000    53277 CC
  *1   0 01/28           3  23000000   112672 @4
  *1   0 12/07           4  23000000   140563 /N/

   5  22:43.35 237:02 77777777
  *1   0 10/09           2  23000000   173502 /ASSM/
  *1   0 01/24           2  23000000    47040 /$/
```

Figure 27. FILES Command Terminated by Carriage Return

```
-G0 /0PER/

*FILES

2/3  14:37
426
 426  16:28.66 809:44 77777777
   *3   0 09/11          3  23000000     27260 /BEN0/
   *3   0 07/25         30  23000000    170205 /PR0C/
   *3   0 07/31          2  23000000    103350 /0CTAL/
   *3   0 08/05          2  23000000    170237 /XX/
   *3   0 11/13          4  23000000      4447 /UNAM/
   *3   0 08/01          3  23000000    103352 /USN0/
   *3   0 11/02          2  22000000    137406 /BLAB/
   *3   0 07/25          2  23000000      4666 /ADD/
   *3   0 07/26          3  23000000    177502 /CARD/
   *3   0 07/24          3  22000000    160251 /BSEX/
   *3   0 01/22         17  21000000     41260 CHECK
   *3   0 10/31         14  23000000     14441 /US/
   *3   0 08/01         13  21000000    164303 /0CTAL USE0/
   *3   0 07/18          2  22000000     57557 /BTPT/
   *3   0 07/05          3  23000000     13532 /INSTCHK/
   *3   0 01/09          7  23000000     13020 /TAX0/
   *3   0 01/22          6  23000000     36423 /CHK/
   *3   0 01/22          3  23000000    116751 /BB00K/
   *3   0 01/29          7  23000000    175367 /./

0VERFL0W: 4365

4365  0:00.00 0:00 0
   *3   0 01/29          4  23000000     71340 /B00K/
   *3   0 11/13          2  22000000      4463 /BUNAM/
   *3   0 01/22          3  22000000     45256 /BT/
   *3   0 11/02          2  23000000     47420 /LAB/
   *3   0 10/31          3  22000000     30313 /BUSN0/
   *3   0 01/22         10  22000000     61326 /BCHK/
   *3   0 09/11          4  22000000     20207 /BBEN/
   *3   0 09/13          3  21000000    100275 /1/
   *3   0 01/22          3  23000000    131316 /T/

0VERFL0W:  426

231
 231  0:07.65 5:53 77777777
   B2   0 04/24          4  23000000    124301 /USC1/
   B2   0 11/13          2  23000000     60476 /MIKE/
   B2   0 11/13          2  22000000     53337 /BM/
   B2   0 04/01         11  23000000    120307 /USC2/

7777

T0TAL:   16:36.31 815:37

END J0B
```

Figure 28. FILES Command Terminated by a Line Feed

2 The number of data blocks in the disc file

230000000 File type (23 means symbolic disc file)

31176 Index block pointer (or file size for magnetic tape).

/CONVERT/ Name of file (control characters are preceded by an & on the teletype or by a Δ on the printer).

GARBAGE

COMMAND: GARBAGE ⏎

FUNCTION: Removes unused overflow areas from the overflow directory area and makes the area available for use. See Figure 29.

The program first determines the current location of the overflow pointer; in other words, it finds the next available overflow area. This is typed out as follows where nnnn is the pointer:

OVERFLOW POINTER AT: nnnn

The program next types the following message:

GARBAGE COLLECTION READY TO START.

ONLY 1 USER ALLOWED ON SYSTEM.

"ESCAPES" WILL BE INHIBITED.

TYPE ⏎ TO CONTINUE.

The program pauses here until a confirming carriage return is typed by the operator. Since this program cannot be run if any users except the operator are logged on the system, the program next checks the number of users on the system. If more than one user is on the system the following message is typed, followed by a return to the EXEC:

MORE THAN 1 USER ON.

If the operator is the only user, the following message is typed:

GARBAGE COLLECTION STARTED.

The program will proceed with no further messages until the garbage collection has been completed. At that time it will again determine the location of the overflow pointer and type the message:

OVERFLOW POINTER AT: nnnn

The difference in the pointers will indicate the gain in overflow directory storage area due to the garbage collection. The message END OF JOB will then type and the operator will be forced to EXIT from the system so that his new file directory overflow pointer on the disc will not be destroyed. The operator should then take the system down and take a disc dump to save the new file directory arrangement.

HELP

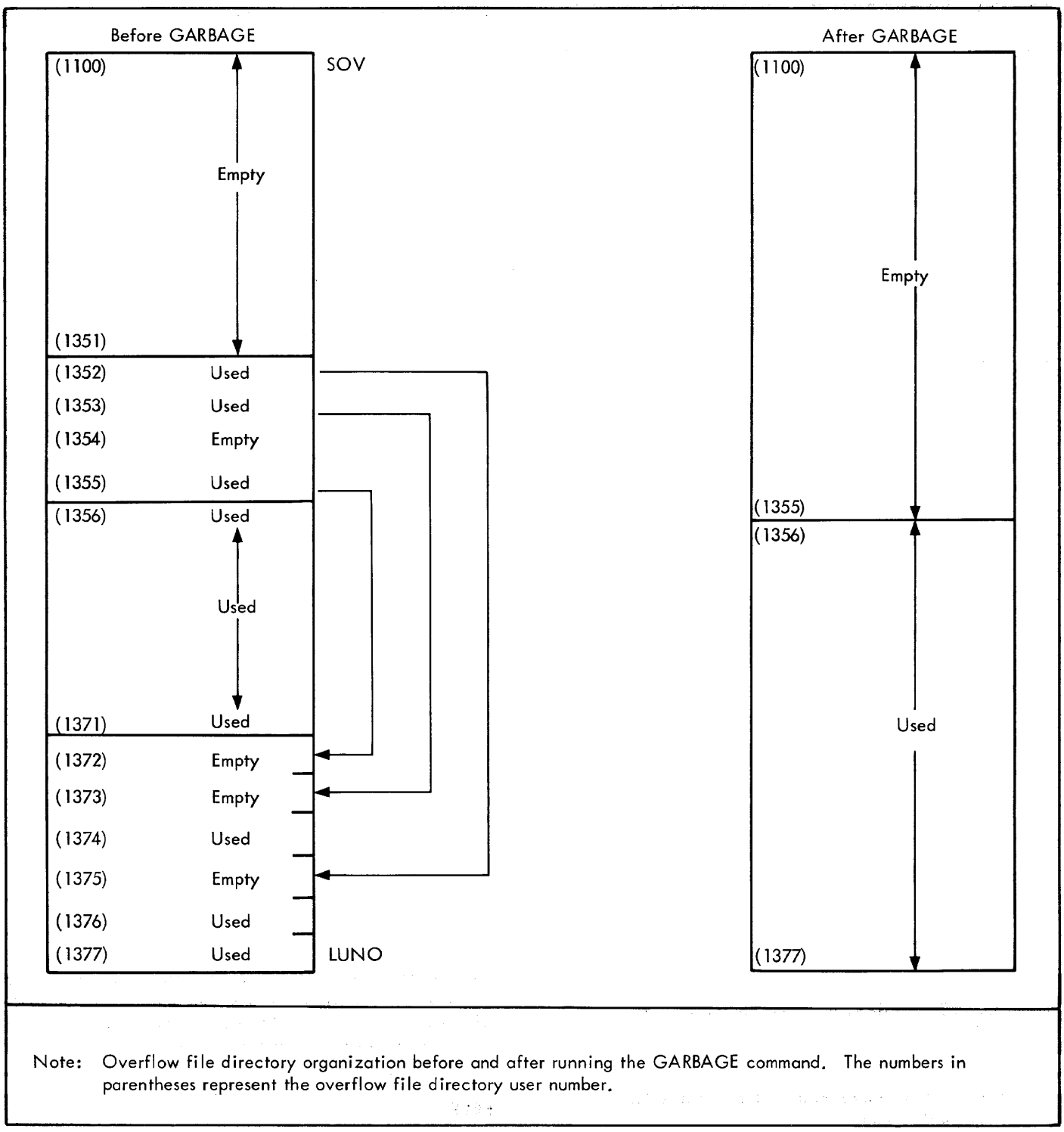
COMMAND: HELP ⏎

FUNCTION: Lists all of the commands the operator Executive routine will recognize.

LENGTH

COMMAND: LENGTH ⏎/⏎

FUNCTION: Outputs the amount of disc storage used by the account number.



Note: Overflow file directory organization before and after running the GARBAGE command. The numbers in parentheses represent the overflow file directory user number.

Figure 29. Garbage Collection

If command is terminated by carriage return, the program asks for an output file by typing OUTPUT FILE:. The operator should then type any appropriate output file name. If a bad file name is supplied, the program will type the message OUTPUT FILE: again. The message END OF JOB will be typed when the last user's file directory has been processed, and the output file has been closed.

If command is terminated by line feed, the program asks for an output file as described in the previous paragraph. The operator must type a user number for each file directory (and each overflow directory for which size he desires. The operator types a user number greater than the last valid user number (normally 7777), to terminate the list of user numbers. The disc storage by account for the selected user numbers will then be output as in paragraph 1.

LETTER

COMMAND: LETTER (RET)

FUNCTION: Allows the operator to create a broadcast letter.

Before giving this command, the operator must first set the EXEC letter switch to OFF. This is done by giving the EXEC command LETTER (RET). The EXEC will respond with LETTER OFF. Then the operator may GO TO the operator program and give the LETTER command. The program will respond with:

LETTER NO.:

and the operator must respond with a number from 1 to 6, corresponding to the letter that he wishes to create. The operator should then type a carriage return (after the letter number) and normally should type another carriage return so that the letter starts at the left edge of the paper. The operator should then type the letter and terminate with a control "D", the E.O.T. character. If the operator makes a mistake and would like to delete the character just typed, he may type a # sign; one character is deleted for each pound sign typed. When the operator has typed the control D, indicating end of letter, the program will respond with the asterisk. The operator must then return to the EXEC and type LETTER again. The EXEC will respond with LETTER ON and the new letter will be typed for the operator.

EXAMPLE: (starting from the EXEC)

-LETTER (RET) Giving the EXEC command to turn the letter switch off.

LETTER OFF

-GO /OPER/ (RET) Calling the OPER program

*LETTER (RET) Giving the command

LETTER NO.: 2 (RET) The program asks for a letter number

(RET) (RET) (RET)

TEXT OF LETTER (RET) (RET) The letter; maximum size is 189 characters

D^c
*

-LETTER 2 (RET) EXEC command to type a letter

TEXT OF LETTER EXEC types the letter

-LETTER (RET) Turning the letter switch on

-LETTER ON

TEXT OF LETTER The operator and everyone currently on the system receives the letter when they come back to the Exec.

NAME

COMMAND: NAME (RET)

FUNCTION: Creates a new user name, changes a user name in an account/user directory, or changes the parameters for a user.

After giving the command, terminated by a carriage return, the operator types the account number and the user name, followed by a carriage return. The program will respond with one of the two following messages:

OLD

NEW

indicating that the user name is new (not presently in the account user directory) or old (already in the account directory). If OLD is typed, the operator may continue if he desires to change the parameters. The operator types the parameter word, terminated by a carriage return. Note that the parameter word must contain the user number in the low order 12 bits and the user's control status in the high order 12 bits. If the user account directory for the account indicated already has 11 names assigned to it, the following messages will type:

NEW

FULL

The operator must first cancel an old name before he can add a new name if the directory is full (see CANCEL NAME).

EXAMPLES:

*NAME (RET)

B1JONES (RET)

OLD

60000023 (RET) / (LF)

This command can be continued for subsequent account and user names. Note that the parameter word can be followed by either a carriage return or line feed.

*NAME (RET) / (LF)
A1ABC (RET) / (LF)
00001025 (RET) / (LF)
A2BCA (RET) / (LF)
00001026 (RET) / (LF)

POINTER

COMMAND: POINTER (RET)

FUNCTION: Determine the next available overflow file directory storage area.

After the command has been given, the program will respond with the message:

OVERFLOW POINTER AT: nnnn

where nnnn is the current location of the overflow printer.

REMOVE LETTER

COMMAND: REMOVE LETTER ⏎

FUNCTION: Allows the operator to remove a broadcast letter from the letter bit map so that it is no longer addressed to anyone. Note that this makes the count (see COUNT LETTER) equal to zero. The actual letter text is not changed.

REMOVE LETTER automatically resets the letter switch to "ON". Therefore, the Executive will begin typing letters to any users which are on the system. The operator should not attempt to add a letter after removing one without first returning to the Executive and executing the LETTER command.

After giving the command, the program will type:

LETTER NO.:

and the operator must respond with a letter number, which must be a number from 1 to 6. The program will then remove the letter from the letter bit map.

EXAMPLE:

*REMOVE LETTER ⏎
LETTER NO.: 2 ⏎
*
—

RESET TIME

COMMAND: RESET TIME ⏎ / ⏎

FUNCTION: Provides to an OUTPUT FILE the real and computer time for all users and clears the computer and real times from the file directory storage area; the command may also be used for selected users.

Same as for the command TIME.

For examples of output see TIME.

Note that this command actually clears the computer and real time words from the file directories after outputting the information.

SET DAY

COMMAND: SET DAY ⏎ / ⏎

FUNCTION: Validates all or selected users for 24-hour usage of the time-sharing system.

If command is terminated by carriage return, no other action is required by the operator. The routine will set the valid time word in every file directory to 77777777 which validates the users for 24-hour usage of the system. The program will type END OF JOB when completed.

If command is terminated by line feed, operator must type the user numbers for the users to be validated for 24 hours. The command must be terminated by typing a user number greater than the last valid user number such as 7777; this will cause the program to type the END OF JOB message.

EXAMPLES:

-GO /OPER/ ⏎

*SET DAY ⏎

END OF JOB

—

—GO /OPER/ ⏎

*SET DAY ⏎

121 ⏎

23 ⏎

7777 ⏎

END OF JOB

SET HOUR

COMMAND: SET HOUR ⏎ / ⏎

FUNCTION: Validates all or selected users for any selected time of the day.

If command is terminated by carriage return, the program will type each user number, together with computer time, real time, and valid on-time and will pause after typing out the parameters for each user to allow the operator to change the valid on-time. If the operator does not care to change the valid on-time for a particular user, he merely types a line feed. Otherwise, he types the valid on-time word terminated by a carriage return. The program will then type out the parameters for the next user. After the last user parameters have been typed out, the program will type END OF JOB.

If command is terminated by line feed, the operator must type the user numbers of those users whose time parameter he wishes to change. The program will respond by typing the user number, computer time, real time and valid on-time. The operator may then type the new on-time parameter and terminate by typing a carriage return. (If a line feed is used to terminate the valid time word, the program will not change the valid time word.) The operator must terminate the command with a user number greater than the last valid user

number, normally 7777. The program will then write out the last file directory and type the message, END OF JOB.

Note: The time parameter word consists of one bit for each hour of the day where the left-most bit validates a user from 0000 to 0059, the second bit from 0100 to 0159 etc. To validate a user from noon to 1559, the operator would type the following time parameters:

7400 (RET)

EXAMPLES:

-GO /OPER/ (RET)
*SET HOUR (RET)
1 0/03.41 1:10 77777777 (LF) (Does not desire to change)
2 0:00.00 0:00 7777777 70 (RET) (Validates a user for the hours 1800 to 2059 only)
3 etc.
-GO /OPER/ (RET)
*SET HOUR (LF)
240 (RET)
240 0:01.23 3:45 70000000 (LF) (No change)
137 (RET)
137 0:02.23 10.54 7777 17777 (RET) (Changes the valid hours from "1200 to 2359" to "1100 2359")
7777 (RET)
END OF JOB

SIZE ACCOUNT

COMMAND: SIZE ACCOUNT (RET)

FUNCTION: Computes the maximum disc storage used by account from LENGTH outputs and provides this maximum as an input for the next SIZE ACCOUNT run.

The routine requires two input files and two output files which are requested by the program as needed. The contents of these files are as follows and the file names must be typed in the order indicated:

File 1 – INPUT FILE:

New input. Normally the output of a LENGTH run for the current day.

File 2 – INPUT FILE:

Previous maximum. Normally the maximum output from a previous SIZE ACCOUNT run which was produced as output file 4 previously. This input may also be the output of a LENGTH run if there has been no previous SIZE ACCOUNT run.

File 3 – OUTPUT FILE:

The complete report of the current run. First column is the same as input number 1, second column is the maximum between input file 1 and 2 and is the same as output file 4. The third column is the difference between input file 1 (new input) and the input file 2 (previous maximum) a number preceded by a minus sign indicates that the new SIZE is less than the previous maximum and that output on file 4 will not change from the previous maximum. If the third column is positive, then the new size file 1 was greater than the previous maximum so that the new maximum output will be equal to the new input.

File 4 – OUTPUT FILE:

New maximum. The format of this output is exactly the same as the format of the LENGTH output. This will normally be input file 2 for the next days run of SIZE ACCOUNT.

The program will type END OF JOB when completed. Figure 30 shows an example of SIZE ACCOUNT.

TIME

COMMAND: TIME (RET) / (LF)

FUNCTION: Provides to an OUTPUT FILE the real and computer time for all users or types the real and computer time for a selected user.

If command is terminated by a carriage return, the program will ask for the output file by typing "OUTPUT FILE". The operator should then type any appropriate output file name. If a bad file name is supplied the program will type the message "OUTPUT FILE:" again. The message END OF JOB will be typed when the last user's time has been output.

If command is terminated by a line feed, the program assumes the output file will be the teletype. The operator must type the user numbers for the time parameters to be typed out. When a user number is typed that is greater than the last valid user number (normally 7777), the program will type out the total that has been typed and then will type END OF JOB. Figure 31 shows an example of TIME.

UAD

COMMAND: UAD (RET) / (LF)

FUNCTION: Outputs to a file all or selected user account directories.

If command is terminated by carriage return, the program will ask for the output file by typing OUTPUT FILE. The operator should then type any appropriate output file name. If a bad file name is supplied, the program will type the message OUTPUT FILE:. The message END OF JOB will be typed when the last user account directory has been output.

```
7/1   0:31

ACT    NEW INP.     NEW MAX.     -    DIFF.

*1     797696       1330176      -    532480
*2     260352        574720      -    314368
*3     445696        557056      -    111360
*4     721408       1173248      -    451840
*5     856832       1026816      -    169984
*6      88320        122624      -     34304
*7     317696        338432      -     20736
A8      12544         12544           512
A1     217088        217088           4608
A2      52736         52736              0
A4     182784        183552      -     768
A5      87040        133376      -    46336
A6      12800         13312      -     512
B8      27136         27136              0
B1      13824         22784      -    8960
B2      80384         85760      -    5376
B3      11008         11008           1024
B5     123904        124928      -    1024
B6     528640        539904      -   11264
C8     133888        136704      -    2816
C1      58880         72960      -   14080
C2       2560          2560           @
 .
 .
 .
J1       2560          2560              0
J2      12288         13312      -    1024
J4       6912          8704      -    1792
K1      70400        100608      -   30208
K2     175872        177408      -    1536
L2      24064         46592      -   22528
L3      22784         27136      -    4352
M1      13824         28672      -   14848
N1       1536         10752      -    9216
N2       4352          9728      -    5376
N3      38656         44800      -    6144
N4      10240         10240           1792
N5      39936         39936           8960
N6       7936          7936              0
N7          0         10496      -   10496
08          0         35840      -   35840
07          0          5376      -    5376


TOT: 7373056      9600768      -   2200832
```

Figure 30. Example of SIZE ACCOUNT

If command is terminated by line feed, the program will ask for an output file as above. After typing the output file name, the operator should then type the account number of the user/account directories that he desires with each account number except the last one terminated by a line feed. The last one will be terminated by a carriage return. The account numbers are typed by the operator in the usual letter/number format. Figure 32 shows examples of users' output.

```
-GO  /OPER/ (RET)

*TIME (RET)

OUTPUT TO:  TEL (RET)

7/3       14:08
 1        0:00.71      0:24      77777777
 5        0:03.40      0:29      77777777
 6        0:00.20      0:03      77777777
17        0:00.15      0:05      77777777
20        0:00.26      0:08      77777777
25        0:96.53      1:26      77777777
27        0:00.00      0:01      77777777
32        0:00.41      1:14      77777777
45        0:0 (ESC)
-


-GO  /OPER/ (RET)

*TIME (LF)

7/3       14:09
25 (RET)
25        0:06.53      1:26      77777777
20 (RET)
20        0:00.26      0:08      77777777
5 (RET)
 5        0:03.40      0:39      77777777
7777 (RET)


TOTAL:  0:10.20      2:03

END JOB
```

Figure 31. Example of TIME

USERS

COMMAND: USERS (RET)

FUNCTION: Provides the operator with a list of valid users on the system, sorted by user number, account number or user name.

The operator types the command USERS terminated by a carriage return. The program will then request an output file. After the operator types the file name, the program will respond with:

SORT ON WHAT COL. (1, 2, or 3):

The operator then types 1, 2, or 3 for output sorted by user number, account number, or user name, respectively.

EXAMPLE: See user output on previous page.

DISC ZERO

COMMAND: DISC ZERO (LF)

<Password>

FUNCTION: This command will zero out all the discs on the system.

```
-GO  /OPER/ (RET)

*USERS

OUTPUT TO:  TEL

7/3      14:13

SORT ON WHAT COL.?  (1,2,OR 3) : 3

      12  *2     *              1 = User Number
    1166  G1     0038           2 = Account number
     522  G1     0035           3 = User Name
     521  G1     0036
    1171  G1     0500
     525  E5     1
     277  D5     1
     536  E5     10
     537  E5     11
     654  G3     141
     655  G3     142
     656  G3     143
     526  E5     2
     301  D5     2
     624  G7     200    WRIGHT
     623  G7     200    SPEIR
     622  G7     200    CHIOCHIO
     621  G7     200    JOHNSON
     711  G7     200    PATAPOFF
    1170  G1     2000
     626  G7     250    BRICK
     625  G7     250    SNYDER
     527  E5     3
     302  D5     3
    1167  G1     3000
     657  G3     345
     660  G3     349
     530  E5     4
     531  E5     5
     630  G7     500    SCHWARTZ
     627  G7     500    GUGGENHE
     532  E5     6
    1210  I2     658&C
    1206  I2     671&0
    1207  I2     674&N
     533  E5     7
     661  G3     7466
     534  E5     8
     662  G3     8466
     602  E4     8803
     604  E4     8811
     603  E4     8810
     357  E4     8812
     535  E5     9
    1172  G1     9000
     663  G3     9466
    1232  N8     @LMY&SK&C
       6  *2     A
     433  F3     A  BROWN
     171  B1     A. COX
    1203  G4     A. BELL
```

Figure 32. Examples of User Output

The operator types the command DISC ZERO, then a line feed. After the line feed which terminates the command, there will be no output on the teletype. However, the program is waiting for the password to be typed. Suppose the password is the word SDS and the control key must be depressed while all of the letters are typed. The following example applies:

*DISC ZERO (LF)

Control SDS

*

If NDISC (a system parameter) were equal to 8, then all 8 discs would be zeroed.

DISC EDIT PROGRAM

The purpose of the disc edit is to arrange the files on the disc for optimum data access. This routine places the files that are least often accessed farthest from the center arm positions (outside the mapped area). The files most often accessed are placed nearest the center, thus leaving all available free space at the access center.

The I/O control section of the Monitor always writes files within the mapped area of the disc. If the user opens for output (rewrites) an old file that has been moved to the outer arm positions by EDIT, the Monitor will rewrite the file within the mapped area.

PHASE ONE

In Phase One, each file directory (FD) is read, the edit bit set (bit 2 of word 1 of the file entry), the file length computed and stored in the file entry, and the FDs rewritten. The index blocks, for files which have been updated since the last time this program was run, are read, the user number stored in the last word, and the block written back.

An entry for every file on the disc is created in one of two tables that are built as the FDs are read. On table contains a two word entry for each file which has been read or written since the last time this program was run. (This is called the AB table.) The two words contain the last access data, access count, index block, disc address, and the number of data blocks in the file (includes the index block). The other table contains an entry for each file which has not been accessed since the last time the program was run. Each entry is one word containing the index block blocks of the file. (This is called the C table.)

PHASE TWO

The two tables (AB, C) are sorted. The AB table is sorted by access count and access date. The C table is sorted by index block disc address.

PHASE THREE

The files in the AB table are read from the disc, new index blocks created, and the files written on tape. A three-word record for each file is created and written on the RAD. The record contains the user number, the old index block disc address and the new index block disc address.

PHASE FOUR

Reads files in the C table and either writes them at their new location on the disc or writes them on tape until sufficient space is available to start writing them on the disc. This situation would occur if the total space required for the files was reduced since the last time this program was run and the free space exceeded 25 percent of the total capacity. A three-word record is created for each file and is written on the RAD. It contains the same information as the records written for the AB type files.

PHASE FIVE

The AB type files are read from tape and written in their assigned locations on the disc.

PHASE SIX

The C type files (if there are any on tape) are read from tape and written in their assigned locations on the disc.

PHASE SEVEN

The FDs are updated using the records written on the RAD. All records on the RAD are, so to speak, "passed" twice against each block of 64 FDs. When a RAD record is found to match a file entry for a particular user, it is marked and written back to the RAD and the index block disc address and access count is updated and the edit bit reset in the file entry of the FD. On the second "pass", before the block of FDs are written back to the disc, the edit bit is checked in each file entry and if found still set, a message is output to the typewriter with the user number and file name of the entry. Any file lost because its index block or data blocks were in error or couldn't be read or written, would appear in this list. All the RAD records are read and checked for mark bit set. If the mark bit is not set the user number and the old and new index block disc addresses are output to the typewriter. These should match with previous typeouts from the edit bit check of the FDs.

PHASE EIGHT

The operator is allowed to correctly enter new index block disc addresses for files which the program could not correctly identify.

PHASE NINE

All the index blocks are read and the last word checked against the user number to see that all files are properly placed. A message is output for files which do not check and should match with messages from phase seven. This is a check only and is not really necessary. It can be aborted at any time.

OPERATING INSTRUCTIONS

The operator may wish to take a disc dump to preserve the old status of the disc before running the edit.

1. With the system in time sharing mode, the operator should log in and type:

   -PLACE /Name of disc edit/

   -SYSDP 0 (RET)

   TO 1 (RET)

   LOC 1 (RET)

   This will put the edit program onto disc 1.

2. Take the system down by setting breakpoint 4. After a few seconds, lower the run switch to "idle" and press "start".

3. Mount the utilities tape which contains the DSWAP program. Also, mount a scratch tape on unit 6, set at 800 bpi. The edit routine uses this tape. After DSWAP has been read in, mount a second scratch tape on unit 7, set at 800 bpi.

4. Insure that the console teletype is turned on. The edit routine converses through this teletype.

5. Set break points 1 and 4 and load the DSWAP program. DSWAP will bring in the edit program from disc 1.

COMMANDS TO THE EDIT PROGRAM

EDIT will begin to type the following commands. New unit and old unit will be the same except when it is desired to increase or decrease the number of discs being used by the system. The part of the command typed by the operator is underlined.

FILE REORDER ROUTINE. READY MT 6, 7

TYPE DATA AS FOLLOWS: MM-DD-YR 12-06-68

TYPE LAST USER NUMBER-: 2377

TYPE NUMBER OF DISCS IN NEW UNIT AS :DD:
(Type :08:, :16:, or :32:)

TYPE NUMBER OF DISCS IN OLD UNIT AS :DD:
(Type :08:, :16:, or :32:)

SET USER NUMBER IN I. B. -: (Type YES (RET) or NO (RET) (RET))

Used for initial edit or when user number has been lost from the index block.

BEGINNING – 00146440 ENDING 00177374

At this point the system will type any error messages. The error messages will be discussed later.

JOB FINISHED

ANY FILES TO CORRECT: (The operator responds YES or NO depending on whether or not there were error messages.)

CHECK STARTED

END CHECK

ERROR MESSAGES

The following procedure can be used to recover files which the EDIT program could not identify. EDIT will first list the files for which it has a name but has lost the new index block pointer. Next it will list the files for which it has a new index block pointer but has lost the user number and file name. The operator can match up the two entries (if possible) when the line "ANY FILES TO CORRECT" is typed.

The format of the error messages is:

I. B. L: iiiiiiii USER uuuuuuuu FILE NAME ffffff

I. B. OLD iiiiiiii USER nnnnnnnn IB NEW iiiiiiii LOST FILE

If the operator responds "YES" to the "ANY FILES TO CORRECT" message, the following will be typed.

USER NUMBER: (Type the 4 digit user number)

OLD INDEX BLOCK ADDRESS: (Type the 8 digit index block address)

NEW INDEX BLOCK ADDRESS: (Type the 8 digit index block address)

OK: (Type YES (RET) if the information input is correct) (Type NO (RET) (RET) to start over)

IB OK: iiiiiiii USER uuuuuuuu FILE NAME ffffff

OK: (Type YES (RET) if this is the correct file and NO (RET) (RET) if not correct file. It will allow you to try again.)

ANY FILES TO CORRECT: (Type YES (RET) if more files to correct, otherwise type NO (RET) (RET))

CHECK STARTED (This procedure may be aborted any time.)

END CHECK

MESSAGES REQUIRING OPERATOR ACTION

FATAL ERROR, CAN'T R/W FDs AT (Disc Address)

Action: This message indicates a bad disc spot.

TAPE NOT READY - TOGGLE RUN SWITCH WHEN READY

Action: Put tape in ready and follow the instruction in the message.

SET 800 BPI----TOGGLE RUN SWITCH WHEN READY

Action: Follow the instruction.

PUT RING IN----TOGGLE RUN SWITCH WHEN READY

Action: Follow the instruction.

BAD TAPE----TOGGLE RUN SWITCH WHEN READY

Action: Mount different tape.

BAD TAPE----RESTART THE JOB

Action: Restart the program from disc, will not be necessary to restore the disc since nothing has been written yet.

FATAL ERROR WHILE WRITING ON TAPE--OK TO RESTART

Action: Restart the program from disc, will not be necessary to restore the disc since nothing has been written.

RAD ERROR HAS DESTROYED DISC DATA
RECOVER DISC FROM PREVIOUS DUMP AND RESTART

Action: Total restart.

DISC ERROR HAS DESTROYED DISC DATA
RECOVER DISC FROM PREVIOUS DUMP AND RESTART

Action: Total restart.

TAPE ERROR HAS DESTROYED DISC DATA
RECOVER DISC FROM PREVIOUS DUMP AND RESTART

Action: Total restart.

IF CORRECT TAPE TOGGLE RUN TO TRY AGAIN

Action: Caused by a tape label error. If the correct tape is mounted, follow instructions.

TAPE READ ERROR--
RECORD NUMBER IS-: nnnnnnnn S/B nnnnnnnn TOGGLE RUN TO START AGAIN

Action: Follow instructions. If it does not correct, the record can be accepted as is by setting switch 4 before toggling run, then resetting switch 4.

DISC WRITE PROTECT

Action: Set disc write protect switches off and toggle run switch.

IB RD: iiiiiiii USER uuuuuuuu FILE NAME ffffff

Action: Look in the listing of files under the user given the message for a matching index block disc address and file

name. Recover this file from a previous disc dump. Files indicated by this message cannot be recovered during file recovery procedure.

DISC ERROR AT (disc address)

FILE BAD IB: iiiiiiii

Action: The index block address will match with q and r messages. This file can be recovered during the file recovery procedure but will have errors in it.

IB L: iiiiiiii USER uuuuuuuu FILE NAME ffffff

Action: Match with r messages for recovery.

IB OLD iiiiiiii USER uuuuuuuu IB NEW iiiiiiii LOST FILE

MESSAGES REQUIRING NO OPERATOR ACTION

JOB FINISHED

CHECK STARTED

END CHECK

TAPE ERRORS--: nnnnnnnn

BEGINNING - nnnnnnnn ENDING - nnnnnnnn

EXAMPLE OF AN EDIT RUN

FILE REORDER ROUTINE. READY MT 6:7

TYPE DATA AS FOLLOWS: MM-DD-YR 07-05-67
TYPE NUMBER OF DISCS IN OLD UNIT AS :DD: :16:
TYPE NUMBER OF DISCS IN NEW UNIT AS :DD: :16:
SET USER NUMBER IN I.B.-: NO (RET) (RET)
BEGINNING - 00327454 ENDING - 00377374

DISC ERROR AT 00025362
DISC ERROR AT 00025365
FILE BAD IB: 00025361

IB L: 00075304 USER 00000027 FILE NAME /ER/

IB L: 00025361 USER 00000107 FILE NAME /EVEN/

IB L: 00025044 USER 00000414 FILE NAME /MOON/

IB L: 00005615 USER 00000650 FILE NAME /RAB/

IB L: 00002231 USER 00000667 FILE NAME /PAYROLL/

IB OLD 00075304 USER 03300200 IB NEW 00026457
LOST FILE

IB OLD 00025361 USER 40000107 IB NEW 00062155
LOST FILE

IB OLD 00025044 USER 03027447 IB NEW 00051333
LOST FILE

IB OLD 00055615 USER 01627460 IB NEW 0006571333
LOST FILE

IB OLD 00002231 USER 0000000 IB NEW 00045140
LOST FILE

JOB FINISHED

ANY FILES TO CORRECT: YES (RET)

USER NUMBER: 0027
OLD INDEX BLOCK ADDRESS: 00075304
NEW INDEX BLOCK ADDRESS: 00025457
OK: YES (RET)

IB OK: 00075304 USER 00000027 FILE NAME /ER/

OK: YES (RET)

ANY FILES TO CORRECT: YES (RET)

(Repeat the above procedure for users 107, 414, 650 and 667)

ANY FILES TO CORRECT: NO (RET) (RET)

CHECK STARTED

IB CH:00026457 USER 00000027 FILE NAME /ER/
IB CH:00062155 USER 00000107 FILE NAME /EVEN/
IB CH:00051333 USER 00000414 FILE NAME /MOON/
IB CH:00065713 USER 00000650 FILE NAME /RAB/
IB CH:00045140 USER 00000667 FILE NAME /PAYROLL/

END CHECK

MAP PROGRAM

The Monitor assigns disc space for files by utilizing a disc bit map. Each bit in the map represents a 256 word block. If the bit is set, the block is available. The bit map is in the Monitor. The purpose of this program is to initialize the bit map, release the disc space occupied by bad files (files with invalid index block pointers, conflicting information, etc), and indicate in the user's file directory that the file has been deleted.

Note that MAP does not attempt to either re-allocate or optimize the disc files. This function is performed by the EDIT program. MAP simply reads each file directory in the system and initializes the bit map to reflect the current status of the disc.

The MAP program is run if a new version of the Monitor has been brought into core using the DSWAP program. When the Monitor is assembled the bit map contains all ones. The map must be changed to reflect the current disc environment. The MAP program is also run after the edit program has changed the positions of the files.

The disc map does not reflect all of the area of the disc. On an 8 or 16 disc system the bit map only represents arm positions 12 through 51. On a 24 or 32 disc system, arm positions 22 through 41 are mapped.

It is not advisable to restart the MAP program if any errors occur while it is running. The program will have partially initialized the bit map at this point. To start the program again would cause conflicts in the map. One should reload the Monitor system and then restart MAP.

In order to reduce disc arm movement the index block pointers are first collected from the file directories and sorted according to disc location. In the process of going through the file directories, a message is typed if the end-of-file directory flag is missing or if the index block pointer (BP) is invalid. The sorting is accomplished by first extracting off the high order six bits of the disc address of the IBP and packing the remaining bits with the user number into a single word (the user number occupies the low-order 12 bits). The user number would be used in case of an error to remove the file from the file directory. The high-order six bits are used as an index into 64 pockets, two pockets for each disc. The packed IBP and user number is sorted into the appropriate pocket by using strings of 128 word packets as necessary. Since the storage area consists of six pages, a total of 96 packets are available, allowing the storage of well over 10,000 index block pointers.

After the sort, the index block pointers are delivered to the Monitor using the BRS BE+5. If there is an error return, the program stores the IBP and the user number in an error list and sets an error flag. Also the program will type the IBP, the word returned by the Monitor in the A register (which is usually the disc address of the error), and the user number and then wait until a carriage return is typed by the operator before continuing.

After executing the BRS BE+5 for all IBPs, the error flag is checked. If there has been an error, the program types a message that the file deletion is about to start and waits for a carriage return to be typed before continuing. After receiving the carriage return, the program goes through the list of errors and releases the disc space occupied by the bad files. The MAP program does not delete the bad file entry from the customers file directory (FD). However, the file type (see bits 0 through 5 of word 2 of a FD entry) is set to 40B. Since 40B is not a legitimate file type, the file will be removed from the FD the next time the user's file directory is written (by LOGOUT, WRITE FD, etc.).

If there are no errors or when the last bad entry has been deleted, the program searches the accounting storage area for the next available sector, indicated by the first word of the sector being zero. The sector number of the accounting area is typed for the operator. The program then searches the file directory overflow area and determines the location of the next available overflow area and types it out. If there are less than 40 overflow areas left, a special message is typed giving the actual number of areas left. The disc address of the next available accounting area and the address of the next overflow area are delivered to the Monitor using the BRS BE+5 along with a termination flag. This completes the operation of the MAP program.

OPERATING INSTRUCTIONS

The program is a "GO TO" type program and the operator starts the program by typing:

-GO /MAP/ (RET)

The program responds by typing the following message and begins going through the file directories:

SYS. V MAP STARTED.

V is the MAP program version number.

No further action is required by the operator unless there is an error message. The program will type the following messages upon completion of building the map:

ACCOUNTING AT: aaa

OVERFLOW POINTER AT: bbb

END JOB

where "aaa" is an octal number corresponding to the accounting area sector (64 words) area. If "aaa" is 0 there is no accounting information stored. If it is 177 there is no remaining accounting storage area and the area must be cleared by using the CLEAR RECORDS command in the /OPER/ program. Note that the system must be taken down after clearing the accounting area so that the /MAP/ program can be rerun. The number "bbb" is the next available overflow area and normally varies from 1377 to 1100 on an eight-disc system and from 2377 to 2000 on a 16- or 32-disc system. Note that the first number (1377 and 2377) shows that no overflow directories are being used while the second number would show that no more overflow directories are available. The /OPER/ command GARBAGE must be executed before these second numbers are reached.

ERROR MESSAGES AND ACTION

FILE DIRECTORY END FLAG MISSING FOR USER: nnn

where "nnn" is the user number. The operator may:

1. Type a carriage return in which case the program will continue and the bad file directory will remain on the disc.

2. Stop the MAP program and use DDT to supply a valid end pointer to the file directory. The system must then be reloaded and MAP restarted.

CONFLICT AT ppppp qqqqqq nnn (RET)

where "ppppp" is the index block pointer and "qqqqqq" is the disc address of the data block that caused the conflict, and "nnn" is the user number (or overflow pointer) of the file directory that contains the conflict. A conflict implies that two or more files reference the same disc area. MAP will always map the first file and regard the second file as the conflict. The conflict is easily recognized because the

BRS BE+5 will attempt to reset a bit that is already reset. The operator may:

1. Type a carriage return in which case the program will continue after storing the IBP and user number in an error list.

2. Take the system down and reload the disc.

BAD INDEX BLK. PTR.: ppppp nnn (RET)

where "ppppp" is the bad index block pointer and "nnn" is the user number. This message indicates that the number "ppppp" is an illegal disc address. A disc address must be greater than 0 and less than or equal to LIBP :

LIBP:	= 37777B	8 Disc
	= 77777B	16 Disc
	= 137777B	24 Disc
	= 177777B	32 Disc

The operator may:

1. Type a carriage return in which case the program will continue and the bad pointer will remain in the file directory.

2. Stop the MAP program and use DDT to remove the file directory entry from the disc. If the file type is changed to 40B, the file will be deleted from the user's directory the next time his directory is written.

BAD FILE DELETION STARTING.

This message will only type if the error message "CONFLICT AT" has previously been typed by the program. At this point the bit map has been built and the program is ready to delete the bad files. The operator may:

1. Type a carriage return in which case the bad files will be deleted.

2. Take the system down and either reload the disc or start up again in the case of disc errors.

ONLY nn OVERFLOW F.D. AREAS LEFT

where nn is the actual number of overflow file directory areas left. This message is only typed if there is less than 40 areas left. The operator may:

1. Run the "GARBAGE" routine in the "OPER" program (see "OPER" instructions).

2. Ignore the message, hoping that the remaining areas will be sufficient until the GARBAGE routine can be run at a later time.

18. STRING PROCESSING SYSTEM (SPS)

A resident part of the system is a package of string handling outlines. These are discussed in detail in the second half of this manual. They are:

GCI	Get character and increment
WCI	Write character onto string
WCH	Write character onto string storage
SKSE	Skip on string equal
SKSG	Skip on string greater
GCD	Get character and decrement
WCD	Write character and decrement
BRS 5	Look up string in hash table
BRS 6	Insert string in hash table (must be preceded by BRS 5)
BRS 33	Input string
BRS 34	Output string given word address
BRS 35	Output string given string pointer
BRS 37	General command lookup

SPS includes symbol table lookup facilities, and a string storage data collector, available as a library routine. Strings are composed of 8-bit characters packed 3 per word and are addressed by 2-word string pointers. Two SYSPOPs formally part of SPS but useful in floating point operations and in general programming are:

LDP	Load pointer
STP	Store pointer

These are doubleword operations which load A and B from the effective address and the next location, or store A and B into the effective address and the next location.

String pointers are discussed under "Executive Treatment of Files". The general concept of manipulating characters that are packed three per word is discussed under "Echo Tables". String handling routines are discussed in detail under "String Processing".

19. FLOATING-POINT

Floating-point arithmetic is incorporated into the 940 system through the use of programmed operators. Floating-point hardware (referred to as "The Floating Point Arithmetic Unit (FPAU)") is available as an option. Conditional assembly of the Monitor allows the floating-point SYSPOPs to utilize the FPAU or to use software simulation. Three sets of floating-point SYSPOPs are included: standard SYSPOPs, Quick SYSPOPs, and FORTRAN II SYSPOPs. A detailed description of the SYSPOPs is available in Chapter 25.

STANDARD SYSPOPS

The floating-point numbers are normalized doubleword values. The first word is a sign bit followed by the 23-bit high-order part of the mantissa; the second word consists of the 15 low-order bits of the mantissa followed by a 9-bit exponent. Both words are two's complement. These SYSPOPs require that the result of any operation be returned in the A and B registers. If the FPAU is implemented, the current contents of the A and B registers are output to the FPAU, the operation performed, and the result copied to the A and B registers. The following SYSPOPs are implemented:

FAD	Floating-add
FSB	Floating-subtract
FMP	Floating-multiply
FDV	Floating-divide
BRS 21	Floating-negate
SKNF	Skip if floating-accumulator negative
LDP	Load double precision (FPAU not affected)
STP	Store double precision (FPAU not affected)

QUICK SYSPOPS

These SYSPOPs are included for use with the FPAU if it is not necessary to have the result of each operation returned in the A and B registers. Thus, a time saving is provided during successive floating-point operations. All of the SYSPOPs perform the operation with the current contents of the FPAU and leave the result of the operation in the FPAU. For compatibility, these SYSPOPs are also included if the FPAU is not implemented. In this case, the quick SYSPOPs behave identically to the standard SYSPOPs. The format of the floating-point words is the same as the standard SYSPOPs. The quick SYSPOPs include:

QFAD	Quick floating-add
QFSB	Quick floating-subtract
QFSI	Quick floating-subtract inverse
QFMP	Quick floating-multiply
QFDV	Quick floating-divide
QFDI	Quick floating-divide inverse
QFNA	Quick floating-negate
QLDF	Quick double precision load
QSTF	Quick double precision store
CAF	Copy A and B to FPAU
CFA	Copy FPAU to A and B

FORTRAN II SYSPOPS

These SYSPOPs are used by the FORTRAN II run time and library. FORTRAN II requires that the first floating word consist of the 15 low-order bits of the mantissa followed by a 9-bit exponent while the second word contain a sign bit followed by the 23 high-order bits of the mantissa. For ease in handling array variables, the SYSPOPs double the contents of the index register before performing the effective addrdss computation, and then restore the initial value of X. The following SYSPOPs are included:

FFAD	FORTRAN floating-add
FFSB	FORTRAN floating-subtract
FFMP	FORTRAN floating-multiply
FFDV	FORTRAN floating-divide
FLDF	FORTRAN load floating
FSTF	FORTRAN store floating

Consider the following example which uses the FORTRAN floating SYSPOPs to access array variables.

Example:

A and B are array variables dimensioned for ten elements: A (1) through A (10) and B (1) through B (10). Perform the following operation:

A (3) + B (2) ⟶ TEMP

```
            LDX     = 2
            FLDF    A,2
            LDX     = 1
            FFAD    B,2
            FSTF    TEMP
            :
A           BSS     20
B           BSS     20
TEMP        BSS     2
            :
            END
```

FLOATING-POINT OVERFLOW

If the floating-point operations are software simulated, esponent overflow will cause the CPU overflow indicator to be set. If the indicator is set by a floating-point operation, it will remain set until the user executes the "OVT" instruction.

If the Floating-Point Arithmetic Unit is implemented, exponent overflow will cause the overflow indicator in the FPAU to be set. The setting of the indicator forces an interrupt to occur. The interrupt routine will reset the FPAU overflow indicator, set a flag that indicates that overflow occurred, and arm the Monitor-to-user transition trap. When the transition trap routine is entered, the overflow flag is examined. If the flag is set, the CPU overflow indicator is set and software interrupt five, if armed, will be triggered. If the software interrupt is not armed, a return to the user program will occur. The user can issue the "OVT" instruction to test for overflow and reset the CPU overflow indicator.

INPUT/OUTPUT OF FLOATING-POINT NUMBERS

The BRS 52 and 53 can be used for the input/output of floating-point numbers to a file. Both of these BRSs require a format word as input. The description of the format word is given in Figure 33.

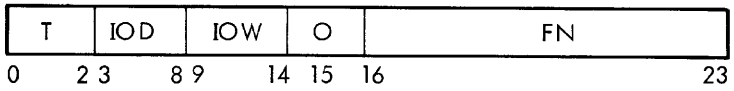
The BRS 52 requires a format specification in X, and inputs characters from the file number supplied in the format word. If the number input is floating-point, the skipping return is taken and the integer value is in the A register. Free form input will accept characters until a terminating character (any character except +, −, . , E, or digit) is input.

The BRS 53 requires a floating-point number in the A and B registers (the most significant fractional part in A and the least significant fractional part and exponent in B). The number will be output according to the format specified in X. If an error is detected during conversion, the index register on return will contain the error codes described in Table 7. If free form output is used, the number will be output in F16 format with five leading spaces if the exponent is less than 37B. For exponent values between 37B and 377B, E16.9 format is used.

The ISC and SIC SYSPOPs function similarly to the BRS 52 and 53 except that the values are not input/output to a file. SIC converts the characters in a given string and returns the value in the A (integer) or the A and B (floating-point) registers. SIC exits skipping if the value is floating-point. ISC converts the number input in the A and B registers and stores the characters into a designated character string.

Table 7. Error Conditions

Error No.	Error Type
X=0	No error was detected.
X=1	Number of decimal digits after the decimal point exceeds 12 for single precision and 18 for extended precision on formatted input. Twelve and 18 used respectively.
X=2	Field too short for E format on output. Overflow action will be taken depending on the value of bit 15 of the format word.
X=3	Input number exceeds the maximum allowable bounds.
X=4	Field too short for F or I format on output. Overflow action will be taken depending on the value of bit 15 of the format word.
X=5	An E format was specified for input but the input string does not contain an "E" or ".". The number will be converted using an equivalent F format.
X=6	An illegal character was encountered in the input scan. Character is ignored.



where

T Format Type

- 0 Illegal format type
- 1 Integer (I format)
- 2 E format with number right justified in field
- 3 F format with number right justified in field
- 4 E format with number left justified in field
- 5 F format with number left justified in field
- 6-7 Illegal format type

IOD Number of digits following the decimal point

IOW Total field width. If IOW is zero, free form output is used.

O Overflow action. If the field width is too small on output and this bit is set, the first character of the output field will be an asterisk. If this bit is reset and field width overflow occurs, characters on the right will be lost.

FN File number to be used for BRS 52 and 53. If a file number is not supplied, the teletype is assumed.

Figure 33. Format Word for Floating-Point Input/Output

20. SCHEDULING, FORKS AND PROGRAM INTERACTION

NUMBER: 78

NAME: SAIR

FUNCTION: Arm/Disarm Software Interrupts

STATUS: User

CALLING SEQUENCE:
```
LDA    M
BRS    78
```

M is the complete new interrupt mask.

DESCRIPTION: The new interrupt mask is substituted for the old one. A user may arm interrupt 1-10. An exec fork may arm interrupt 11 also. Interrupt 1 is in bit 4 of the mask word. The interrupts are as follows:

1 Interrupt if Program Panic (BRS 10 or escape)

2 Interrupt if Memory Panic

3 Interrupt if Lower Fork terminates

4 Interrupt if any I/O condition occurs which sets a flag bit (0, 7 or 8 in file number word)

5 Interrupt if FPAU overflow

6 through 10 interrupts on condition set by user

11 Interrupt if DSU error

Location 200B plus the interrupt number must be set to point to a routine to process the interrupt. When an interrupt occurs, the fork which has the interrupt armed is placed on one of the scheduled queues (QIO) with an activation code of 5 (see description of PTEST word in Chapter 2). When the scheduler activates this fork, the execution of a SBRM * to the interrupt location (201B-213B) is simulated. The SBRM * stores into the mark word the location where the fork was interrupted. Therefore, the user can return to the interrupted location by executing a BRU * on the saved mark word.

Example:

```
            LDA    =ESCAPE
            STA    201B
            LDA    =2B6
            BRS    78
             .
             .
ESCAPE      ZRO    MARK
             .
             .
            Process Interrupt
             .
             .
            BRU *  MARK
             .
MARK        ZRO
```

REGISTERS AFFECTED: None

NUMBER: 79

NAME: SIIR

FUNCTION: Cause Interrupt

STATUS: User

CALLING SEQUENCE:
```
LDA    N
BRS    79
```

N Interrupt number. N has the range of 6 to 10.

DESCRIPTION: Parallel forks in the structure are searched first and then higher forks. The interrupt will be caused in the first fork found which has the interrupt armed. If no fork has the interrupt armed, it is treated like a NOP. This would normally be used to cause interrupts 6 through 10 to interrupt.

REGISTERS AFFECTED: None

NUMBER: 49

NAME: SRIR

FUNCTION: Read Interrupts Armed

STATUS: User

CALLING SEQUENCE: BRS 49

DESCRIPTION: Reads the interrupt mask into the A register. Bit 4 corresponds to interrupt number 1, 5 to number 2, etc. There are 11 programmable interrupts. See BRS 78.

REGISTER AFFECTED: A

NUMBER: BE+12

NAME: TIMINT

FUNCTION: Interrupts a Fork After a Specified Period of Time.

STATUS: User

CALLING SEQUENCE:

```
LDA     M
LDB     T
LDX     N
BRS     BE+12
NORMAL  RETURN
```

M New interrupt mask.
T Time in milliseconds after which the fork will be interrupted.
N Interrupt number.

DESCRIPTION: The fork issuing this BRS will be interrupted after the delay if the interrupt specified by N is armed at that time. (Exception: The interrupt will be ignored if the fork is dismissed on a BRS 9 at the time of the interrupt.) If a fork gives this BRS again with the same N before the time has passed, the new time will be set. A fork may have a maximum of three timing interrupts pending simultaneously. See BRS 81.

REGISTERS AFFECTED: None

NUMBER: 90

NAME: DFR

FUNCTION: Declare a Fork for "Escape"

STATUS: User

CALLING SEQUENCE: BRS 90

DESCRIPTION: The PACT pointer of the fork that executes this BRS will be placed into location TTYASG (see Teletype Tables). If the user types "escape", this fork and all lower forks will be terminated. The fork above TTYASG will be activated. However, the Executive fork will never be terminated, even if TTYASG has been assigned to it.

REGISTERS AFFECTED: None

NUMBER: 46

NAME: NROUT

FUNCTION: Turn Escape Off

STATUS: System

CALLING SEQUENCE: BRS 46

DESCRIPTION: This BRS causes the NT bit (see PIM word of PAC table) to be set. If an escape occurs after this BRS has been executed, it will not be honored. However, the TP bit will be set (see PIM). If the TP bit is set when the user executes the BRS 47, the escape will then be honored. This scheme allows the first escape that occurs to be processed later and ignores any subsequent escapes.

A program running with escape turned off cannot be terminated by a higher fork.

See also, BRS 26 and 47.

REGISTERS AFFECTED: None

NUMBER: 47

NAME: SROUT

FUNCTION: Turn Escape On

STATUS: System

CALLING SEQUENCE: BRS 47

DESCRIPTION: This BRS reverses BRS 46; that is, reactivates the escape interrupt. If an escape occurred while in an off condition, the escape will now be processed.

REGISTERS AFFECTED: None

NUMBER: 26

NAME: SKROUT

FUNCTION: Skip if Escape Waiting

STATUS: System

CALLING SEQUENCE: BRS 26
EXCEPTION RETURN
NORMAL RETURN

DESCRIPTION: Checks for a stacked escape for this program and if there is one, transfers control to the "normal return" or, if not, to the "exception return". Significant only after BRS 46.

REGISTERS AFFECTED: None

NUMBER: 9

NAME: FKST

FUNCTION: Open Fork

STATUS: User

CALLING SEQUENCE: LDA T
BRS 9

T Address of a "Panic Table". (See format of Panic Table in Chapter 3).
Bits 0 through 5 of register A have the following significance:

0 Make fork system if current fork is system.

1 Set fork relabeling from panic table. Otherwise use current relabeling.

2 Propagate escape assignment to fork (see BRS 90).

3 Make fork fixed memory. It is not allowed any more memory than it started with.

4 Make fork local memory. New memory will be assigned to it independent of the controlling fork. (See section on "Memory Acquisition").

5 Make fork subsystem status if current fork is subsystem.

DESCRIPTION: BRS 9 is used to create a lower fork. The panic table indicated by register A must not be the same for two forks of the same fork or overlap a page boundary; if it is, BRS 9 is illegal. BRS 9 creates a new fork as a fork of the fork creating it, which is called the controlling fork. The fork is lower in hierarchy of forks than the controlling fork. The controlling fork may itself be a fork of some still higher fork.

When BRS 9 is executed by a user fork, the user fork is dismissed until the lower fork terminates. This has the same effect as issuing a BRS 31 immediately after a BRS 9. A user may not have more than eight forks in his fork structure. This includes the system fork and one fork for each system BRS that is active. Only one system BRS can be active.

REGISTERS AFFECTED: None

NUMBER: 57

NAME: CQO

FUNCTION: Guarantee 16 msec Computing

STATUS: User

CALLING SEQUENCE: BRS 57

DESCRIPTION: This BRS guarantees to the user upon return at least 16 msec. of uninterrupted computation. This is done by dismissing the user if less than 16 msec. remain in his time quantum.

This time will include some system overhead. Thus, if the time required is very close to 16 msec., a BRS 45 should be used. BRS 45 guarantees several times this amount.

REGISTERS AFFECTED: None

NUMBER: 30

NAME: FKRD

FUNCTION: Read Fork

STATUS: User

CALLING SEQUENCE:
```
LDA     P
BRS     30
```
P Panic Table Address

DESCRIPTION: Reads the current status of a lower fork into the panic table indicated by the A register. It does not influence the operation of the fork in any way.

REGISTERS AFFECTED: None

NUMBER: 107

NAME: FKRA

FUNCTION: Read All Fork Statuses

STATUS: User

CALLING SEQUENCE: BRS 107

DESCRIPTION: The status of all lower forks is recorded in the appropriate panic tables.

REGISTERS AFFECTED: None

NUMBER: 45

NAME: SQO

FUNCTION: Dismiss on Quantum Overflow

STATUS: User

CALLING SEQUENCE: BRS 45

DESCRIPTION: This BRS causes the user to be dismissed as though he had overflowed his long quantum. It guarantees that the next time he is started he will have a complete short time quantum. See BRS 57 to guarantee 16 msec.

REGISTERS AFFECTED: None

NUMBER: 72

NAME: EXDMS

FUNCTION: System Fork Dismissal

CALLING SEQUENCE: LDX N
BRS 72

N The number of the queue that the fork is to be put on.

DESCRIPTION: Dismisses a system fork and puts it on the specified queue. Returns to call +1 when recalled. The reactivation condition must be in the Monitor. This BRS is used to dismiss the Phantom User.

0 Teletype queue

1 I/O queue

2 Short time quantum queue

3 Long time quantum queue

REGISTERS AFFECTED: None

NUMBER: 81

NAME: WREAL

FUNCTION: Dismiss for Specified Amount of Time

STATUS: User

CALLING SEQUENCE: LDA T
BRS 81

T Dismissal time in milliseconds.

DESCRIPTION: The fork is dismissed for the number of milliseconds specified in A. See BE+12

REGISTERS AFFECTED: A

NUMBER: 31

NAME: FKWT

FUNCTION: Wait for Fork to Cause a Panic

STATUS: User

CALLING SEQUENCE: LDA P
BRS 31

P Panic Table Address

DESCRIPTION: Causes the controlling fork to be dismissed until the lower fork, or forks, causes a panic. When it does, the controlling fork is reactivated at the instruction following this BRS, and the panic table contains the status of the fork on its dismissal. The status is also put into the X register. The panic table address is put into the A register.

The controlling fork must have armed an interrupt or a lower fork must execute a BRS 10.

REGISTERS AFFECTED: X, A

NUMBER: 106

NAME: FKWA

FUNCTION: Wait for Any Fork to Terminate

STATUS: User

CALLING SEQUENCE: BRS 106

DESCRIPTION: Fork is dismissed until some lower fork terminates. When a lower fork terminates, the panic table address will be left in A.

REGISTERS AFFECTED: None.

NUMBER: 109

NAME: DMS

FUNCTION: Dismiss

STATUS: User

CALLING SEQUENCE: BRS 109

DESCRIPTION: The fork is dismissed. It can only be activated again by a program interrupt which has been armed by this fork or the termination of a lower fork.

REGISTERS AFFECTED: None.

NUMBER: 10

NAME: PPAN

FUNCTION: Programmed Panic. Terminates a Fork.

STATUS: User

CALLING SEQUENCE: BRS 10

DESCRIPTION: BRS 10 terminates the fork that issues it and returns control to the higher fork. It is like typing "escape" on the teletype. This condition can be distinguished from a panic caused by the escape key only by the fact that in the former case the program counter in the panic table points to a word containing BRS 10. This BRS would normally be used to terminate a fork when it is finished. The information in the panic table would, therefore, only be useful to a higher fork or to this fork if interrupt 4 has been armed by this fork.

REGISTERS AFFECTED: None

NUMBER: 32

NAME: FKTM

FUNCTION: Terminate a Fork

STATUS: User

CALLING SEQUENCE:
```
LDA     P
BRS     32
```
P Panic Table

DESCRIPTION: Causes a lower fork to be unconditionally terminated and its status to be stored into the panic table. The X register contains the status word upon return.

REGISTERS AFFECTED: X

NUMBER: 73

NAME: EPPAN

FUNCTION: Economy Panic

STATUS: User

CALLING SEQUENCE:
```
LDA     N
BRS     73
```
N Number of forks to terminate.

DESCRIPTION: This is like doing a BRS 10 for each of the forks specified. Forks are terminated going up until the Executive fork is reached or until N forks have been terminated.

REGISTERS AFFECTED: None.

NUMBER: 108

NAME: FKTA

FUNCTION: Terminates All Forks

STATUS: User

CALLING SEQUENCE: BRS 108

DESCRIPTION: All lower forks are terminated and their status read into the corresponding panic tables.

REGISTERS AFFECTED: None

21. INPUT/OUTPUT

NUMBER: 1

NAME: MONOPN

FUNCTION: Open a File of a Specific Device

STATUS: System

CALLING SEQUENCE:

```
LDA   ±I
LDB   ±L (BCDTAPE output only)
LDX   D
BRS   1
EXCEPTION RETURN
NORMAL RETURN
```

File number will be in register A on Normal Return.

I The relative address (DSU Address MOD 4) of the file's Index block for DSU files, or unit number for magnetic tape, otherwise anything. (I = 0 for a new output file since the Index Block address is unknown.)

\- Make the file read only.

\+ Make the file read/write.

D Device number.

L + for 80 char. records, - for 132 char. records.

Available device numbers are as follows:

1. Paper tape input.
2. Paper tape output.
3. BCD tape input.
4. Magnetic tape input.
5. Magnetic tape output.
6. Card input BCD.
7. Card input BIN.
8. Sequential DSU input.
9. Sequential DSU output.
10. BCD tape output.

(Continued on next page)

11. Line printer.
12. Card punch BCD
13. Card punch BIN

DESCRIPTION: The "open file" BRS is used to condition a file for input or output processing. If the file is successfully opened, control is transferred to the normal return with A continuing the file number; otherwise control is transferred to the exception return. Exception conditions are as follows:

1. Device or file in use or not available.
2. Too many files open.
3. No disc space left.

A file may be opened for input any number of times for the purpose of multiple user access or multiple processing by a single user. A file that is opened for output must be closed before it is opened. See also, BRSs 2, 3, 20, 82.

REGISTERS AFFECTED: A, X

NUMBER: 110

NAME: RDU

FUNCTION: Read Device and Unit

STATUS: User

CALLING SEQUENCE:
```
LDA     FILE No.
BRS     110
NORMAL RETURN
```

DESCRIPTION: Output X device number.
A unit number.

See BRS 1 for device number description.

REGISTERS AFFECTED: A, X

NUMBER: 2

NAME: MONCLS

FUNCTION: Close a File

STATUS: User

CALLING SEQUENCE:
```
LDA     N
BRS     2
NORMAL RETURN
```

N File number (obtained when file was opened).

DESCRIPTION: The "close file" BRS is used to indicate to the system all processing is completed on this file. All necessary termination processing will be completed and control will be transferred to the normal return. See BRS 20.

REGISTERS AFFECTED: None

NUMBER: 20

NAME: CFILE

FUNCTION: Close a File

STATUS: User

CALLING SEQUENCE:
```
LDA     N
BRS     20
```

N File Number

DESCRIPTION: The "close file" BRS is used to indicate to the system all processing is completed on this file. If the file number indicates magnetic tape, the file will be terminated and, if output, the End of File will be written; but in either case, the tape will be positioned at the start of the next file and the tape is de-allocated. All registers are changed.

REGISTERS AFFECTED: All

NUMBER: 8

NAME: IOH

FUNCTION: Close all Files

STATUS: User

CALLING SEQUENCE:
```
BRS     8
        NORMAL RETURN
```

DESCRIPTION: The "close all files" BRS is used to indicate the the system that all processing is completed on all files. The system will complete all necessary termination processing on all files and transfer control to the normal return. This BRS will not close magnetic tape files correctly. See BRS 2, 20, and 17.

REGISTERS AFFECTED: None

NUMBER: 66

NAME: DFDL

FUNCTION: Delete DSU File Data

STATUS: User

CALLING SEQUENCE:
```
LDA     N
BRS     66
        NORMAL RETURN
```

N File Number

DESCRIPTION: This BRS will return to available storage all DSU blocks which are assigned to the indicated file and clear the index block of DSU addresses. This BRS does not release the index block nor does it delete the file directory entry from the Customer File Directory.

REGISTERS AFFECTED: None

NUMBER: 67

NAME: DFER

FUNCTION: Delete a Specified Block of the DSU

STATUS: System

CALLING SEQUENCE:
```
LDA     D
BRS     67
        NORMAL RETURN
```

D Address of the DSU block.

DESCRIPTION: This BRS will return the DSU block indicated by the address in register A to available storage and transfers control to the normal return. This BRS should be used to delete Index Blocks. The BRS does not clear the Index Block address from the Customer File Directory, nor does it delete the file entry from the Customer File Directory.

REGISTERS AFFECTED: None

NUMBER: 87

NAME: DFRX

FUNCTION: Read DSU File Index Block

STATUS: System

CALLING SEQUENCE:
```
LDA     D
LDX     W
BRS     87
NORMAL RETURN
```

D DSU address of the index block (MOD 4)
W Core address into which the block is to be read.

DESCRIPTION: Reads the specified block into the given core location and transfers control to the normal return. The block read is the size of the currently defined index block. The size of an index block varies with the assembly.

REGISTERS AFFECTED: None

NUMBER: 104

NAME: RSYB

FUNCTION: Read a Page from the RAD

STATUS: System

CALLING SEQUENCE:
```
LDA     C
LDB     R
BRS     104
```

C Core Address

R RAD Address

DESCRIPTION: Reads one page from the RAD starting at the address R into a page in core. C may be any location in that page. The data will start in the first word of the page.

Uncorrectable RAD errors result in an instruction trap or interrupt 11 if it is armed. Try command again.

REGISTERS AFFECTED: None

NUMBER: 105

NAME: WSYB

FUNCTION: Write a Page on the RAD

STATUS: System

CALLING SEQUENCE:
```
LDA     C
LDB     R
BRS     105
NORMAL RETURN
```

DESCRIPTION: Writes one page on the RAD starting at the address R from a page in core. C may be any location in that page. The data will start in the first word of the page.

Uncorrectable RAD errors result in an instruction trap or interrupt 11, if it is armed. Try command again.

REGISTERS AFFECTED: None

NUMBER: 113

NAME: DFCD

FUNCTION: Compute File Size of a DSU File

STATUS: User

CALLING SEQUENCE:
```
LDA     File Number
BRS     113
NORMAL RETURN
```

DESCRIPTION: Adds the number of data words (in multiples of 255) in the file to the number in the X register. Returns the result in X.

REGISTERS AFFECTED: X

NUMBER; 118

NAME: TGET

FUNCTION: Allocate Magnetic Tape Unit

STATUS: System

CALLING SEQUENCE:
```
LDA     Tape Number
BRS     118
EXCEPTION RETURN
NORMAL RETURN
```

DESCRIPTION: Assigns the tape requested to the user. If the tape is already busy with someone else the exception return is executed.

REGISTERS AFFECTED: None

NUMBER: 119

NAME: TREL

FUNCTION: De-allocate Magnetic Tape Unit

STATUS: System

CALLING SEQUENCE:
```
LDA     Tape Number
BRS     119
NORMAL RETURN
```

DESCRIPTION: Releases the tape specified.

REGISTERS AFFECTED: None

NUMBER: BE+9

NAME: RDSYB

FUNCTION: Read DSU Page

STATUS: System

CALLING SEQUENCE:
```
LDA     C
LDB     R
BRS     BE+9
```

C Core Address
R Disc Address

DESCRIPTION: Use like 105. Can only be called by the Executive. BE+2 can be used to perform this function if less than a page is to be written.

REGISTERS AFFECTED: None

NUMBER: BE+10

NAME: WDSYB

FUNCTION: Write DSU Page

STATUS: System

CALLING SEQUENCE:
```
LDA     C
LDB     R
BRS     BE+10
```

C Core Address
R RAD Address

DESCRIPTION: Use like 104. Can only be called by the Executive. BE+1 can be used to perform this function if less than a page is to be read.

REGISTERS AFFECTED: None

NUMBER: BE+7

NAME: BPTEST

FUNCTION: Test a Breakpoint Switch

STATUS: System

CALLING SEQUENCE:
```
LDX     Switch Number
BRS     BE+7
SWITCH UP RETURN
SWITCH DOWN RETURN
```

DESCRIPTION: Tests the breakpoint switch (1, 2, 3, 4) indicated in X. If the switch is down, the BRS skips on return.

REGISTERS AFFECTED: None

NUMBER: BE+1

NAME: ARD

FUNCTION: Read DSU

STATUS: System

CALLING SEQUENCE:
```
LDA     Core Address
LDB     Disc Address
LDX     Number of Words
BRS     BE+1
NORMAL RETURN
```

DESCRIPTION: Reads up to 2K words from the disc. Transfer must not cross a page boundary and must be in multiples of 16 words. Errors result in an instruction trap or programmed interrupt 11, if it is armed. No two forks that are to run simultaneously should both use this BRS.

REGISTERS AFFECTED: None

NUMBER: BE+2

NAME: AWD

FUNCTION: Write DSU

STATUS: System

CALLING SEQUENCE:
```
LDA     Core Address
LDB     Disc Address
LDX     Number of Words
BRS     BE+2
```

DESCRIPTION: Like BRS BE+1. The number of words must be a multiple of 64.

REGISTERS AFFECTED: None

NUMBER: BE+15

NAME: RDPGE

FUNCTION: Read an SMT page from RAD

STATUS: SYSTEM

CALLING SEQUENCE:
```
LDA     N
BRS     BE+15
```

N SMT number

DESCRIPTION: Reads an SMT page from the RAD. The page must already be in memory. It returns the RAD address in B if a read occurs; otherwise there is no change. The purpose of this BRS is to read in another copy of the page in the event that the copy of the page in core has been altered.

REGISTER AFFECTED: B

NUMBER: BE+17

NAME: CKBUF

FUNCTION: Test for last buffer free

STATUS: User

CALLING SEQUENCE: BRS BE+17

DESCRIPTION: If a buffer in the user's TS block is available, the program continues. If not, an instruction trap will occur.

REGISTERS AFFECTED: None

NUMBER: BE+19

NAME: GTFDT

FUNCTION: Get creation date, access count for a file

STATUS: System

CALLING SEQUENCE:
```
LDA     L
BRS     BE+19
```

L Address in file directory hash table that corresponds to this file. (See contents of A register on normal return from BRS 37)

DESCRIPTION: Extracts the creation date and the access count for a file. A – creation date, B – access count.

REGISTERS AFFECTED: A and B

NUMBER: 15

NAME: GFN

FUNCTION: Reads Input File Name from a Command File and Looks up the File Name in the User's File Directory.

STATUS: User

CALLING SEQUENCE:
```
LDA     N
BRS     15
EXCEPTION RETURN
NORMAL RETURN
```

N Command File Number

DESCRIPTION: The routine ignores leading spaces, leading multi-blanks, and leading carriage return characters. It then uses the BRS 37 to look up the file name in the user's file directory hash table.† It returns in the registers for both returns exactly what BRS 37 puts there, which is:

Exception Return:	X	Pointer to the input file name string pointers
	A & B	Input file name string pointers
Normal Return:	A	Location of the file in the file directory hash table.
	B	The value woed of the hash table entry
	X	Changed

Note: The information contained in the registers cannot be used directly by the user since the addresses are in the TS Block; this BRS is normally followed by the BRS 16.

If the input file name string begins with a left parenthesis, or with the full quote, the file name will be located in another user's file directory or in the public file directory, respectively; in these cases, the input command file must be the teletype. Since the BRS 37 is not used in this case, the information in the registers is of no practical use to the user, and the BRS must be followed by the BRS 16 as indicated under the BRS 16.

REGISTERS AFFECTED: None

---

†The exception return is taken if the input file name string cannot be located in the file directory.

NUMBER: 16

NAME: GIFNB

FUNCTION: Open Input File in File Directory.

STATUS: User

CALLING SEQUENCE:
```
LDA     N
BRS     15
BRU     (Error)
BRS     16
EXCEPTION RETURN
NORMAL RETURN
```

N Command file number

DESCRIPTION: Opens an input file located in the user's file directory. BRS 16 requires in A the location of the first word of the entry in the file directory hash table. The exception return is taken if the pointer in A is not pointing to a proper location in the hash table, or if the file cannot be opened for any reason, such as a physical device that cannot be an input file. The file directory pointer may be obtained from BRSs 15, 48, and 60.

Exception Return:		All registers changed
Normal Return:	A:	File Number
	B:	File Type (0-4)
	X:	File Size

REGISTERS AFFECTED: All

NUMBER: 17

NAME: UABORT

FUNCTION: Close all Files (Including magnetic tape)

STATUS: User

CALLING SEQUENCE: BRS 17

DESCRIPTION: If magnetic tape has been used, the last record will be terminated and if output, the End of File will be written; in either case, the tape will be positioned at the start of the next file. The tape is then closed and the unit is de-allocated. See BRS 8. All registers are changed.

REGISTERS AFFECTED: All

NUMBER: 18

NAME: GOFNA

FUNCTION: Reads File Name from a Command File and Looks Up the File Name in the User's File Directory. The Command File Must Be an Input File.

STATUS: User

CALLING SEQUENCE:
```
LDA     N
BRS     18
EXCEPTION RETURN
NORMAL RETURN
```

N = Command File Number
Bit 1 = 1 of A Register = Assume a file a file name is correct and does not type "OLD FILE" or "NEW FILE".

DESCRIPTION: The routine ignores leading spaces, leading multiblanks, and leading carriage return characters. If the string begins and ends with a single quote or a slash, the string is terminated for look-up with this character and the string is looked up in the user's file directory using the BRS 5. A confirming carriage return must follow the quote or slash before the string is looked up. The exception exit is taken if the character is not a carriage return. If the string is found in the file directory hash table, the message "OLD FILE" is typed, otherwise the message "NEW FILE" is typed. If a confirming line feed, carriage return, or period is then next in the input string, the normal return will be taken, otherwise the exception return. In the case of a new file, the file name is inserted into the file directory.

If the string begins with a character other than a single quote or a slash, the string is looked up in the user's file directory using the BRS 37. If the string is not located, the error exit is immediately taken causing the exception return. The exception return will also be caused if the file is read only as indicated by the flag in the file directory. Normally, this BRS is followed by a BRS 19 which opens an output file.

Exception Return:		All registers changed.
Normal Return:	A:	Location of the file in the directory hash table.
	B:	Confirming character in case of a quote or slash file; otherwise, the file directory hash table value word.
	X:	Changed.

REGISTERS AFFECTED: All

NUMBER: 19

NAME: GOFNB

FUNCTION: Open Output File Located in File Directory

STATUS: User

CALLING SEQUENCE:
```
LDA     N1
LDB     N2 (For Tape Files Only)
LDX     N3
BRS     19
EXCEPTION RETURN
NORMAL RETURN
```

N1 Information supplied in A by BRS 18 (location in the file directory).
N2 File Size (for tape files only).
N3 File Type.

DESCRIPTION: Opens an output file located in the user's file directory. The information required in the register is indicated above. The word in A is checked for legality. If it is not a valid pointer, the exception return is taken. The exception return is also taken if the file cannot be opened for any reason, such as a physical device that cannot be used for output. In the case of a new file, the file directory entry is completed. If the new file is a disc file and it cannot be opened, the message "NO ROOM" is typed. The message "FILE TYPE WRONG" is typed as appropriate.

Exception Return:	All	changed.
Normal Return:	A	File Number.
	B & X	Changed.

REGISTERS AFFECTED: All

NUMBER: 48

NAME: GSFN

FUNCTION: Look up Input/Output File Name

STATUS: User

CALLING SEQUENCE:
```
LDP     N
BRS     48
EXCEPTION RETURN
NORMAL RETURN
```

N String pointers for the file name.

DESCRIPTION: The file name is looked up in the file directory hash table using the BRS 5. If it is not there, the exception return is taken.

Exception Return:	A & B:	No change.
	X:	Changed.
Normal Return:	A:	Location in file directory hash table. Can be used by BRS 16 or BRS 19.
	B:	Same as A (Location in hash table)
	X:	Changed.

REGISTERS AFFECTED: All

NUMBER: 60

NAME: GSFI

FUNCTION: Look Up Input/Output File Name and Insert if New.

STATUS: User

CALLING SEQUENCE:
```
LDP     N
BRS     60
EXCEPTION RETURN
NORMAL RETURN
```

N String pointers for the file name.

DESCRIPTION: The file name is looked up in the file directory hash table using the BRS 5. If it is not there, it is inserted in the hash table. The exception return is taken if it cannot be inserted in the case of a full directory.

Exception Return:	A & B:	No change.
	X:	Changed.
Normal Return:	A:	Location in file directory hash table.
	B:	Same as A (Location in hash table)
	X:	Changed.

REGISTERS AFFECTED: All

NAME: CIO

FUNCTION: Character Input/Output

STATUS: User

CALLING SEQUENCE:
```
LDA     C (Output only)
CIO     N
```

C 8 bit data character right justified.
N Address of word containing a file number.

DESCRIPTION: CIO is used to input or output a single character from, or to, a file from the A register. On input an End of Record or End of File condition will set bits 0 and 8 or bits 0 and 7 in the file number and return a 134_8 or 137_8 character, respectively. If interrupt 4 is armed (see BRS 78), it will occur. The End of Record occurs on the next input operation after the last character of the record has been input and the End of File condition occurs on the next input operation after the End of Record which signals the last record of the file. If an error occurs, bits 0 and 6 will be set in N and interrupt 4 will occur if it is armed.

WIO and BIO should not be mixed with CIO to read or write a given file.

REGISTERS AFFECTED: A

NAME: WIO

FUNCTION: Word Input/Output

STATUS: User

CALLING SEQUENCE:
```
LDA     D (Output only)
WIO     N
```

D Data word to be written.
N Address of word containing a file number.

DESCRIPTION: WIO is used to input or output a word of data to or from the A register. On input an End of Record condition returns a word of three 134B characters and sets bits 0 and 8 in the file number word. If interrupt 4 is armed (see BRS 78), it will occur. An End of File condition returns a word of three 137B characters and sets bits 0 and 7 in the file number word. If interrupt 4 is armed, it will occur. If an End of Record or File condition occurs with a partially filled out word, the word is completed with 134B or 137B characters. If an error occurs, bits 0 and 6 are set in N. If interrupt 4 is armed it will occur.

CIO and WIO should not be mixed to read or write a given file.

REGISTERS AFFECTED: A

NAME: BIO

FUNCTION: Blocked Input/Output

STATUS: User

CALLING SEQUENCE:
```
LDA     W
LDX     I
BIO     N
EXCEPTION RETURN
NORMAL RETURN
```

I Starting memory address.
W Number of words to be read or written.
N Address of word containing a file number.

DESCRIPTION: BIO is used to input a block of words to memory or output a block of words from memory. The A register will contain the first memory location unaffected at the end of the operation. If the operation is completed successfully, control will be transferred to the normal return, otherwise control will be transferred to the exception return.

On input an End of Record or End of File condition will set bits 0 and 8 or 0 and 7 respectively in the file number. An error will set bits 0 and 6. Interrupt 4 will occur if armed, when any of these bits are set. Exception conditions are end of record, end of file, and bad record.

If bit 1 is set in the Data Block disc address in the Index Block of a DSU file, it indicates the end of the data blocks and is the end of a logical record.

REGISTERS AFFECTED: A, X

NAME: CTRL

FUNCTION: Input/Output Control (for paper tape and magnetic tape only)

STATUS: System

CALLING SEQUENCE:
```
LDB     N1
LDA     C
CTRL    N
```

C Control number
N File number
N1 Number for control 3 or 4

DESCRIPTION: CTRL provides the following control functions for tape files;

Control	Description
1	Write end of record on output. Record count not used.
2	Backspace physical block.
3	Forward space (B) files.
4	Backspace (B) files.
5	Erase tape (output only) (3 inches).
6	Rewind.
7	Write EOF. Output only.
8	Long erase. Output only.

REGISTERS AFFECTED: None

22. TELETYPES

NUMBER: 23

NAME: LNKS

FUNCTION: Link Teletypes

STATUS: System

CALLING SEQUENCE:
```
LDA     T
BRS     23
EXCEPTION RETURN
NORMAL RETURN
```

T Teletype Number

DESCRIPTION: This BRS will link the controlling teletype with the teletype specified in the A register. The exception return will occur if: the teletype specified by T is already linked; is in the 8-level mode; does not have the accept message bit set; or has the XOFF, XON, or P bits set in TTYBL. If the exception return occurs, the A register will contain either the number of the teletype that is currently linking to the teletype specified by T or A will contain the TTYBL word for teletype T for all other conditions. The controlling teletype number will then be placed into bits 1 through 7 of the LCW word of teletype T.

If the normal return occurs, bit 0 of LCW of both teletypes will be reset and bits 18 through 23 will contain the linked teletype number.

REGISTER AFFECTED: A

NUMBER: 24

NAME: LNKC

FUNCTION: Break teletype link

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     24
```

T Teletype Number

The controlling teletype link with the teletype indicated by bits 18-23 of LCW is broken (i.e., bit 0 of the LCW word for both teletypes is set).

REGISTERS AFFECTED: None

NUMBER: 25

NAME: MSGS

FUNCTION: Accept/Refuse Messages (links)

STATUS: User

CALLING SEQUENCE:
```
LDX     T
LDA     I
BRS     25
```

T Any teletype number (-1 to indicate controlling teletype)

I Bit 22 set to accept messages (links)
Bit 22 reset to refuse messages (links)
Bit 23 set to accept input
Bit 23 reset to refuse input

DESCRIPTION: This BRS will set or reset bit 8 (the accept input bit) and/or bit 9 (the accept message bit) of the TTYBL word for the teletype indicated by the X register.

REGISTERS AFFECTED: None

NUMBER: 27

NAME: ASTT

FUNCTION: Attach TTY to this program

STATUS: System

CALLING SEQUENCE:
```
LDA     T
BRS     27
EXCEPTION RETURN
NORMAL RETURN
```

T Teletype Number

DESCRIPTION: To give total control over a teletype to the requesting program. If the indicated type is free, it is attached to the requesting program and transfers control to the "normal return". If it is not free, control is transferred to the "exception return".

REGISTERS AFFECTED: None

NOT IMPLEMENTED

NUMBER: 28

NAME: RSTT

FUNCTION: Release TTY

STATUS: System

CALLING SEQUENCE:
```
LDA     T
BRS     28
```

T Teletype Number

DESCRIPTION: Returns to a free status the teletype indicated by the A register. If the teletype was not attached to the requesting program a "panic" will be executed.

Note: All attached teletypes are released when the user logs out.

REGISTERS AFFECTED: None

NOT IMPLEMENTED

NUMBER: 11

NAME: CIB

FUNCTION: Clear the Teletype Input Buffer

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     11
```

T Teletype number (-1 is used to indicate the controlling teletype).

DESCRIPTION: Sets the buffer pointers to indicate there are no characters in the teletype input buffer.

REGISTERS AFFECTED: None

NUMBER: BE+6

NAME: TTYON

FUNCTION: Turns a Teletype Line On or Off.

STATUS: System

CALLING SEQUENCE:
```
LDA     =TTY no.
LDB     =0 (off) or -1 (on)
BRS     BE+6
NORMAL RETURN
```

DESCRIPTION: Issues the EOM and POT commands which cause the line to be turned off (hung up) or made ready to accept an incoming call.

REGISTERS AFFECTED: None

NUMBER: 12

NAME: CET

FUNCTION: Declare Echo Table

STATUS: User

CALLING SEQUENCE:
```
LDX     T
LDA     R
BRS     12
```

T Teletype number (-1 is used to indicate the controlling TTY).
R =0, 1, 2, or 3 to indicate the proper echo table. A user can request 8-level input by setting the sign bit of R and providing the terminal character in bits 16 through 23 of R.

DESCRIPTION: BRS 12 sets the echo table for the TTY indicated by register X. Echo tables are as follows:

0 Echo each character just as it was received and break on all characters.
1 Same echo as 0 but all characters except letters, digits and spaces are break characters.
2 Same echo as 0, but the only break characters are control characters (including carriage return and line feed).
3 No echo for any character and break on all characters.

The BRS 12 can be used to request 8-level input. While in 8-level mode, characters are not converted to internal format, echoes are not generated, escapes are not processed by the system (the escape character is placed into the user's teletype buffer), and the terminal character is regarded as the break character. As soon as the terminal character is received, the system reverts to echo table zero.

8-level output is always reset by any execution of the BRS 12.

REGISTERS AFFECTED: None

NUMBER: 29

NAME: COB

FUNCTION: Clear the Output Buffer

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     29
```

T Teletype Number (-1 indicates the controlling TTY).

DESCRIPTION: Sets the buffer pointers to indicate there are no characters in the teletype output buffer.

REGISTERS AFFECTED: None

NUMBER: 40

NAME: RDET

FUNCTION: Read Echo Table

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     40
```

T Teletype number

DESCRIPTION: Reads the echo table number (0, 1, 2, 3) into the A register.

If the teletype is not in 8-level input mode, reads the echo table number (0, 1, 2, 3) into the A register. If the teletype is in 8-level input mode, the sign bit of A is set, the address field contains the terminal character.

REGISTER AFFECTED: A

NUMBER: 13

NAME: SKI

FUNCTION: Test Input Buffer for Empty

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     13
EXCEPTION RETURN
NORMAL RETURN
```

T Teletype number (-1 is used to indicate the controlling TTY).

DESCRIPTION: This BRS tests for the presence of input characters in the buffer. If the buffer is empty, control is transferred to the "normal return". If there are any characters in the input buffer, control is transferred to the "exception return".

REGISTERS AFFECTED: None

NUMBER: 14

NAME: DOB

FUNCTION: Dismiss Until the Teletype Output Buffer is Empty

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     14
```

T Teletype number (-1 is used to indicate the controlling user TTY).

DESCRIPTION: Dismiss this fork until the teletype output buffer indicated is empty. It is dismissed only until the last character is transmitted. This BRS is useful anytime the user wishes to halt program flow until a message has been completely output. A fork using 8-level output should execute this BRS previous to the execution of a BRS 12 or BRS 86 to guarantee that the entire buffer is output in 8-level mode.

REGISTERS AFFECTED: None

NUMBER: 85

NAME: SET8P

FUNCTION: Set 8-level teletype output

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     85
```

T Teletype number (-1 is used to indicate controlling user TTY).

DESCRIPTION: Sets teletype to 8-level input/output mode. The teletype specified must either be the controlling teletype or an attached teletype. 8-level is transmitted to or from the teletype exactly as it is received from the user program.

REGISTERS AFFECTED: None

NUMBER: 86

NAME: CLR8P

FUNCTION: Reset 8-level teletype output

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     86
```

T Teletype number (-1 is used to indicate controlling user TTY).

DESCRIPTION: Restores teletype output to normal mode. The teletype specified must either be the controlling teletype or attached to it.

REGISTER AFFECTED: None

NAME: TCI

FUNCTION: Teletype Character Input

STATUS: User

CALLING SEQUENCE:
```
TCI     M
```

M Memory address

DESCRIPTION: This SYSPOP reads the character from the teletype input buffer and places it into location M right justified. The remainder of location M is cleared. The character is also placed in the A register right justified.

REGISTER AFFECTED: A

NAME: TCO

FUNCTION: Teletype Character Output

STATUS: User

CALLING SEQUENCE: TCO M

M Memory address

DESCRIPTION: This SYSPOP outputs the character from the right-most 8 bits of location M to the controlling teletype. In addition to the ordinary ASCII characters, all teletype output operations will accept 135B as a multiple blank character. The next character will be taken as a blank count, and the indicated number of blanks will be typed.

REGISTERS AFFECTED: None

NAME: OST

FUNCTION: Output to Specified Teletype

STATUS: User

CALLING SEQUENCE: OST T

T Teletype number

DESCRIPTION: OST is used to output a character in the A register to a specified teletype. This instruction is used for output to an attached teletype. Its accept message bit must be set or an illegal instruction panic will be generated.

REGISTERS AFFECTED: None

NOT IMPLEMENTED

23. MEMORY

NUMBER: 4

NAME: MPT

FUNCTION: Release a Page of Memory

STATUS: User

CALLING SEQUENCE:
```
LDA   N
BRS   4
```

N Contains any address in the page to be released.

DESCRIPTION: The PMT entry for the block is removed and the byte in any other fork which has this PMT byte. in its relabeling is cleared.

REGISTERS AFFECTED: None

NUMBER: 121

NAME: DPMTE

FUNCTION: Release Specified PMT Entry

STATUS: User

CALLING SEQUENCE:
```
LDA   R
BRS   121
```

R Relabeling byte

DESCRIPTION: Releases the specified page from the PMT. It is similar to a BRS 4 except that it takes a byte number instead of an address.

Instruction Trap:

1. Byte not in PMT.
2. A user fork tried to release a system page.

REGISTERS AFFECTED: None

NUMBER: 120

NAME: APMTE

FUNCTION: Assign PMT Entry

STATUS: System

CALLING SEQUENCE: LDA R
BRS 120

R Relabeling byte

DESCRIPTION: Obtains a new page for the relabeling byte specified. This BRS is used only in the recover routine in the executive.

Instruction Trap:

1. PMT entry is already assigned.
2. The relabeling byte number was not in the PMT.

REGISTERS AFFECTED: None

NUMBER: 43

NAME: RDRL

FUNCTION: Read Pseudo-Relabeling

STATUS: User

CALLING SEQUENCE: BRS 43

DESCRIPTION: Reads the current pseudo-relabeling registers into registers A and B.

REGISTERS AFFECTED: A, B

NUMBER: 44

NAME: STRL

FUNCTION: Set Pseudo-Relabeling

STATUS: User

CALLING SEQUENCE: LDA R1
LDB R2
BRS 44

R1 & R2 Relabeling factors

DESCRIPTION: This BRS takes the contents of registers A and B and stores them into the current pseudo-relabeling registers. It also causes the real relabeling to be reset to correspond to the new pseduo-relabeling.

If one page or less must be read from the RAD in order to satisfy the relabeling, the RAD I/O will occur immediately. If more than one page must be read, the new pseudo-relabeling will be stored into the PAC table, and the fork will be dismissed onto QIO with an immediate activation condition. When the fork is alloted a time-slice, it will be activated with the new pseudo-relabeling.

This BRS will result in an instruction trap for any of the following reasons:

1. Swapping in the new pages was not completed (usually because of a RAD failure).
2. The user tried to relabel over a system page.
3. The user tried to relabel over a page he did not have. (This is not the way to obtain more memory.)

REGISTERS AFFECTED: None

NUMBER: 116

NAME: RURL

FUNCTION: Read User Relabeling

STATUS: System

CALLING SEQUENCE: BRS 116

DESCRIPTION: Puts the program relabeling into A and B. This is what the system Executive uses as program relabeling. It is kept in the TS block.

REGISTERS AFFECTED: A, B

NUMBER: 117

NAME: SURL

FUNCTION: Set User Relabeling

STATUS: System

CALLING SEQUENCE:
```
LDA     RL1
LDB     RL2
BRS     117
```

RL1 and RL2 are the new values for the program relabeling.

DESCRIPTION: Sets the program relabeling as specified. This BRS is used by the system. User programs should use BRS 44 to set relabeling for a fork.

Instruction Trap:

1. A specified relabeling byte was not assigned.
2. A user fork tried to relabel a system byte.

REGISTERS AFFECTED: None

NUMBER: 122

NAME: MPAN

FUNCTION: Simulate Memory Panic

STATUS: System

CALLING SEQUENCE:
```
LDA     A
BRS     122
```

A Core address

DESCRIPTION: This BRS gets new memory for a class 3 BRS. If it succeeds the new memory is put into the relabeling of the calling program. Can be issued from a class 3 BRS only.

If a memory trap occurs, it looks to the calling program as if it came from a BRS instruction.

REGISTERS AFFECTED: None

NUMBER: 56

NAME: MBEX

FUNCTION: Make Page System

STATUS: System

CALLING SEQUENCE:
```
LDA     P
BRS     56
```

P Pseudo-Relabeling byte for page.
If bit 0 of A = 1, page will be made system.
If bit 0 of A = 0, page will be made not system.

DESCRIPTION: Sets the use mode of a page depending on the value of bit 0 in the A register.

Bit 0 of A is set to 1 if page was formerly system or 0 if it was not.

REGISTER AFFECTED: A

NUMBER: 80

NAME: MBRO

FUNCTION: Make Page Read Only

STATUS: User

CALLING SEQUENCE:
```
LDA     P
BRS     80
```

P PMT/SMT number
If bit 0 of A = 1, make page read only.
If bit 0 of A = 0, make page read-write.

DESCRIPTION: Sets the read-write status of the entry according to the value of A. An SMT entry can only be changed by a system fork. The former status of the entry is returned in A.

Instruction Trap:

1. Specified entry is not in use.
2. The swap failed.

REGISTER AFFECTED: A

NUMBER: BE+4

NAME: PEBRS

FUNCTION: Reads or Sets One Word in the Monitor

STATUS: System

CALLING SEQUENCE:
```
LDA     V
LDB     0 or -1
LDX     =Location in Monitor relabeling
BRS     BE+4
RETURN
```

V New value for word if it is to be set.
The contents of the location are returned in the A register.
If B is positive, the word is read.
If B is negative, the word is changed and the old value returned in A.

DESCRIPTION: Allows a system program to read or set the contents of any location in the monitor relabeling.

The original contents of the location are always returned in the A register.

REGISTER AFFECTED: A

NUMBER: 68

NAME: EBSM

FUNCTION: Enter Block in SMT

STATUS: System

CALLING SEQUENCE:
```
LDA     B
BRS     68
```

B Byte number in users pseudo-relabeling

DESCRIPTION: A free SMT entry is found and the PMT entry put into it. The SMT number is returned in A.

REGISTER AFFECTED: A

NOT IMPLEMENTED

NUMBER: 69

NAME: GBSM

FUNCTION: Get SMT Block to PMT

STATUS: Subsystem

CALLING SEQUENCE:
```
LDA     S
BRS     69
```

S SMT number

DESCRIPTION: Puts the SMT entry into the first free PMT entry. The PMT entry number is returned in A.

Instruction Trap:

1. A user program tries to relabel a system SMT entry.
2. The SMT number is not valid.

Memory Trap:

There were no free PMT entries.

REGISTER AFFECTED: A

24. STRING PROCESSING

NUMBER: 33

NAME: GETSTR

FUNCTION: Read String

STATUS: User

CALLING SEQUENCE:
```
LDA   A
LDB   T
LDX   T
BRS   33
```

A Address of string pointer
T Terminal character
F File number
Bit 0 of A set = The string is taken as null with the second pointer equal to the first.

DESCRIPTION: This BRS reads characters from the file and appends them to the string until the terminal character is reached. The terminal character is not appended to the string. It returns the updated string pointers in the A and B registers and updates the end string pointer in memory.

REGISTERS AFFECTED: A, B

NUMBER: 34

NAME: OUTMSG

FUNCTION: Output Message

STATUS: User

CALLING SEQUENCE:
```
LDX   F
LDA   W
LDB   C
BRS   34
```

F File number
W Beginning word address
C Character count or -1

DESCRIPTION: This BRS outputs C consecutive characters starting with the first character of the specified word. If B = -1, characters are output until a / is encountered; the character $ is interpreted as a carriage return and line feed.

REGISTERS AFFECTED: None

NUMBER: 35

NAME: OUTSTR

FUNCTION: Output String

STATUS: User

CALLING SEQUENCE:
```
LDX   F
LDA   P
LDB   P+1
BRS   35
```

F File number
P, P+1 A string pointer pair

DESCRIPTION: Outputs the string indicated by the string pointers in registers A and B to the specified file.

REGISTERS AFFECTED: None

NUMBER: BE+14

NAME:

FUNCTION: Input String with Edit

STATUS: User

NOT IMPLEMENTED

NAME: CIT

FUNCTION: Character Input and Test

STATUS: User

CALLING SEQUENCE:
```
LDA     N
CIT     F
EXCEPTION RETURN
NORMAL RETURN
```

N Character to be tested
F File Number (see CIO Input Only)

DESCRIPTION: The character in the A register is compared against the next character in the input file. If it compares, the normal return is taken and the character is removed from the input buffer. If it does not compare, the character is left in the input buffer and is returned in A. If the input buffer is empty the user will be dismissed until the next break character.

Exception Return:	A	The next character in the input buffer
	B & X	No change
N	A	The character supplied remains in A (the character is removed from the input buffer).

REGISTERS AFFECTED: A

NAME: SKSE

FUNCTION: Skip String Equal

STATUS: User

CALLING SEQUENCE:
```
LDA     B
LDB     E
SKSE    A
EXCEPTION RETURN
NORMAL RETURN
```

A Address of a string pointer pair
B Beginning string pointer
E End string pointer

DESCRIPTION: If the string addressed by the pointers in the A and B registers is identical with the string addressed by A of the calling sequence, control will be transferred to the normal return. Otherwise, control will be transferred to the exception return. If the strings are of different lengths or have different contents, control will be transferred to the exception return.

REGISTERS AFFECTED: None

NUMBER: 5

NAME: SSCH

FUNCTION: Look Up String in Hash Table

STATUS: User

CALLING SEQUENCE:
```
LDA     P
LDB     P+1
LDX     T
BRS     5
EXCEPTION RETURN
NORMAL RETURN
```

P and P+1 String pointers for a string to be looked up
T Address of a three-word table (see control section FDCTL of a hash table) with the format:

```
ZRO     Hash Table Beginning Address
ZRO     Hash Table End Address
ZRO     0     (working cell)
```

DESCRIPTION: BRS 5 searches the hash table for a string to match the string indicated by A and B registers. If successful, it returns in register B the address of the hash table string pointers (the location of the first entry of the three hash table entries), and in register A, the "hash value" (the third word of the hash table entry) and executes the normal return. Otherwise, it executes the "exception" return with registers A, B and X unchanged and the address of the next free hash table entry in word 3 of the table. (Word 3 will be -1 if the table is full.)

See BRS 6.

REGISTERS AFFECTED: A, B

NAME: LDP

FUNCTION: Load Double Precision

STATUS: User

CALLING SEQUENCE: LDP M

M Address of a double word

DESCRIPTION: The contents of memory location M are loaded into the A register and the contents of memory location M+1 are loaded into the B register. This SYSPOP can be used to load string pointers or floating point words. Note that LDP and STP do not affect the Floating Point Arithmetic Unit.

REGISTERS AFFECTED: A, B

NUMBER: 37

NAME: GSLOOK

FUNCTION: General String Lookup

STATUS: User

CALLING SEQUENCE:
```
LDA   F
LDB   S
LDX   T
BRS   37
EXCEPTION RETURN
NORMAL RETURN
```

F Input file number
S Address of string pointer pair
T Address of the Hash Table Control Table

DESCRIPTION: The hash table is scanned for a string to match the given one. If an exact match is found the normal return is taken. If the given string does not match the initial part of any hash table string, the exception return is taken. If the given string matches the initial part of some hash table string, characters from the input file are appended until the string is long enough either to determine a unique hash table string, with a matching initial part, or for no match to be possible; in which case the exception return is taken. In the case where a unique hash table string has been located, more characters are taken from input until an exact match is obtained, in which case the normal return is taken, or until the last character causes a mismatch. If the last character is alphanumeric, the exception return is taken since it is assumed that only a non-alphanumeric character, such as a space, carriage return, punctuation marks, etc., can be considered a proper terminator. The last character (which caused the mismatch) is left in the input file.

(Continued on next page)

Exits are as follows: (1) The exception return is taken on the nomatch condition with a string pointer in A. B points to the string so far collected. X is undisturbed. (2) The normal return is taken on a match with the address of a hash table string pointer in A and the "hash value" in B. X is undisturbed.

REGISTERS AFFECTED: None

NAME: STP

FUNCTION: Store Double Precision

STATUS: User

CALLING SEQUENCE: STP M

M Address of a doubleword

DESCRIPTION: The contents of register A are stored into location M and the contents of register B are stored into location M+1. This SYSPOP can be used to store string pointers or floating-point words. Note that LDP and STP do not affect the Floating Point Arithmetic Unit.

REGISTERS AFFECTED: None

NUMBER: 6

NAME: SSIN

FUNCTION: Insert String in Hash Table

STATUS: User

CALLING SEQUENCE: A, B, & X must have the output from BRS 5
```
BRS    6
```

DESCRIPTION: BRS 6 inserts the string pointer into the hash table at the point determined by the last BRS 5 which did not find a match. If the hash table is full (word 3 of the table pointed to by X is -1) an "Illegal Instruction" trap results. BRS 6 is intended for use in conjunction with BRS 5. It should be used only after BRS 5 has failed to find a match. Furthermore, string pointers should not be placed in the hash table in any manner other than with BRS 6 (otherwise the scanning algorithm used in BRS 5 may cause undesired results).

BRS 6 does not physically move the string to which registers A and B point. On return, register B contains the address of the first word of the new hash table entry and register A contains the "value" word of the entry.

REGISTERS AFFECTED: A, B

NAME: SKSG

FUNCTION: Skip on String Greater

STATUS: User

CALLING SEQUENCE:
```
LDA    B
LDB    E
SKSG   A
EXCEPTION RETURN
NORMAL RETURN
```

B Beginning string pointer
E End string pointer
A Address of a string pointer pair

DESCRIPTION: The SYSPOP compares the string indicated by A and B registers with the string indicated by A of the calling sequence, character by character and terminates with the first unequal character. The numerical internal code representation of characters is used to determine inequality. If the strings are unequal for the entire length of the shorter one, the longer one is indicated as greater. If the contents of the string addressed by the A and B registers is greater than the contents of the string addressed by A, control will be transferred to the normal return. Otherwise, control is transferred to the exception return.

REGISTERS AFFECTED: None

NAME: GCI

FUNCTION: Get Character and Increment

STATUS: User

CALLING SEQUENCE:
```
GCI    A
EXCEPTION RETURN
NORMAL RETURN
```

A Address of a string pointer pair

DESCRIPTION: This SYSPOP reads into the A register, the first character from the string indicated by the beginning string pointer given in the calling sequence. If the string is null or empty, nothing is done and control is transferred to the exception return. If the string is not null its first character is loaded into the A register right-justified, and the beginning string pointer is incremented by one such that the beginning string pointer now points to the string with the first character deleted. Control is transferred to the normal return. Unless a copy of the original pointer is saved, the contents of the string are effectively destroyed.

REGISTER AFFECTED: A

NAME: WCI

FUNCTION: Write Character and Increment

STATUS: User

CALLING SEQUENCE: WCI P

P Address of string pointer pair

DESCRIPTION: WCI writes the character in the A register on the end of the string addressed by the end string pointer. The end string pointer is incremented by 1.

REGISTER AFFECTED: B

NAME: GCD

FUNCTION: Get Character and Decrement

STATUS: User

CALLING SEQUENCE: GCD P
EXCEPTION RETURN
NORMAL RETURN

P Address of a string pointer pair

DESCRIPTION: A GCD is, in every way, similar to GCI except that the character is taken from the end of the specified string.

The last character on the string is loaded in the A register, and end string pointer is decremented so that it points to the previous character in the string. Control is transferred to the exception return if the end pointer is not greater than the beginning pointer before it is decremented.

REGISTER AFFECTED: N

NAME: WCD

FUNCTION: Writes Character and Decrement

STATUS: User

CALLING SEQUENCE: WCD P

P Address of a string pointer pair

DESCRIPTION: This SYSPOP writes the character in the A register on the beginning of the string and decrements the beginning string pointer.

REGISTERS AFFECTED: None

NAME: WCH

FUNCTION: Write Character

STATUS: User

CALLING SEQUENCE: LDA C
WCH T

C A character right-justified in the A register
T The address of a three word table. The table is as follows:

Word 0 A character address
Word 1 A character address
Word 2 A transfer address

DESCRIPTION: This SYSPOP tries to write a character into the area defined by the character addresses in the table. Provided that the second address in the table is greater than the first address, WCH will write the character in A register into the character position indicated by the first character address plus one and will increment the first character address in the table. If the first character address is equal to or greater than the second character in the table the character is not written and control is transferred to the third word of the table with A and X registers undisturbed and the address of the WCH in the B register. The address in the third word of the table can be an exit to a routine which allocates more memory or GARBAGE collects the remaining characters.

REGISTERS AFFECTED: None

25. NUMBERS

NUMBER: 36

NAME: OUTNUM

FUNCTION: Output Number

STATUS: User

CALLING SEQUENCE:
```
LDX   F
LDA   N
LDB   R
BRS   36
```

F File number
N Number to be output
R Radix

DESCRIPTION: Outputs a number in the radix R. The number will be output as an unsigned 24-bit integer. If the radix is less than 2, an instruction trap will be given.

REGISTERS AFFECTED: None

NUMBER: 38

NAME: GETNUM

FUNCTION: Read Number

STATUS: User

CALLING SEQUENCE:
```
LDX   F
LDB   R
BRS   38
```

F File number
R Radix

DESCRIPTION: Inputs an integer to any radix. The number may be preceded by a plus or minus sign. On exit the number will be in the A register. The conversion is terminated by any non-numeric character which will be in the B register on exit. The number is computed by multiplying the number obtained at each stage by the radix and adding the new digit.

REGISTERS AFFECTED: A, B

NUMBER: 52

NAME: FFI

FUNCTION: Formatted Input

STATUS: User

CALLING SEQUENCE:
```
LDX   FORMAT
BRS   52
BRU   X
```

DESCRIPTION: This routine reads characters from a file specified in the format word, FORMAT. FORMAT also specifies the format of the input. Free form input from the teletype results when FORMAT = 0. A skip return is generated if and only if (1) the input is free form, and (2) the input is floating point. The internal translation of the input file is stored in A, B. See Chapter 19 (Figure 33) for FORMAT description.

REGISTERS AFFECTED: A, B, X

NUMBER: 53

NAME: FFO

FUNCTION: Formatted Output

STATUS: User

CALLING SEQUENCE:
```
LDX   FORMAT
BRS   53
```

DESCRIPTION: The integer in A or the double word floating point value in A, B is output to the file according to the file number and format specified in FORMAT. See Chapter 19 (Figure 33) for FORMAT description.

REGISTERS AFFECTED: None

NAME: SIC

FUNCTION: String to Internal Conversion

STATUS: User

CALLING SEQUENCE:
```
LDX     FORMAT
SIC     POINTER
BRU     INTEGER
BRU     FLOATING
```

DESCRIPTION: The character string designated by the beginning string pointer (POINTER) is converted according to the FORMAT statement supplied in X. (See Figure 33 in Chapter 19 for discussion of FORMAT). If the number is integer, the normal return is taken, the value is returned in A, and B is zero. If the number is floating point, the skipping exit is taken, and the value is returned in the A and B registers. If an error occurs, an error code (described in Table 7 of chapter 19) is returned in X.

REGISTERS AFFECTED: A, B, X

NAME: ISC

FUNCTION: Converts Internal Numbers to Formatted Strings

STATUS: User

CALLING SEQUENCE:
```
LDX     FORMAT
LDP     VALUE
ISC     POINTER
```

DESCRIPTION: The value supplied in the A (integer) register or the A and B (floating point) registers is converted according to the format supplied in X (See figure 33 in Chapter 19 for discussion of FORMAT). Location POINTER should contain the beginning string pointer. On exit, location POINTER+1 will contain the ending string pointer. (See Chapter 12 for discussion of STRING POINTERS). If an error occurs during conversion, the X register contains an error code (see Table 7 in Chapter 19).

REGISTERS AFFECTED: A, B, X

NUMBER: 50

NAME: FFIX

FUNCTION: Conversion from Floating Point to Fixed Point

STATUS: User

CALLING SEQUENCE: BRS 50

DESCRIPTION: Fixes the double word floating point value in (A, B). The integer part is left in A. The fractional part is left adjusted in B.

REGISTERS AFFECTED: A, B

NUMBER: 51

NAME: FFLT

FUNCTION: Conversion from Fixed Point to Floating Point

STATUS: User

CALLING SEQUENCE: BRS 51

DESCRIPTION: The integer in A is converted to a normalized floating point value in A, B.

REGISTERS AFFECTED: A, B

NUMBER: 21

NAME: FNA

FUNCTION: Floating Negate

STATUS: User

CALLING SEQUENCE: BRS 21

DESCRIPTION: The contents of the A and B registers are output to the Floating Point Arithmetic Unit (FPAU). The FPAU is then negated. The result is copied from the FPAU to the A and B registers. If exponent overflow occurs, the overflow indication is set.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, the current contents of the A and B registers are negated. If overflow occurs, the CPU overflow indicator will be set.

NAME: FAD

FUNCTION: Floating Point Addition

STATUS: User

CALLING SEQUENCE: FAD M

M most significant fractional part of mantissa
M+1 least significant fractional part of mantissa and the exponent

DESCRIPTION: The contents of the A and B registers are output to the Floating Point Arithmetic Unit (FPAU). A floating addition is performed between the contents of memory locations M and M+1 and the FPAU. The result is copied from the FPAU and placed into the A and B registers. If exponent overflow occurs the overflow indicator is set.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, the following occurs:
(A, B) + (M, M+1) —→(A, B)
The result is left in the A and B registers. Exponent overflow will cause the CPU overflow indicator to be set.

NAME: FSB

FUNCTION: Floating Point Subtraction

STATUS: User

CALLING SEQUENCE: FSB M

M most significant fractional part of mantissa
M+1 least significant fractional part of mantissa and the exponent

DESCRIPTION: The contents of the A and B registers are output to the Floating Point Arithmetic Unit (FPAU). The contents of memory locations M and M+1 are subtracted form the FPAU. The result is copied from the FPAU to the A and B registers. If exponent overflow occurs, the overflow indicator is set.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, the following occurs:
(A, B) – (M, M+1 —→(A, B)
The result is left in the A and B registers. Exponent overflow causes the CPU overflow indicator to be set.

NAME: FMP

FUNCTION: Floating Point Multiplication

STATUS: User

CALLING SEQUENCE: FMP M

M most significant fractional part of mantissa
M+1 least significant fractional part of mantissa and the exponent

DESCRIPTION: The contents of the A and B registers are output to the Floating Point Arithmetic Unit (FPAU). The FPAU is multiplied by the contents of memory locations M and M+1. The result is copied f rom the FPAU to the A and B registers. If exponent overflow occurs, the overflow indicator will be set.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, the following occurs:
(A, B) x (M, M+1)—→(A, B)
The result is left in the A and B registers. Exponent overflow will cause the CPU overflow indicator to be set.

NAME: FDV

FUNCTION: Floating Point Divide

STATUS: User

CALLING SEQUENCE: FDV M

M most significant fractional part of mantissa
M+1 least significant fractional part of mantissa and the exponent

DESCRIPTION: The contents of the A and B registers are output to the FPAU. The FPAU is divided by the contents of memory locations M and M+1. The result is copied from the FPAU to the A and B registers. Exponent overflow will cause the overflow indicator to be set. An attempt to divide by zero will always cause the overflow indicator to be set.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, the following occurs:
(A, B) / (M, M+1 → (A, B)
The result is left in the A and B registers. Exponent overflow (or an attempt to divide by zero) will cause the CPU overflow indicator to be set.

NAME: SKNF

FUNCTION: Test Sign of Floating Accumulator

STATUS: User

CALLING SEQUENCE: SKNF
Accumulator Positive Return
Accumulator Negative Return

DESCRIPTION: If the Floating Point Arithmetic Unit (FPAU) is negative, the skipping return is taken. If the FPAU is positive, the next instruction in sequence is executed.

REGISTERS AFFECTED: None *

* If floating point hardware is not implemented, the SYSPOP executes as described above, except that the sign of the A register is tested.

NAME: QLDF

FUNCTION: Quick Load Floating

STATUS: User

CALLING SEQUENCE: QLDF M

M most significant fractional part of mantissa
M+1 least significant fractional part of mantissa and the exponent

DESCRIPTION: The contents of memory locations M and M+1 are loaded into the Floating Point Arithmetic Unit (FPAU) and into the A and B registers.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, this SYSPOP is identical to the LDP SYSPOP.

NAME: QSTF

FUNCTION: Quick Store Floating

STATUS: User

CALLING SEQUENCE: QSTF M

M most significant fractional part of mantissa
M+1 least significant fractional part of mantissa and the exponent

DESCRIPTION: The contents of the Floating Point Arithmetic Unit (FPAU) are copied into the A and B register and memory locations M and M+1. The A register will contain the most significant fractional part.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, this SYSPOP is identical to the STP SYSPOP.

NAME: QFNA

FUNCTION: Quick Floating Negate

STATUS: User

CALLING SEQUENCE: QFNA

DESCRIPTION: The current contents of the floating point arithmetic unit (FPAU) are negated. The result is left in the FPAU. Overflow will cause the overflow indicator to be set.

REGISTERS AFFECTED: FPAU *

* If floating point hardware is not implemented, this SYSPOP is identical to the BRS 21. The current contents of the A and B registers are negated. Overflow will cause the CPU overflow indicator to be set.

NAME: QFAD

FUNCTION: Quick Floating Point Addition

STATUS: User

CALLING SEQUENCE: QFAD M

M most significant fractional part
M+1 least significant fractional part and exponent

DESCRIPTION: A floating addition is performed between memory location M and M+1, and the current contents of the floating point arithmetic unit (FPAU). The answer is left in the FPAU. Exponent overflow causes the overflow indicator to be set.

REGISTERS AFFECTED: A, FPAU *

* If floating point hardware is not implemented, this SYSPOP is identical to the FAD SYSPOP. The result is left in the A and B registers. Exponent overflow causes the CPU overflow indicator to be set.

NAME: QFSB

FUNCTION: Quick Floating Point Subtract

STATUS: User

CALLING SEQUENCE: QFSB M

M most significant fractional part
M+1 least significant fractional part and exponent

DESCRIPTION: The contents of memory locations M and M+1 are subtracted (floating subtraction) from the current contents of the floating point arithmetic unit (FPAU). The answer is left in the FPAU. Exponent overflow causes the overflow indicator to be set.

REGISTERS AFFECTED: A, FPAU *

* If floating point hardware is not implemented, this SYSPOP is identical to the FSB SYSPOP. The result is left in the A and B registers. Exponent overflow causes the CPU overflow indicator to be set.

NAME: QFSI

FUNCTION: Quick Floating Point Subtract Inverse

STATUS: User

CALLING SEQUENCE: QFSI M

M most significant fractional part
M+1 least significant fractional part and exponent

DESCRIPTION: The current contents of the floating point arithmetic unit are subtracted from M and M+1. The result is left in the FPAU. Memory locations M and M+1 are not affected. Exponent overflow causes the overflow indicator to be set.

REGISTERS AFFECTED: A, FPAU *

* If floating point hardware is not implemented, the current contents of the A and B register are subtracted from the operand. The result is left in the A and B registers. Exponent overflow causes the CPU overflow indicator to be set.

NAME: QFMP

FUNCTION: Quick Floating Point Multiply

STATUS: User

CALLING SEQUENCE: QFMP M

M most significant fractional part
M+1 least significant fractional part and exponent

DESCRIPTION: The current contents of the floating point arithmetic unit (FPAU) is multiplied by the contents of memory locations M and M+1. The result is left in the FPAU. Exponent overflow causes the overflow indicator to be set.

REGISTERS AFFECTED: A, FPAU *

* If floating point hardware is not implemented, this SYSPOP is identical to the FMP SYSPOP. The result is left in the A and B registers. Exponent overflow causes the CPU overflow indicator to be set.

NAME: QFDV

FUNCTION: Quick Floating Point Divide

STATUS: User

CALLING SEQUENCE: QFDV M

M most significant fractional part
M+1 least significant fractional part and exponent

DESCRIPTION: The contents of the floating point arithmetic unit (FPAU) are divided by the operand (contents of memory locations M and M+1). The result is left in the FPAU. If exponent overflow occurs, the overflow indicator is set. An attempt to divide by zero always causes exponent overflow. If exponent overflow does not occur, the overflow indicator is reset.

REGISTERS AFFECTED: A, FPAU *

* If floating point hardware is not implemented, this SYSPOP is identical to the FDV SYSPOP. The result is left in the A and B registers. Overflow will cause the CPU overflow indicator to be set.

NAME: QFDI

FUNCTION: Quick Floating Divide Inverse

STATUS: User

CALLING SEQUENCE: QFDI M

M most significant fractional part
M+1 least significant fractional part and exponent

DESCRIPTION: This SYSPOP is identical to QFDV except that the operand (contents of memory locations M and M+1) is divided by the current contents of the Floating Point Arithmetic Unit. The result is left in the FPAU. Memory locations M and M+1 are not affected. Exponent overflow will cause the overflow indicator to be set.

REGISTERS AFFECTED: A, FPAU *

* If floating point hardware is not implemented, the operand is divided by the current contents of the A and B registers. The result is left in the A and B registers. If exponent overflow occurs the CPU overflow indicator will be set.

NAME: CFA

FUNCTION: Copy FPAU into A and B

STATUS: User

CALLING SEQUENCE: CFA

DESCRIPTION: The contents of the Floating Point Arithmetic Unit (FPAU) are copied into the A and B registers.

REGISTERS AFFECTED: A, B

* If floating point hardware is not implemented, the execution of this SYSPOP results in a NOP.

NAME: CAF

FUNCTION: Copy A and B to FPAU

STATUS: User

CALLING SEQUENCE: CAF

DESCRIPTION: The contents of the A and B registers are copied into the Floating Point Arithmetic Unit.

REGISTERS AFFECTED: FPAU*

* If floating point hardware is not implemented, the execution of this SYSPOP results in a NOP.

NAME: FFAD

FUNCTION: FORTRAN Floating Add

STATUS: User

CALLING SEQUENCE: FFAD N

N least significant fractional part of number and exponent
N+1 most significant fractional part of number

DESCRIPTION: The contents of the effective address represented by N and N+1 are added to the current contents of the Floating Point Arithmetic Unit. The result is left in the FPAU. For ease in addressing array variables, this SYSPOP doubles the contents of the X register before calculating the effective address. The X register is then restored. The A and B registers' contents are destroyed. Exponent overflow will cause the overflow indicator to be set.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, FFAD executes as described above, except that the result is left in the A and B registers. Exponent overflow will cause the CPU overflow indicator to be set.

NAME: FFSB

FUNCTION: FORTRAN Floating Subtract

STATUS: User

CALLING SEQUENCE: FFSB N

N least significant fractional part of number and exponent
N+1 most significant fractional part of number

DESCRIPTION: The contents of the effective address, represented by N and N+1, are subtracted from the current contents of the Floating Point Arithmetic Unit. The result is left in the FPAU. For ease in addressing array variables, this SYSPOP doubles the contents of the X register before calculating the effective address. The X register is then restored. The A and B registers' contents are destroyed. Exponent overflow will cause the overflow indicator to be set.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, FFSB executes as described above, except that the result is left in the A and B registers. Exponent overflow will cause the CPU overflow indicator to be set.

NAME: FFMP

FUNCTION: FORTRAN Floating Multiply

STATUS: User

CALLING SEQUENCE: FFMP N

N least significant fractional part of number and exponent
N+1 most significant fractional part of number

DESCRIPTION: The contents of the effective address, represented by N and N+1, are multiplied by the current contents of the Floating Point Arithmetic Unit. The result is left in the FPAU. For ease in addressing array variables, this SYSPOP doubles the contents of the X register before calculating the effective address. The X register is then restored. The A and B registers' contents are destroyed. Exponent overflow will cause the overflow indicator to be set.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, FFMP executes as described above, except that the result is left in the A and B registers. Exponent overflow will cause the CPU overflow indicator to be set.

NAME: FFDV

FUNCTION: FORTRAN Floating Divide

STATUS: User

CALLING SEQUENCE: FFDV N

N least significant fractional part of number and exponent
N+1 most significant fractional part of number

DESCRIPTION: The current contents of the Floating Point Arithmetic Unit (FPAU) are divided by the contents of the effective address. The result is left in the FPAU. For ease in addressing array variables, this SYSPOP doubles the contents of the X register before calculating the effective address. The X register is then restored. The A and B registers' contents are destroyed. Exponent overflow will cause the overflow indicator to be set.

REGISTERS AFFECTED: A, B, FPAU *

* If floating point hardware is not implemented, FFDV executes as described above except that the A and B registers are divided by the contents of the effective address. The result is left in the A and B registers. Exponent overflow, or an attempt to divide by zero, will cause the CPU overflow indicator to be set.

NAME: FSTF

FUNCTION: FORTRAN Store Floating

STATUS: User

CALLING SEQUENCE: FSTF N

N least significant fractional part of number and exponent
N+1 most significant fractional part of number

DESCRIPTION: The contents of the Floating Point Arithmetic Unit are stored into the effective address represented by N and N+1, and also copied into the A and B registers. For ease in addressing array variables, this SYSPOP doubles the contents of the X register before calculating the effective address. The X register is then restored. The A and B registers' contents are destroyed. Exponent overflow will cause the overflow indicator to be set.

REGISTERS AFFECTED: A*, B*, N, N+1

* If floating point hardware is not implemented, FSTF executes as described above, except that the contents of the A and B register are stored into the effective memory locations.

NAME: FLDF

FUNCTION: FORTRAN Load Floating

STATUS: User

CALLING SEQUENCE: FLDF N

N least significant fractional part of number and exponent
N+1 most significant fractional part of number

DESCRIPTION: The contents of the effective address are normalized and loaded into the Floating Point Arithmetic Unit and also into the A and B registers. For ease in addressing array variables, this SYSPOP doubles the contents of the X register before calculating the effective address. The X register is then restored.

REGISTERS AFFECTED: A, B, FPAU*

* If floating point hardware is not implemented, FLDF executes as described above except that only the A and B registers are loaded.

26. EXECUTIVE COMMAND OPERATIONS

NUMBER: 95

NAME: ECDUMP

FUNCTION: Dump

STATUS: User

CALLING SEQUENCE:
```
LDA     N
BRS     95
```

N File number

DESCRIPTION: This BRS writes the entire current state of the machine (user's program only) on the specified file, which is made type 4. The status of the pseudo-relabeling registers and all information necessary to restart the user from his current situation are written on the dump file so it can be restored by a recovery procedure. The only information not preserved are any shared memory entries which may be in the pseudo-relabeling.

Note: Dumps created by one system cannot be recovered by another.

REGISTERS AFFECTED: All

NUMBER: 96

NAME: ECRECV

FUNCTION: Recover

STATUS: User

CALLING SEQUENCE:
```
LDA     N
BRS     96
```

N File number

DESCRIPTION: This BRS reads the dump file written by a BRS 95 and recovers the machine status as it appeared at the time the dump was taken.

REGISTERS AFFECTED: All

27. MISCELLANEOUS OPERATIONS

NUMBER: 42

NAME: RREAL

FUNCTION: Read Real-Time Clock

STATUS: User

CALLING SEQUENCE:
```
BRS     42
```

DESCRIPTION: Read the real-time clock in the A register. Time is given as a 24-bit binary number representing 60ths of a second. The clock is set to zero when the system is started and it is incremented by one at every 1/60th second. A binary form of the month, date and start-up time is put in B. From A and B the user can calculate the month, date and time.

REGISTERS AFFECTED: A, B

NUMBER: 91

NAME: EXRTIM

FUNCTION: Read Data and Time into a String

STATUS: User

CALLING SEQUENCE:
```
LDA     S
LDB     S+1
BRS     91
```

S Beginning string pointer
S+1 Ending string pointer

DESCRIPTION: The current date and time are appended to the string provided in A and B registers and the resulting string pointers are returned in the A and B registers. The characters appended to the string have the form:

```
MM/dd hh:mm

    MM=Month
    dd =Day
    hh =Hours counted from 0 to 24
    mm =Minutes
```

REGISTERS AFFECTED: None

NUMBER: 88

NAME: RTEX

FUNCTION: Read Execution Time

STATUS: System

CALLING SEQUENCE: BRS 88

DESCRIPTION: Returns the execution time in A in 60 cycle clock ticks accumulated since log in.

REGISTER AFFECTED: A

NUMBER: 41

NAME: IORET

FUNCTION: Return from I/O Subroutine

STATUS: User

CALLING SEQUENCE: BRS 41

DESCRIPTION: This is used by the author of an I/O subroutine to return to the calling program.

REGISTER AFFECTED: A

NUMBER: 111

NAME: BRSRET

FUNCTION: Return from Class 3 BRS

STATUS: System

CALLING SEQUENCE: BRS 111

DESCRIPTION: This BRS is used only by the author of class 3 BRS's. It is the only normal termination of a class 3 BRS. If corresponds to a BRS 10 for other forks.

Instruction Trap:

BRS issued by a fork which was not a class 3 BRS.

REGISTERS AFFECTED: None

NUMBER: 112

NAME: TSOFF

FUNCTION: Turn Off Teletype Station

STATUS: System

CALLING SEQUENCE:
```
LDX     Job Number
BRS     112
```

DESCRIPTION: This BRS is known as suicide. The job disappears completely from the system.

The teletype line associated with the job will be set ready for another job if he merely logged out.

REGISTERS AFFECTED: All

NUMBER: 71

NAME: SKXEC

FUNCTION: Skip if System

STATUS: User

CALLING SEQUENCE: BRS 71

DESCRIPTION: The B register is set to the value of the use code which the user has set for the job. These values are:

Value of B	Use Code
1	Subsystem User
0	User
-1	Both
-2	System

The BRS skips if the B register is negative.

REGISTER AFFECTED: B

NUMBER: BE+5

NAME: SDBM

FUNCTION: Set Disc Bit Map

STATUS: System

CALLING SEQUENCE: LDA Address of X block Mod 4
BRS BE+5
EXCEPTION RETURN
NORMAL RETURN

Exception Return - A contains address that was in conflict.

DESCRIPTION: Turns off bits in the disc bit map for the X block and each data block referenced by the index block. If any conflicts occur (the bit is already off), the address is left in the A register and the exception return is taken. A conflict also increments one of two counters, XBERR or FDERR, for errors in the X block or the file directory respectively.

When the bit map has been set, one more call is made to this BRS with A negative. At that time a switch is set allowing output files to be opened; the new overflow pointer is set from B and the accounting area pointer is set from X.

REGISTER AFFECTED: A

NUMBER: BE+8

NAME: CRASH

FUNCTION: To Crash the System

STATUS: System

CALLING SEQUENCE: BRS BE+8
NO RETURN

DESCRIPTION: Saves the registers in SS01, SS02, SS03. Saves 0 in MCRO. Turns off the clock and disables the interrupts. Moves the TS block into real page 7 and the current relabeled page into real page 6.

REGISTER AFFECTED: None

NUMBER: BE+13

NAME: SETSW

FUNCTION: Sets System Exec Switches in COMPG file.

STATUS: System

CALLING SEQUENCE: LDA V
LDX N
BRS BE+13
NORMAL RETURN

V New switch value
N Switch number

DESCRIPTION: The switch is set to the new value and the old value is returned in A.

REGISTER AFFECTED: A

NUMBER: BE+16

NAME: MFSYS

FUNCTION: Set Executive -1

STATUS: SYSTEM

CALLING SEQUENCE:
```
LDA     =76543210B
BRS     BE+16
```

DESCRIPTION: Simulates execution of the Executive command -SET EXEC -1. Executive status (indicated by PAC table word PQU bit 0) is given to the fork that executes this BRS. The user must have either operator or subsystem status assigned to him in order to execute this BRS.

REGISTERS AFFECTED: NONE

NAME: EXS

FUNCTION: Execute Instruction in System Mode

STATUS: System

CALLING SEQUENCE: EXS I

I Address of the instruction to be executed

DESCRIPTION: This SYSPOP will cause the instruction pointed to by I to be executed in the system mode.

REGISTERS AFFECTED: Depends on instruction.

NAME: SBRM

FUNCTION: Reentrant Subroutine Branch

STATUS: User

CALLING SEQUENCE: SBRM SUB

DESCRIPTION: The SBRM is used to allow reentrant coding to branch to a subroutine and store the mark word into a temporary storage area. The subroutine returns to the main program by executing an SBRR SYSPOP, or a BRR to the mark word location.

Example:
```
TEMP    BSS     1       Location in temporary storage
          .
          .
MAIN    SBRM    SUB
          .
          .
SUB     ZRO     TEMP
          .
          .
        SBRR    SUB     or BRR      TEMP
```

Note that a branch occurs to the effective address of the SBRM plus one and the mark word is stored into the effective address indicated by the first location of the subroutine.

REGISTERS AFFECTED: P

NAME: SBRR

FUNCTION: Reentrant Subroutine Return Branch

STATUS: User

CALLING SEQUENCE: SBRR SUB

DESCRIPTION: The SBRR is used by a subroutine that was entered by execution of an SBRM. The SBRR functions similarly to a BRR*. See SBRM description.

REGISTERS AFFECTED: None

APPENDIX A. GLOSSARY OF TERMS

A

ACTPU: Phantom User Activation Counter. If positive when scheduler is entered, causes phantom user to be moved from QQE to QTI.

ACTR: I/O Activation Counter. ACTR is incremented each time a fork that is on QTI or QIO is ready. Set to -1 when the scheduler begins searching QSQ.

AUNN: Account And User Number. Indexed by job. Inactive contains 0.

B

BLK31: Flag used in W buffer interrupt routine.

- =0 When W channel is not in use
- =1 When disc is active
- = Address of the interrupt routine for the appropriate driver when any W buffer device (except disc) is active

breakpoint switch: Refers to the four toggle switches physically located on the computer console.

C

command file: The particular file from which the commands to the system Executive and subsystems are input. For teletype input the command file number is zero.

corresponding table: Contains file directory information. Each entry is 3 words. The relative position of the entry corresponds to the position found in the file directory hash table by the execution of a BRS 5. The corresponding table contains data about the file, e.g., file size, type, creation date, etc.

customer file directory: The names of all files for a particular user name are recorded in this directory.

D

DSU block: Four consecutive sectors on the disc whose beginning addresses are MOD 4. A block consists of 256 words.

DSU file: A file stored on the disc storage unit. Each file consists of an index block, and if the file contains data, then a sufficient number of DSU blocks to record the data.

DRQ: The disc queue. Each entry requires 3 locations.

DSWAP: Paper tape routine used at system initialization to fill the first 14K from disc to core.

DTXS1: Contains the count (minus 1) of the number of commands the disc driver wishes to add to disc queue.

DTXS2: Temporary location used by the disc software. Points to the location in DRQ where a command was just added, i.e., the value of EDCL before it was incremented by three.

E

EDCL: Points to the location in DRQ where the disc driver should add the next disc command. After a command is added, EDCL is incremented by three and wrapped around if necessary.

ETTB: Elapsed time table. One entry per job. TJOB points to the ETTB for the running job. One of the ETTB entries is incremented with each clock tick for the purpose of charging compute time.

F

file number: A file number is assigned by the system to files as they are opened. Also, there are fixed file numbers for certain devices. These are as follows:

- 0 Teletype Input
- 1 Teletype output
- 2 Nothing

file type: There are four standard file types. They are as follows:

1. File written by the system Executive as commanded by the "SAVE" command.
2. General binary file created by a subsystem, i.e., a FORTRAN object program.
3. Symbolic file.
4. Dump file.

FPLST: One word pointer to next free PAC table. 0 indicates all PAC tables in use.

FFLST: One word pointer to next available file number.

FULST: One word pointer to next free job number.

I

IDCL: Points to the command in DRQ that the interrupt routine will initiate when the interrupt for the IDCL1 command is received.

IDCL1: Pointer to the entry in DRQ for which I/O transmission is currently in progress.

IDMRET: Flag that =0 when the interrupt routine (IDM) is entered as a result of a subroutine call. =-1 when entered as a response to an interrupt.

index block: A DSU block (256 words) which contains the DSU addresses for all data blocks of a file.

INT31: Address of W buffer general interrupt routine. This routine handles all W buffer I/O except disc. This routine branches indirectly to address specified in BLK31.

J

JOB: Contains the JOB number of the running fork.

M

MAC: Number of unlocked pages minus 1. Used by swapper.

N

NCMEM: Number of entries in SMT table. (60B.)

NDCL: Contains the count (minus 1) of the number of commands that are on the disc queue and are ready for the disc interrupt routine to process. The interrupt routine will sequentially pull commands off DRQ and execute them until NDCL has the value -1. At this point there are no more commands on DRQ that are ready to be executed.

NPPAR: Number of entries in a PAC table.

NPUQ: Number of tasks that can be put on the Phantom User.

NTTY: Number of teletypes.

NUMEM: Number of entries in each PMT table. (20B.)

O

OVFP: Overflow file directory flag. -1 means that this user has not been assigned an overflow file directory. When an overflow file directory has been assigned, OVFP contains a pointer to a disc overflow file directory.

P

PAC table: Each fork is defined by a program active table. This table contains most of the information required to control selection, execution and interruption of the fork (additional information is stored in the user's TS page).

PACPTR: One word in Monitor that contains the PACT pointer of the currently active fork.

page: A page can exist on RAD, DSU or in-core memory but in all cases refers to 2048 words.

panic: A panic is a signal to the system to terminate execution of a fork.

panic, instruction: A panic caused by attempting to execute an instruction which cannot be executed in the user mode, such as a halt or device I/O instruction or a BRS which is not available to the user.

panic, memory: A panic caused by a fork attempting to address memory outside its range or write on memory which is set to read only.

panic table:

Word	
0	Program Counter
1	A Register
2	B Register
3	X Register
4	First Relabeling Register
5	Second Relabeling Register
6	Status

The status word may be:

-2	Dismissed for Input/Output
-1	Running
0	Dismissed on Escape or BRS 10
1	Dismissed on Illegal Instruction Panic
2	Dismissed on Memory Panic

A panic table must not overlap a page boundary.

PB: Table in TS page. 8 words long. Used for saving B register for corresponding fork. Indexed by fork number (XPB).

PIM: Word in PAC table. Contains interrupt mask, fork number, etc.

PL: Word in PACT where location counter is saved (P register).

PMT: Pseudo Memory Table. One PMT table per job. Pointed to by PMTP (which is indexed by job number). Each PMT table is 16 words long. The 16 words correspond to the users virtual 32K of memory. The pseudo-relabeling bytes have values 60B-77B.

PMTJOB: A location which contains the starting address of the current users PMT table using SMT as a reference. PMTJOB is used by the swapper (in conjunction with the pseudo-relabeling byte) to retrieve entries from PMT.

PMTP: One entry per job. Indexed by job number. Points to users PMT table.

PNEXT: Word in PACT that is used to chain the PAC tables when they are on the queues. If PNEXT is negative it contains a PACT pointer to the next PAC table on the queues. If PNEXT is positive it points to the next queue.

PPTR: Word in PAC table. Contains the up pointer (PFORK) and down pointer (PDOWN) to other PAC tables in its forking structure. If on free PACT list (PAC table not in use) has the absolute address of the next free PAC table.

PQU: Word in PAC table. Contains long quantum, Executive bits, etc.

PTAB: Word in PAC table. Contains job number, panic table, address, etc.

PTEST: Word in PAC table. Contains activation condition.

PUBPTR: Pointer to first task on phantom user queue.

PUCLST: Phantom user task activation test list. Indexed by test number. Dispatches to where the decision is made as to whether the phantom user is ready to perform this task.

PUCSET: Phantom user task activation list. Indexed by test number. Represents a dispatch list for the various tasks that the phantom user performs.

PUCT: The phantom user task queue. Each task queue entry consists of four words.

PUCTR: Shows count of the number of tasks put on the phantom user queue. When PUCTR=0 there are no tasks on the phantom user queue and he is dismissed to either QTI or QQE.

PUCTR1: When the system begins to search the PU task queue, PUCTR1 is set to PUCTR. If after the entire PUCT table is scanned, PUCTR1 is still equal to PUCTR, it indicates that the PU was not able to process any of his tasks and no interrupts occurred that placed a new task on PUCT. If this is the case, the PU is placed on QQE.

PUEPTR: Pointer to the last task on the phantom user queue.

PULIM: Phantom user limit. Limits the number of tasks that can be put on the PU queue. This parameter is established when the Monitor is assembled. Causes a crash if limit exceeded.

PX: Table in TS page. 8 words long. Used for saving X register for corresponding fork. Indexed by fork number (XPB).

Q

QIO: Queue of programs dismissed for I/O other than TTY I/O. Forks that are activated by an escape, software interrupt, or panic are also on QIO.

QQE: Queue of programs dismissed for exceeding their long quantum.

QSQ: Queue of programs dismissed when short quantum has expired and other programs on QTI or QTO are ready to run.

QTI: Queue of programs dismissed for TTY input/output.

quantum, long time: The maximum length of time a fork can run before the scheduler checks for other forks to be run.

quantum, short time: The minimum length of time a fork will run before the scheduler checks for other forks to be run which were dismissed for I/O.

R

Real: Real-time counter. Incremented by clock interrupt routine. Initialized when system is brought up.

Relabeling, pseudo: See format of relabeling registers. Each byte points to an SMT or PMT entry.

relabeling registers: The relabeling registers are used to indicate a page number which has been assigned to a use for a particular logical page. They are of the form:

First word	Page 0	Page 1	Page 2	Page 3
Second word	Page 4	Page 5	Page 6	Page 7

RLTS: Contains real page number of running forks temporary storage (TS) page.

RL1: Word in PAC table. Contains the first pseudo relabeling word for the fork.

RL2: Word in PAC table. Contains the second pseudo relabeling word for the fork.

RL3: Pseudo relabeling table for TS page. Indexed by job number.

RMC: Real memory count table. Indexed by real page number. Contains a -1 if page is unlocked. Contains a value greater then -1 if page is locked.

RMT: Real Memory Table. One entry for each page of memory. Points to the PMT or SMT entry responsible for having this page in core.

RRL1: Contains the real relabeling for register 1. This word is potted out to the hardware register.

RRL2: Applies to relabeling register 2. See RRL1.

RRL3: Applies to Monitor relabeling register. See RRL1.

S

SMT: Shared Memory Table. Only one SMT in the system. 60B words long. Indexed by pseudo-relabeling values 0-57B. The reentrant programs have entries in SMT.

SSRL1: Two words in TS page that indicate the subsystem used. The pseudo-relabeling for the system is fetched from the subsystem corresponding table and loaded into these two words.

string pointers: A pair of pointers which contain a character address of the character before the first character of a string and a character address of the last character of the string.

SWOFF: Word in TS used by Executive. -1 means user has logged on or is in the process of logging on. Used by Executive to determine whether $ dump should be effected.

SWTM: Word in TS used by Executive to determine log on status of user.

=0 means time ran out while user is logging on (1.5 mins)

=-1 user is in the process of logging on or has logged on

SYSTL: Word in TS block which contains the hash table address of the subsystem in use.

T

TIIS5: Indexed by channel number. Used when software is processing carriage returns and line feeds.

=0 if last character output was not a CR or LF

Bit 23=1 when software has sent a CR

Bit 23=0 when software has sent a LF

TIME: Contains the short quantum for the running fork.

TJOB: Word which points to job time counter table (ETTB). Used to increment compute time for a job. When the clock interrupt occurs. A MIN *TJOB is performed and the running job is charged. See ETTB.

TTIME: Word where the total time (long time quantum) is maintained for the running fork.

TTNO: Contains TTY channel number. Indexed it by job number. If active, TTNO contains channel number. If inactive, contains the chain for the free job numbers.

TTYASG: TTY assigned table. Indexed it by channel number. If active, it contains the PACTPTR of the fork to terminate in case of rubout. If inactive, contains 37777B.

U

UNO: Set to user number when a user has logged on the system.

User Name: The alphanumeric characters the user inputs after typing in the password and semicolon or CR. The name can be a maximum of 12 characters and may contain any character except semicolon, right paren, or CR. A unique user number is associated with each user name. A user name must only be unique within an account.

User Number: A 4-digit octal number which is unique to each user name. The user number is assigned by the operator. The user number is a pointer to the file directory associated with a particular user name.

UTTY: One word in the system which contains the channel number of the running fork.

W

WERIS: State of the teletype line. Indexed by channel number.

-1 Line free
0 User is in the process of logging on
>0 User number for the user on this teletype

APPENDIX B. BRS AND SYSPOP INDEXES

INDEX OF BRS'S AND SYSPOP'S BY NUMBER

†Class 3 (Executive) BRS
††Class 2 BRS

BRSs	Function	Page
66	Delete DSU file data	93
67	Delete DSU file index block	93
69	Get SMT block to PMT	125
71	Skip if in system	156
72	System dismissal	84
73	Terminates a specified number of lower forks	87
78	Arm/disarm software interrupts	76
79	Cause specified software interrupts	77
80	Make page read only	123
81	Dismiss for specified amount of time	84
85	Set special TTY output	116
86	Clear special TTY output	117
87	Read DSU file index block	94
88	Read execution time	154
90	Declare a fork for escape	79
91[t]	Read date and time into a string	153
95[t]	Dump program and status on file	152
96[t]	Recover program and status from file	152
104	Read a page (2048 words) from RAD	94
105	Write a page (2048 words) to RAD	95
106	Wait for any fork to terminate	85
107	Read status of all lower forks	83
108	Terminate all lower forks	88
109	Dismiss calling fork	86
110	Read device and unit	91
111	Return from exec BRS (exec only)	155
112	Turn off teletype station (exec only)	155
113	Compute file size of a disc file	95
116	Read user relabeling	121
117	Set user relabeling	122
118	Allocate magnetic tape unit	96
119	De-allocate magnetic tape unit	96
120	Assign PMT entry	120
121	Release specified page from PMT	119
122	Simulate memory panic	122
BE+1	Read DSU	98
BE+2	Write DSU	99
BE+3	Test for carrier present (not implemented)	
BE+4	Read/write one word in the Monitor	124
BE+5	Set disc bit map	156
BE+6	Turn a teletype line on or off	112
BE+7	Test a breakpoint switch	98
BE+8	To crash the system for error diagnostic	157
BE+9	Read DSU page	97
BE+10	Write DSU page	97
BE+11	Ignore line feed or carriage return (not implemented)	
BE+12	Arm timing interrupt	78
BE+13	Sets system Executive switches in COMPG	157
BE+15	Read SMT page from RAD	99
BE+16	Set EXEC =-1	158
BE+17	Test if last buffer used	100
BE+19[t]	Get file creation date and access count	100

[t]Class 3 (Executive) BRS

INDEX OF BRS'S AND SYSPOP'S BY TYPE

SCHEDULING, FORKS AND PROGRAM INTERACTION

PROGRAM INTERRUPTS

CONTROL OF THE ESCAPE KEY

BRSs or SYSPOPs	Function	Page
26	Skip if escape waiting	80
46	Turn escape off	79
47	Turn escape on	80
90	Declare a fork for escape	79

ACTIVATION OF FORKS

BRSs or SYSPOPs	Function	Page
9	Open fork	81
57	Guarantee 16 ms computing	82

INTERROGATION OF A FORK

BRSs or SYSPOPs	Function	Page
30	Read status of a lower fork	82
107	Read status of all lower forks	83

TEMPORARY SUSPENSION OF FORKS

BRSs or SYSPOPs	Function	Page
45	Dismiss on quantum overflow	83
72	Executive dismissal	84
81	Dismiss for specified amount of time	84
31	Wait for specific fork to cause a panic	85
106	Wait for any fork to terminate	85
109	Dismiss calling fork	86

TERMINATION OF A FORK

BRSs or SYSPOPs	Function	Page
10	Terminates the calling fork	86
32	Terminates a specified lower fork	87
73	Terminates a specified number of lower forks	87
108	Terminate all lower forks	88

INPUT/OUTPUT

DIRECT CONTROL OF PERIPHERALS

BRSs or SYSPOPs	Function	Page
1	Open a file of a specific device	89
2	Close a file	91
8	Close all files	92
20	Close a tape file	92
66	Delete DSU file data	93
67	Delete DSU file index block	93
87	Read DSU file index block	94
104	Read a page (2048 words) from RAD	94
105	Write a page (2048 words) to RAD	95
110	Read device and unit	91
113	Compute file size of a disc file	95

DIRECT CONTROL OF PERIPHERALS (cont'd.)

BRSs or SYSPOPs	Function	Page
118	Allocate magnetic tape unit	96
119	Deallocate magnetic tape unit	96
BE+1	Read DSU	98
BE+2	Write DSU	99
BE+7	Test a breakpoint switch	98
BE+9	Read DSU page	97
BE+10	Write DSU page	97
BE+15	Read SMT page from RAD	99
BE+17	Test for last buffer used	100

CONTROL OF FILES VIA FILE NAMES

BRSs or SYSPOPs	Function	Page
15	Read input file name	101
16	Open input file in file directory	102
17	Close all files	102
18	Read a file name and look it up in the file directory	103
19	Open output file located in file directory	104
48	Look up input/output file name	105
60	Look up I/O file name and insert in file directory if not found	105
BE+19	Read file creation date and access count	100

I/O OPERATIONS

BRSs or SYSPOPs	Function	Page
BIO	Block input/output	107
CIO	Character input/output	106
CTRL	Input/output control (tape)	108
WIO	Word input/output	106

TELETYPE OPERATIONS

LINKING AND ATTACHING

BRSs or SYSPOPs	Function	Page
23	Link/unlink specified TTY	109
24	Unlink all TTYs	109
25	Set teletype to accept/refuse links	110
BE+3	Test for carrier present (not implemented)	
BE+6	Turn a teletype line on or off	112

INPUT/OUTPUT OPERATIONS

BRSs or SYSPOPs	Function	Page
11	Clear the teletype input buffer	112
12	Declare echo table	113
13	Test input buffer for empty	115
14	Delay until the TTY output buffer is empty	116
29	Clear the output buffer	114
40	Read echo table	115
85	Set special TTY output	116
86	Clear special TTY output	117
BE+11	Ignore line feed or carriage return when followed by same (not implemented)	
OST	Output to specific TTY (not implemented)	118
TCI	Teletype character input	117
TCO	Teletype character output	118

MEMORY OPERATIONS

PRIVATE MEMORY

BRSs or SYSPOPs	Function	Page
4	Release a page of memory	119
43	Read pseudo relabeling	120
44	Set pseudo relabeling	121
56	Make page Executive	123
80	Make page read only	123
116	Read user relabeling	121
117	Set user relabeling	122
120	Acquire a new page	120
121	Release specified page from PMT	119
122	Simulate memory panic	122
BE+4	Read/write one word in the Monitor	124

SHARED MEMORY

BRSs or SYSPOPs	Function	Page
69	Get SMT block to PMT	125

STRING PROCESS

STRING I/O

BRSs or SYSPOPs	Function	Page
33	Read string	126
34	Output message	126
35	Output string	127
CIT	Character input and test	128

HASH TABLE SEARCH

BRSs or SYSPOPs	Function	Page
5	Look up string in hash table	129
6	General string look up	132
37	Insert string in hash table	130

STRING MANIPULATION

BRSs or SYSPOPs	Function	Page
LDP	Load string pointer	130
SKSE	Skip if string equal	128
SKSG	Skip if string greater	133
STP	Store string pointer	131

CHARACTER MANIPULATION

BRSs or SYSPOPs	Function	Page
GCI	Get character from beginning of string and increment beginning pointer	133
GCD	Get character from end of string and decrement end pointer	134
WCD	Put character on beginning of string and decrement beginning pointer	135
WCH	Write character to memory by table	135
WCI	Put character on end of string and increment end pointer	134

NUMBER OPERATIONS

NUMBER I/O

BRSs or SYSPOPs	Function	Page
36	Output number to specified radix	136
38	Input number to specified radix	136
52	Formatted floating point input	137
53	Formatted floating point output	137
ISC	Internal to string conversion	138
SIC	String to internal conversion	138

ARITHMETIC OPERATIONS

BRSs or SYSPOPs	Function	Page
21	Floating point negate	140
50	Conversion from floating point to fixed point	139
51	Conversion from fixed point to floating point	139
CAF	Copy A and B into FPAU	148
CFA	Copy FPAU into A and B	147
FAD	Floating point addition	140
FDV	Floating point division	142
FFAD	FORTRAN floating add	148
FFDV	FORTRAN floating divide	150
FFMP	FORTRAN floating multiply	149
FFSB	FORTRAN floating subtract	149
FLDF	FORTRAN load floating	151
FMP	Floating point multiplication	141
FSB	Floating point subtract	141
FSTF	FORTRAN store floating	150
QFAD	Quick floating add	144
QFDI	Quick floating divide inverse	147
QFDV	Quick floating divide	146
QFMP	Quick floating multiply	146
QFNA	Quick floating negate	144
QFSB	Quick floating subtract	145
QFSI	Quick floating subtract inverse	145
QLDF	Quick load floating	143
QSTF	Quick store floating	143
SKNF	Skip on negative floating accumulator	142

EXECUTIVE COMMAND OPERATIONS

BRSs or SYSPOPs	Function	Page
95	Dump program and status on file	152
96	Recover program and status from file	152

MISCELLANEOUS OPERATIONS

BRSs or SYSPOPs	Function	Page
41	Return from I/O subroutine	154
42	Read real-time clock	153

MISCELLANEOUS OPERATIONS (cont.)

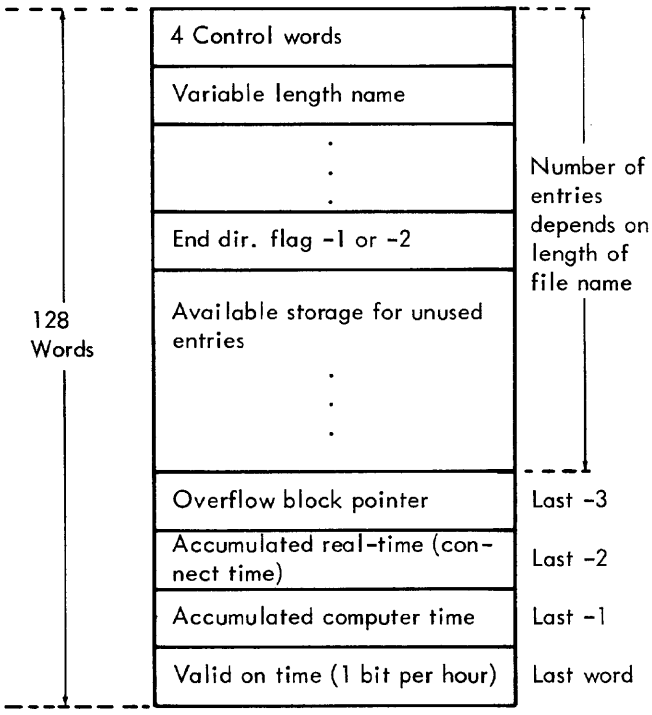
BRSs or SYSPOPs	Function	Page
71	Skip if Executive	156
88	Read execution time	154
91	Read date and time into a string	153
111	Return from	155
112	Turn off teletype station (Executive only)	155
BE+5	Set disc bit map	156
BE+8	To crash the system for error diagnostic	157
BE+13	Sets Executive switches in COMPG	157
BE+16	Set Executive status	158
EXS	Execute instruction in system mode	158
SBRM	Reentrant subroutine branch	159
SBRR	Reentrant subroutine return branch	159

APPENDIX C. GENERAL DESCRIPTION OF THE COMBINED FILE DIRECTORY

A user may have one or two file directory blocks on the disc; the second block is an overflow block. Each block consists of 128 words containing a variable number of file directory entries. Each entry is in the format pictured below.

If the first word of the block is zero, the block is considered to be empty. The last entry is followed by a -1 or -2 word where the -1 indicates additional entries in the overflow block.

FILE DIRECTORY BLOCK

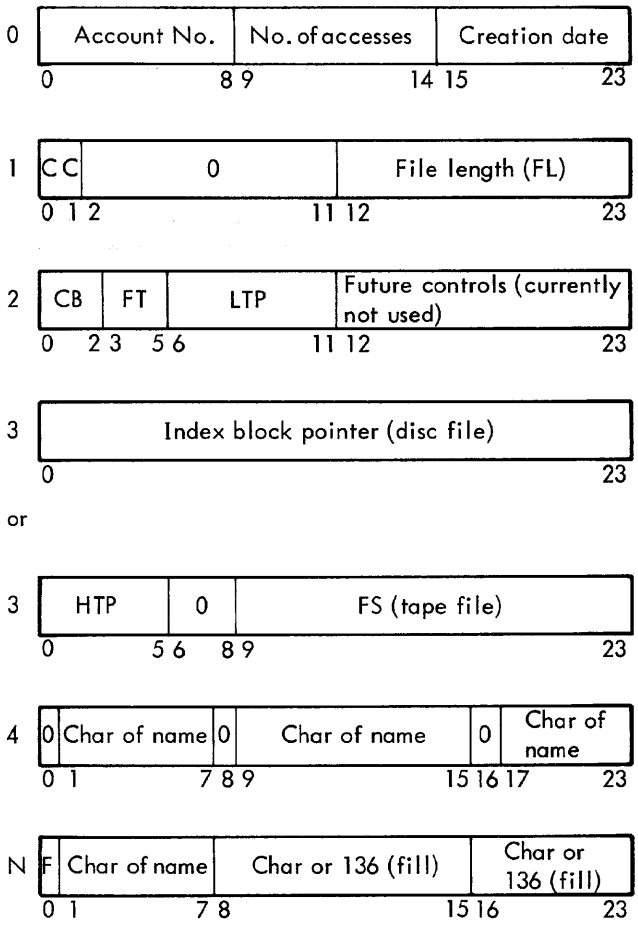


In the case of an overflow block, the last three words are 0, and the overflow block pointer is a backward pointer to the first file directory block (i.e., UNO).

The user number (UNO) is a pointer to the disc address of the user's file directory. The available user numbers vary with the number of discs. Suppose an installation has a disc configuration to allow user numbers to range from 1 through 1377B. An arbitrary decision could be made to assign user numbers 1 through 1077B, thus allowing number 1100B through 1377B for overflow file directories. The overflow file directory pointer is actually a "fictitious user number".

The variable LUNO (Last User Number) designates the end of the overflow file directory area. In the above example LUNO would be set to 1377B and SOV (Start of Overflow area) would equal 1100B. NOVP would be initialized to LUNO and decremented each time an overflow file directory is assigned.

FILE DIRECTORY FORMAT ON DISC OR TAPE



Account No. 41 would be account D1, 32 account C2, etc.

No. of Accesses Number of times the file was accessed since last disc reordering. Reaches a maximum of 77B and remains there until next disc edit.

Creation Date Bits 15-18 are the month number less one. Bits 19-23 are the day of the month less one; for example, 154 is April 13.

CC Indicate a change in file size (the file was written on). These bits are used by the concurrent tape back-up routine and the disc file edit routine.

FL File length for disc files where each bit represents one data block of 255 words.

CB File control bits, 0 = Tape file
2 = Disc file

FT File type (1 through 4)

LTP Low order tape position; for example, if LTP = 5, this is the fifth file on a multi-file tape.

F End of entry flag

HTP High order tape position

FS Tape file size

USER ACCOUNT DIRECTORY ON DISC

Words	0	1	2	3	4	5	6	7
	Acct. password				CA	na	na	na
8	User Name 1				C		N	
13	User Name 2				C		N	
18	User Name 3				C		N	
23	User Name 4				C		N	
28	User Name 5				C		N	
33	User Name 6				C		N	
38	User Name 7				C		N	
43	User Name 8				C		N	
48	User Name 9				C		N	
53	User Name 10				C		N	
58	User Name 11				C		N	
63	p							
	0			11	12			23

where

CA This word could be used to contain status parameters that apply to the entire account. Currently it is not used.

na is not assigned

C is a control parameter

N is a user number

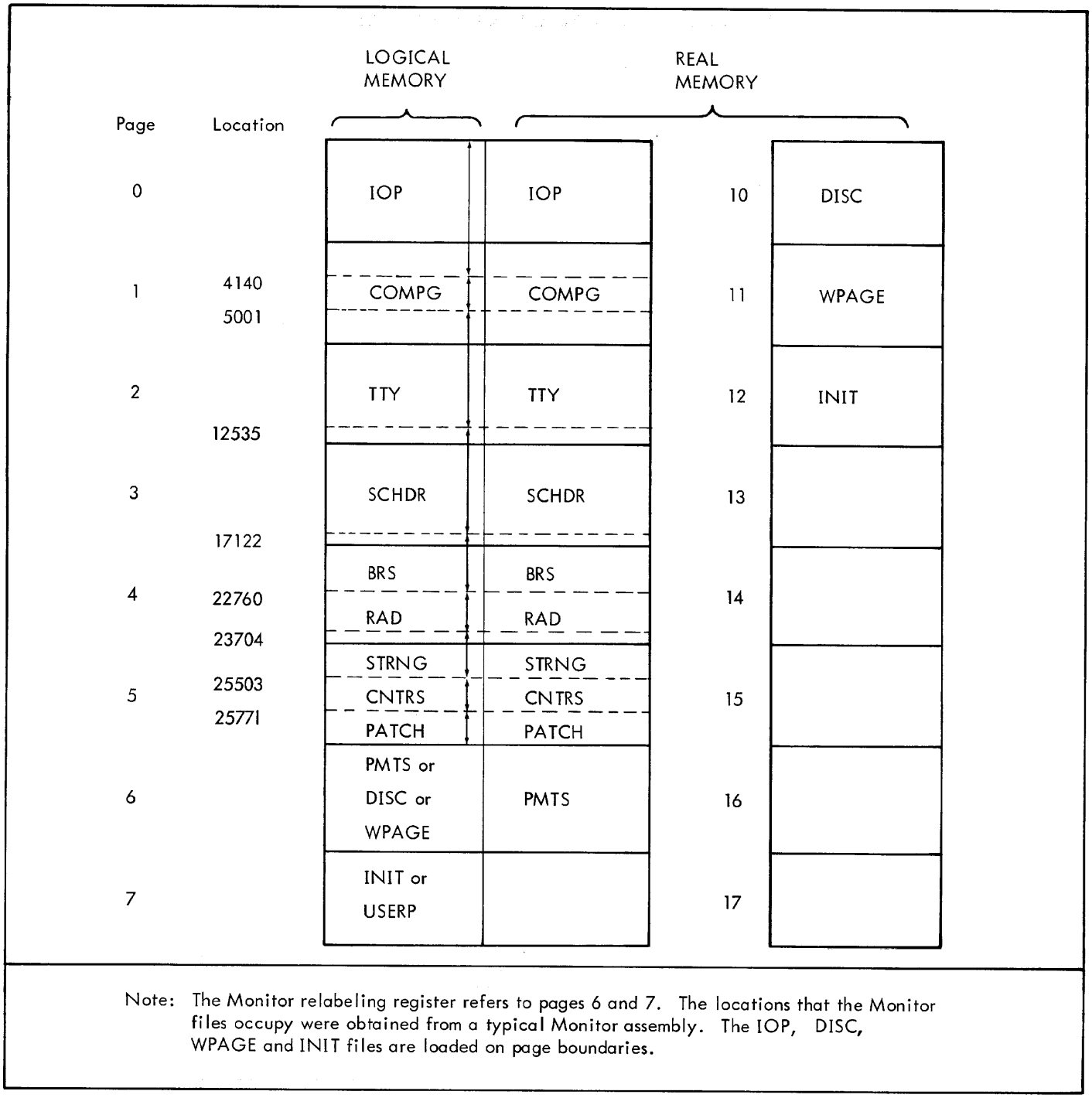
p is reserved for an overflow pointer and not presently used.

The control parameter bits are assigned as follows:

Bit	Use
0	System status
1	Control
2	Operator status
3	Subsystem status
4, 5	Not assigned
6, 11	Subsystem classes

APPENDIX D. MONITOR FILES

BRS	Contains the BRS dispatcher and the routines associated with miscellaneous BRSs. The trap routines, the memory allocation routine and the routines necessary to set up and maintain the forking structure are here.
CNTRS	All of the counters that are used for statistical purposes are here. If the assembly parameter CNTPKG is set to -1, this file is not needed.
COMPG	This file contains some of the constants that are used by both the Executive and Monitor, the job indexed tables, some of the teletype tables, various error counters, etc.
DISC	This file contains the opening and closing routines and the drivers for the disc. However, the disc queue and interrupt routine are in the IOP file.
INIT	This file is used during system initialization. The DSWAP utility routine reads the first 14K of the Monitor from the specified disc and then branches to location SETSET (in SCHDR file) which reads the INIT file from the disc. The coding in INIT initializes pertinent system tables, puts the phantom user onto QTI, and enables the interrupt system. While the time-sharing system is operational, INIT is not core resident. The coding for the automatic restart is also in INIT.
IOP	This file pertains to the I/O devices and file logic. The file control tables and device tables are here. The 131 and 133 interrupt routines and the opening and closing routines are here. IOP contains the general logic for the devices that are on the W-buffer. The drivers for a particular device will be found in either DISC or WPAGE.
MCONST	This file contains the systems configuration dependent parameters, such as number of discs, RAD, description of peripherals, size of buffers, etc. It also contains OPDs and macros. This file is nongenerative (contains only assembly directives) and is used only when the Monitor is assembled. The TS page variables which the Monitor uses are defined in this file.
PMTS	This file contains the scheduled queues, the PMT and SMT tables, and the PAC tables.
RAD	Contains the RAD driver, interrupt routine, and RAD queue.
SCHDR	Contains the routines that are necessary to dismiss one user and activate another. The scheduler, the swapper, the phantom user, the clock interrupt routines, the software interrupt routine, and the crash routine are here.
STRNG	Contains the routines associated with string processing. The routines used for constructing and maintaining the hash tables are here. The floating point POPs are also here.
TTY	This file contains the routines that are associated with the teletype. The five interrupt routines, the TCO and TCI SYSPOPs, the majority of the tables indexed by teletype number, the teletype buffers and various routines that the phantom user performs that are associated with the teletype, such as rubouts and the initialization of the Executive for a user, are here.
USERP	User page (TS page). This file provides a map of the symbols used mainly by the Executive. The TS page serves to make the Executive reentrant. Each user has a TS page with the format shown by this file. This file is loaded into page zero during generation of the Executive.
WPAGE	This file contains the drivers for all of the devices, except the disc, that are on the W-buffer.



Note: The Monitor relabeling register refers to pages 6 and 7. The locations that the Monitor files occupy were obtained from a typical Monitor assembly. The IOP, DISC, WPAGE and INIT files are loaded on page boundaries.

Figure 34. Memory Diagram for the Monitor

APPENDIX E. THE EXECUTIVE FILES

EXCNS This file contains configuration dependent parameters, MACROS, and OPDs referenced by the Executive. This file is non-generative (contains only assembly directives) and is used only when the Executive is assembled.

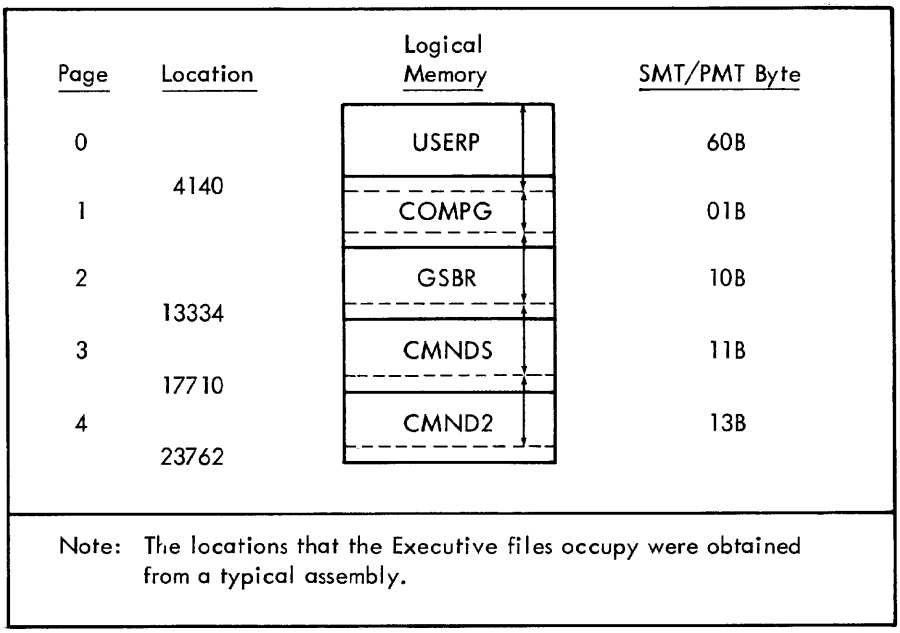
GSBR This file consists of a collection of general subroutines that are used by the Executive.

CMNDS This file contains the command processor for the Executive. An escape to the Executive transfers control to this file. The routines associated with the Executive commands or a transfer to the routines are in this file. The command and subsystem hash tables are here.

CMND2 This file contains the coding for the less frequently used Executive commands (such as logging a user on and off the system).

INTLE This file is both assembled and executed when the Executive is generated. The routine that sets up the commands and subsystem hash tables is here.

FLTIO This file contains the coding for BRSs 48, 52, 53, and 91 and the SIC and ISC SYSPOPS. Although this file is not a part of the Executive, it is assembled with the Executive since it uses the TS page and other Executive constants. FLTIO is referenced by SMT byte 12.



Note: The locations that the Executive files occupy were obtained from a typical assembly.

Figure 35. Memory Diagram for the Executive

APPENDIX F. INITIALIZATION AND ASSIGNMENT OF THE PAC TABLES

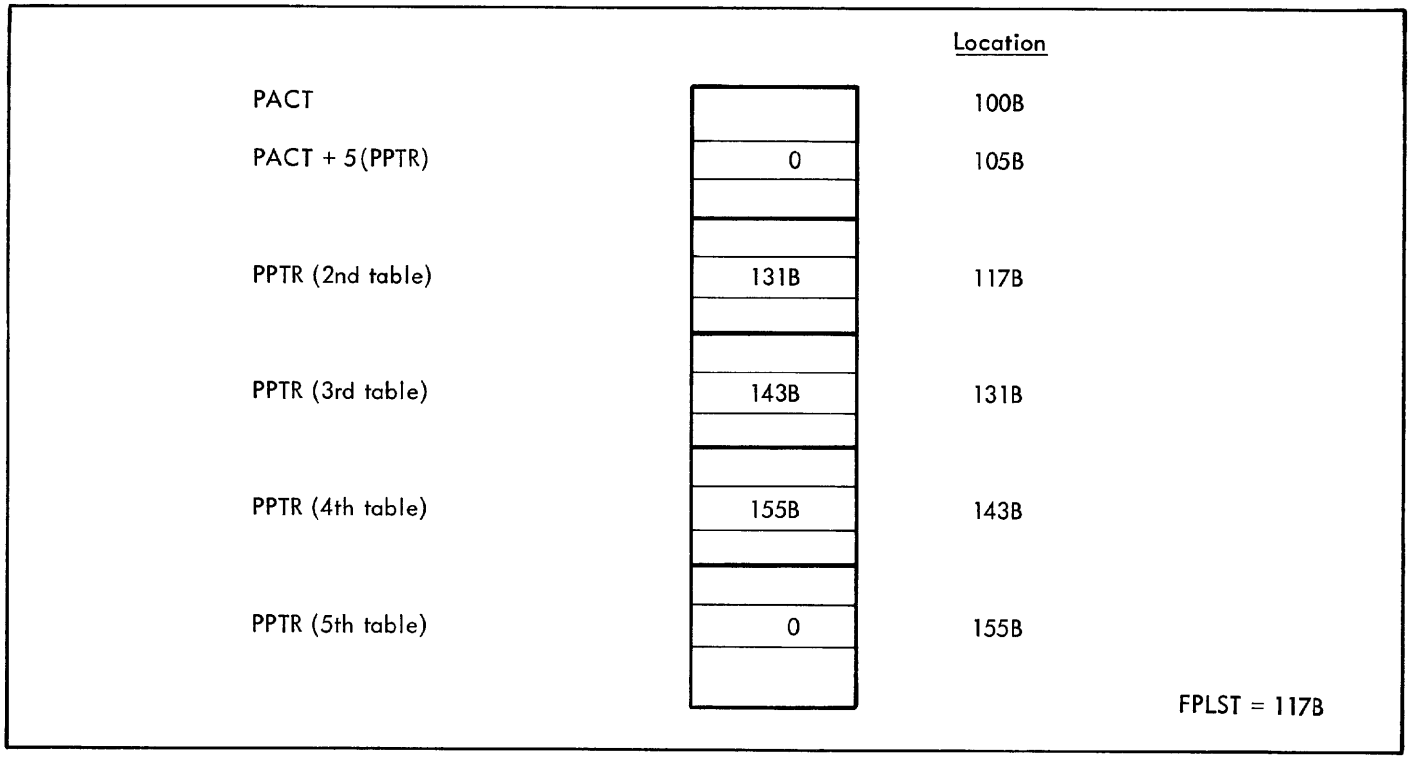
Assume there are only five PAC tables in the system and the tables begin at location 100B. (Actually there are 144 tables that are located in the PMT file.)

```
        NPAC    EQU     5
        NPPAR   EQU     10      (10 entries per PACT)
100B    PACT    BSS     NPAC * NPPAR
        PEND    EQU     *
        PNEXT   EQU     PEND + 0
        PL      EQU     PEND + 1
        PA      EQU     PEND + 2
                .
                .
                .
        PPTR    EQU     PEND + 5
                .
                .
                .
        PIM     EQU     PEND + 9
```

When the program is assembled, the following values will be assigned:

Symbol	Value
PACT	100B
PEND	162B
PNEXT	162B
PL	163B
PA	164B
PPTR	167B

When the system is initialized (in INIT), the following values are inserted into the PAC tables. See Figure 36.



Note: Only the PPTR word of each table is shown.

Figure 36. Initialization of PAC Tables

Note that the PPTR word of each table points to the location of the PPTR word of the next available PAC table. The PPTR word of the last PAC table contains a zero indicating that there are no more available tables.

In order to retrieve entries from the PAC table, there is a PACT pointer associated with each PAC table. The pointer for the currently active fork in the system is always stored into location PACPTR. The PACT pointer is a negative number that allows the PAC table entries to be addressed using the end of the tables (see definition of PNEXT, PA, etc.) as the point of reference.

When the system is initialized the Phantom User is assigned the first PAC table. Location FPLST always contains the address of the PPTR word of the next available table. If the address of the PPTR word of an available table is known, the PACT pointer can easily be calculated.

Considering the above example, the following values would be assigned at system initialization:

FPLST = 117B

PUPACP = -62B = PACT pointer of phantom user

When the phantom user is active, the following coding would retrieve his PAC table entries:

```
        LDX     PUPACP
INST    LDA     PA,2        Fetch PA word
          :
        LDA     PIM,2       Fetch PIM word
```

The effective address of the instruction at INST would be:

PA = 164B

(X) = -62B PACT pointer of phantom user

Effective Address = 164 -62 = 102B = Address of PA entry for phantom user

When a user comes on the system or any fork is declared, a PAC table must be assigned. Coding very similar to the following is executed in subroutine GFK:

```
          :
LDA     FPLST       Get point to a free table.
SKG     = 0         Are there any free tables.
BRU     NOROOM
SUB     =PPTR       Calculate  PACT pointer.
COPY    AX,A        Copy A to X; clear A.
XMA     PPTR,2      Get pointer to next free table.
STA     FPLST       FPLST updated.
STX     TEMP        Save PACT pointer
          :
```

Once the PACT pointer is assigned, it is stored into the PNEXT word of the fork previous to it on the scheduled queues or it is available in the PDOWN or PFORK entries is the fork was dismissed to activate a lower fork.

When a fork is terminated, the PAC table associated with the fork is returned to the free PACT list. Coding similar to the following is performed:

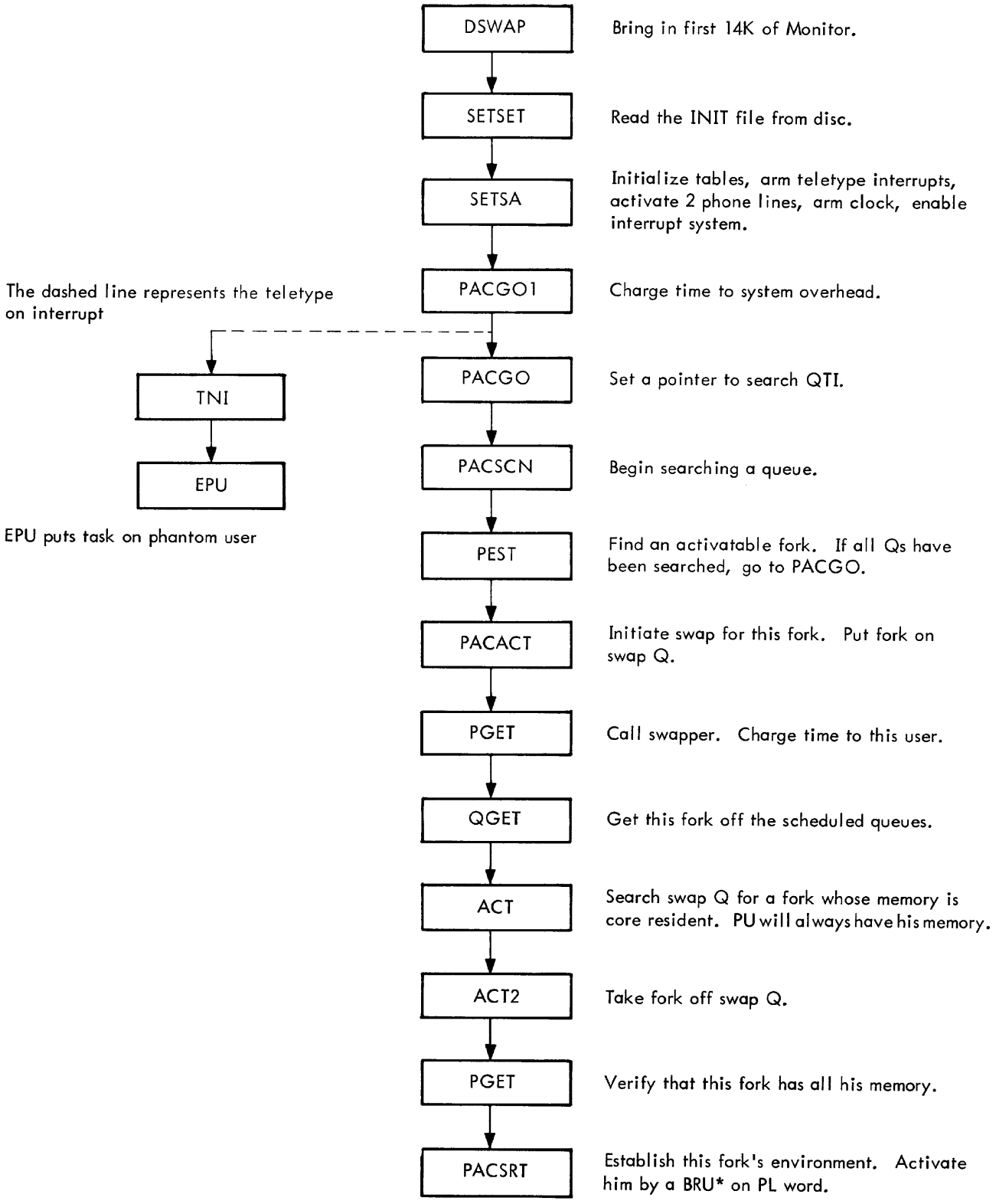
```
LDX     PACPTR      Get pointer for this fork.
COPY    XA
ADD     =PPTR       Get address of PPTR word of
                    this table.
XMA     FPLST
STA     PPTR,2
```

Note that FPLST is pointing to the table that has just been released. PPTR of the table that has just been released is pointing to another free PAC table (i.e., as a fork terminates the PAC table is added to the top of the list of free tables).

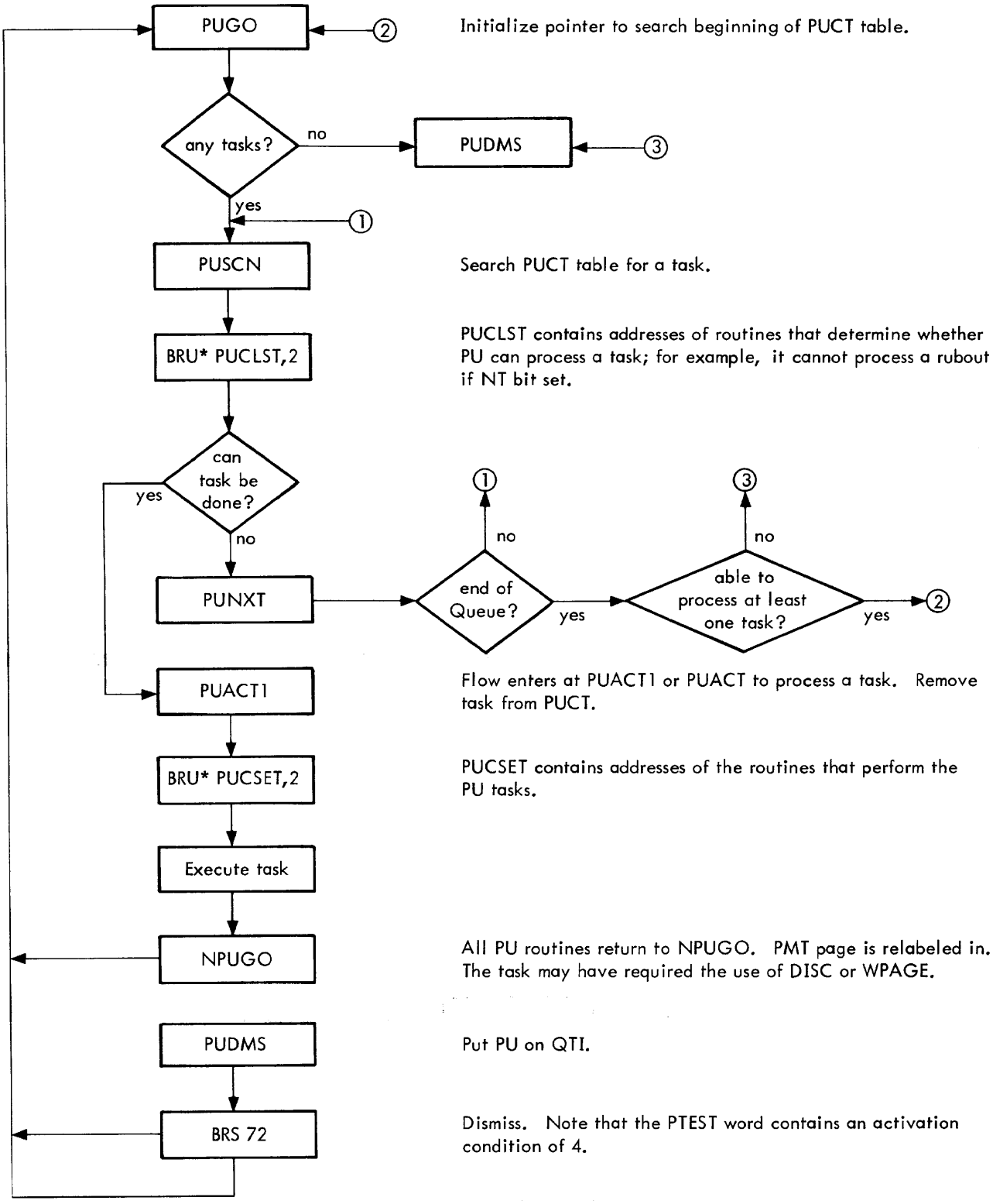
The method that is used for the assignment of PAC tables is similar to the method used for the assigning of the job and file numbers.

The TTNO table contains the chained list of free job numbers. Location FULST contains the next available job number. The FA table and location FFLST are used in conjunction with the file numbers.

APPENDIX G. INITIALIZATION OF SYSTEM AND ACTIVATION OF FIRST USER



APPENDIX H. THE PHANTOM USER LOGIC



Initialize pointer to search beginning of PUCT table.

Search PUCT table for a task.

PUCLST contains addresses of routines that determine whether PU can process a task; for example, it cannot process a rubout if NT bit set.

Flow enters at PUACT1 or PUACT to process a task. Remove task from PUCT.

PUCSET contains addresses of the routines that perform the PU tasks.

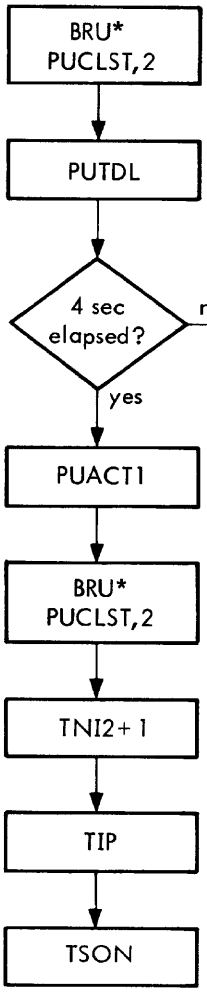
All PU routines return to NPUGO. PMT page is relabeled in. The task may have required the use of DISC or WPAGE.

Put PU on QTI.

Dismiss. Note that the PTEST word contains an activation condition of 4.

Note: When the phantom user is activated it continues to execute until either no tasks are on PUCT, or it is unable at this time to process any of the tasks that remain on PUCT.

APPENDIX I. PHANTOM USER LOGIC TO PROCESS A TELETYPE ON INTERRUPT



The activation code for the tt on interrupt = 6.

Make sure 4 sec have elapsed since interrupt occurred. 4 sec delay allows for TTY carrier to become stable.

Attempt to process next PUCT entry if 4 sec have not elapsed.

Remove task from PUCT table. Decrement PUCTR.

Location PUTST contains TNI2+1.

Make sure data set is ready and carrier is stable.

TIP is a general despatcher for tt on, off, rubout, and BRS 112.

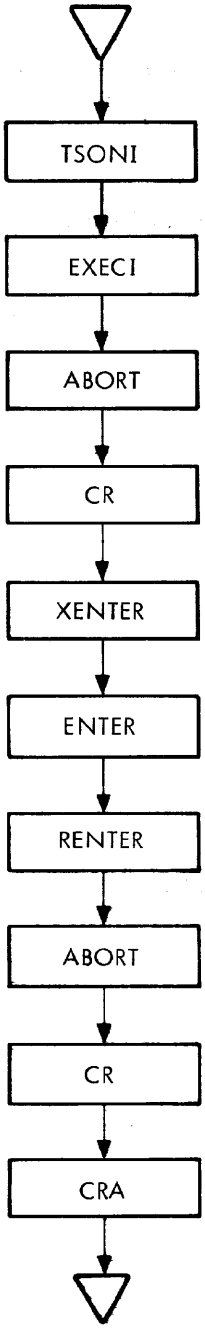
Get job number, get PAC table for Executive fork, set up PAC table parameters, zero PMT table, set PL word to TSONI, put Executive PAC table on QTI.

Format for PUCT table entry for teletype on interrupt:

Word 0	Pointer to next task on PUCT	
1	6 @ TNI2+1	The 6 is the activation code used by PU logic. TNI2+1 is address of routine to process task.
2	1 @ CN	The 1 is used by the dispatcher at TIP.
3	0 @ CN	Teletype channel number.

APPENDIX J. FLOW REQUIRED TO INITIALIZE THE EXECUTIVE WHEN A USER LOGS ON THE SYSTEM

When the phantom user processes a "teletype on" interrupt, the PAC table for the Executive fork is initialized and put on QTI. When the Executive fork is activated it begins execution at TSONI.



Get TS page for user, execute a BRU EXECI,4 to enter user mode.

Sets up various parameters, allows 1-1/2 minutes for log on time. EXECI is the general entry point to the Executive from the Monitor.

This is the general entry point after rubout.

CR is general entry point to process Executive commands. The third page is relabeled out since it is used mainly during logging on and off. Since user has not logged on a branch is taken to XENTER where the third page is relabeled in and a branch to ENTER occurs.

Types "READY" and "PLEASE LOG IN". Reads AUD and file directory. Verifies log in parameters.

This is the general entry point into CMNDS from CMND2. Third page relabeled out.

Types the dash and begins to read the command.

Interprets the command and takes appropriate action.

APPENDIX K. SUBROUTINE TRACE OF THE SWAPPER

CHRL					Load pseudo-relabeling into A, B, X.
	SWAP				
		DCRL			Decode pseudo-relabeling into SRT table.
			UPRL		Pseudo Byte → SRT Entry 0 → 40B Page in core → Real page number including read-only bit (bits 18-23 of PMT) PMT entry on RAD → 0 @ PMT adr, 4 SMT entry on RAD → 0 @ PMT adr, 5
	SWP1 through SWP41				Lock page, reduce MAC for core resident pages. For pages that must be read, build the following tables: SWT5 – RAD address (in bits 10-17) of page SWT14 – SRT entry of page SPT and SAT – These tables cross-reference each other. They allow the commands to be later inserted onto the RAD queue in a manner that is optimum for RAD access.
	SWP2				Check MAC to determine if there are enough unlocked pages to attempt to bring in this fork.
	MA8 through MA12				Determine which pages will be swapped. The method for selection of pages is described in Chapter 5. The write commands are added to the SPT and SAT tables. SWT6 contains the first location in SWT5 that was not used by the read commands. SWT5 – Contains PMT/SMT entry of page to be swapped SWT14 – Contains RMT entry of page to be swapped
	SWP10				If a read-only (RO) page is released, the in-use bit (bit 0 of RMT) is reset so that later the system knows that it can put a request on the RAD queue to read into this page without waiting for the previous contents to be written out. Bit 0 (on RAD) of the PMT entry that currently corresponds to this page is set.
	SWP12 through SWP36				The current position of the RAD is obtained in order to pull commands from SPT and SAT in an optimum manner.
	SWP15 through SWP23				The RAD requests are built from the information in the SPT, SAT, SWT14, and SWT5 tables. A RAD read calls OMR while a RAD write calls OMW. The system distinguishes reads from writes by the SWT6 pointer. If a read and write command refer to the same page, the write command is entered onto the RAD queue before the read command. This is accomplished by examining bit 0 (in use) of RMT. This bit is reset after a write command is added to read queue.
		OMW	RTW	RTC	Add write command to RAD queue. Lock page while RAD I/O is active.
	or				
		OMR			Adjust bits 0, 1, and 2 of RMT appropriately. If call for activation, insert swap queue (SWQ) pointer into RMT and adjust page count word of SWQ. The real page number is inserted into PMT and bit 0 of PMT is reset. If the

					call was from MGET (DROBIT of PMT set), RTC is not called since no read is required.
			RTC		Add read command to RAD queue. Lock page while RAD I/O is active.
		RST			RST is called when all the commands are on RAD Q. Start the RAD. Do not wait for I/O to complete if call was for activation. Otherwise wait and exit skipping if error occurred.
			RTS		
				RDR	
	SWP3				PTRL calls PKRL to form the 3 hardware relabeling registers in A, B, and X. The read-only bit is set for all the bytes.
		PTRL			
			PKRL		
CHRL+7					SWAP exits skipping if successful.
	LABEL				The real relabeling is stored into SRRL1, SRRL2, and SRRL3. An exit is taken if this was a call for activation. Otherwise the TS page is marked as being not read-only. The Monitor accesses the TS page and Monitor mapping does not include a read-only bit. The hardware relabeling is then set.

APPENDIX L. THE DISC LOGIC

The software initiates disc I/O by first placing an entry onto the disc queue (DRQ). Each entry on the disc queue requires three locations. The format of a disc queue entry is shown in Figure 37.

The disc drivers will add an entry to DRQ at the location indicated by EDCL. Location NDCL contains the count (minus one) of the number of commands that are on the disc queue and are ready for the disc interrupt routine to process. The disc driver will increment NDCL when it adds an entry and the disc interrupt routine (IDM) will decrement NDCL when it processes an entry. IDM will sequentially pull commands off of DRQ and execute them until NDCL has the value -1. At this point there are no more commands on DRQ that are ready to be executed.

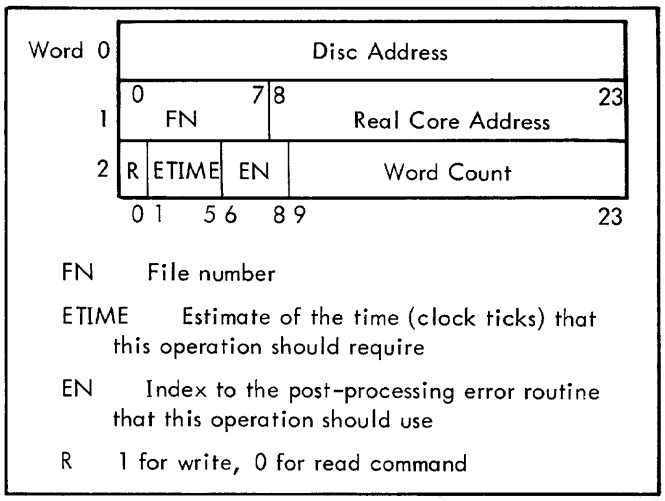


Figure 37. A Disc Queue Entry

Since both the disc drivers and the disc interrupt routine alter the contents of NDCL, the drivers must disable the interrupts while NDCL is being adjusted. To make this process more efficient, the variable DTXS1 is implemented. Suppose the driver wished to perform an operation that required the adding of two commands to DRQ. The driver would add both commands to DRQ and increment DTXS1 each time a command was added. Then the driver would disable the interrupts, adjust NDCL by the value indicated by DTXS1, and restore DTXS1 to a -1. Thus DTXS1 contains the number of commands (minus 1) that the driver has added to DRQ for any given operation.

The disc interrupt routine is entered in two ways: (IDMRET indicates how IDM was entered.)

1. as a response to an interrupt (IDMRET = -1)

2. as a subroutine called by a disc driver (IDMRET=0)

If the disc is inactive, the driver must not only add an entry to DRQ, but also initiate the disc I/O by calling IDM.

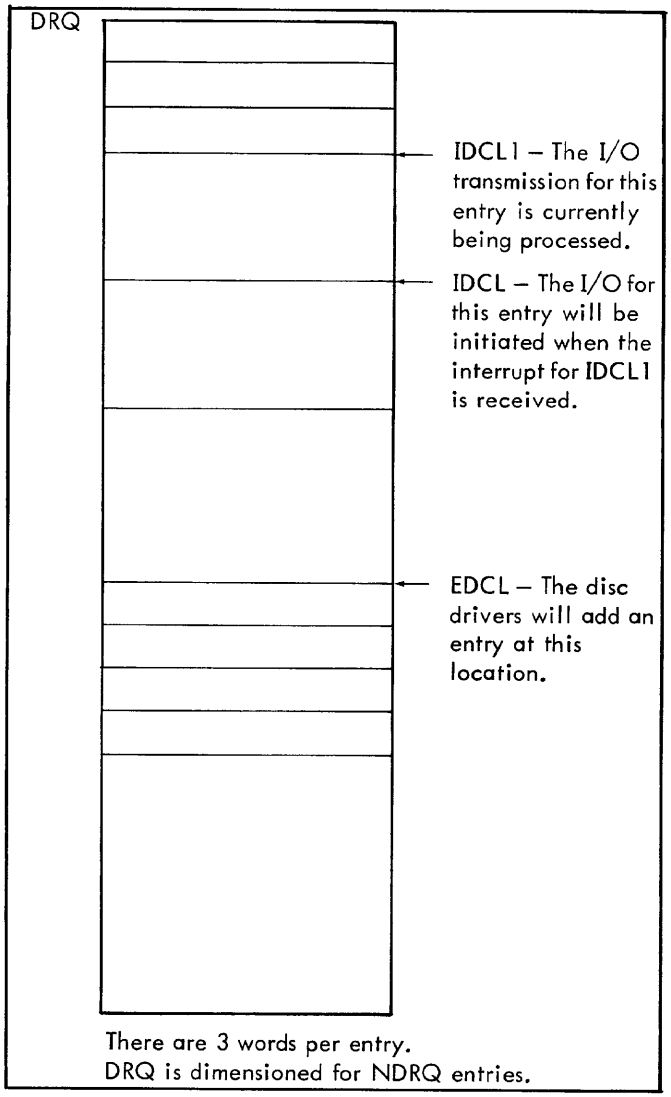


Figure 38. The Disc Queue (DRQ)

When IDM is entered as a response to an interrupt, the following occurs:

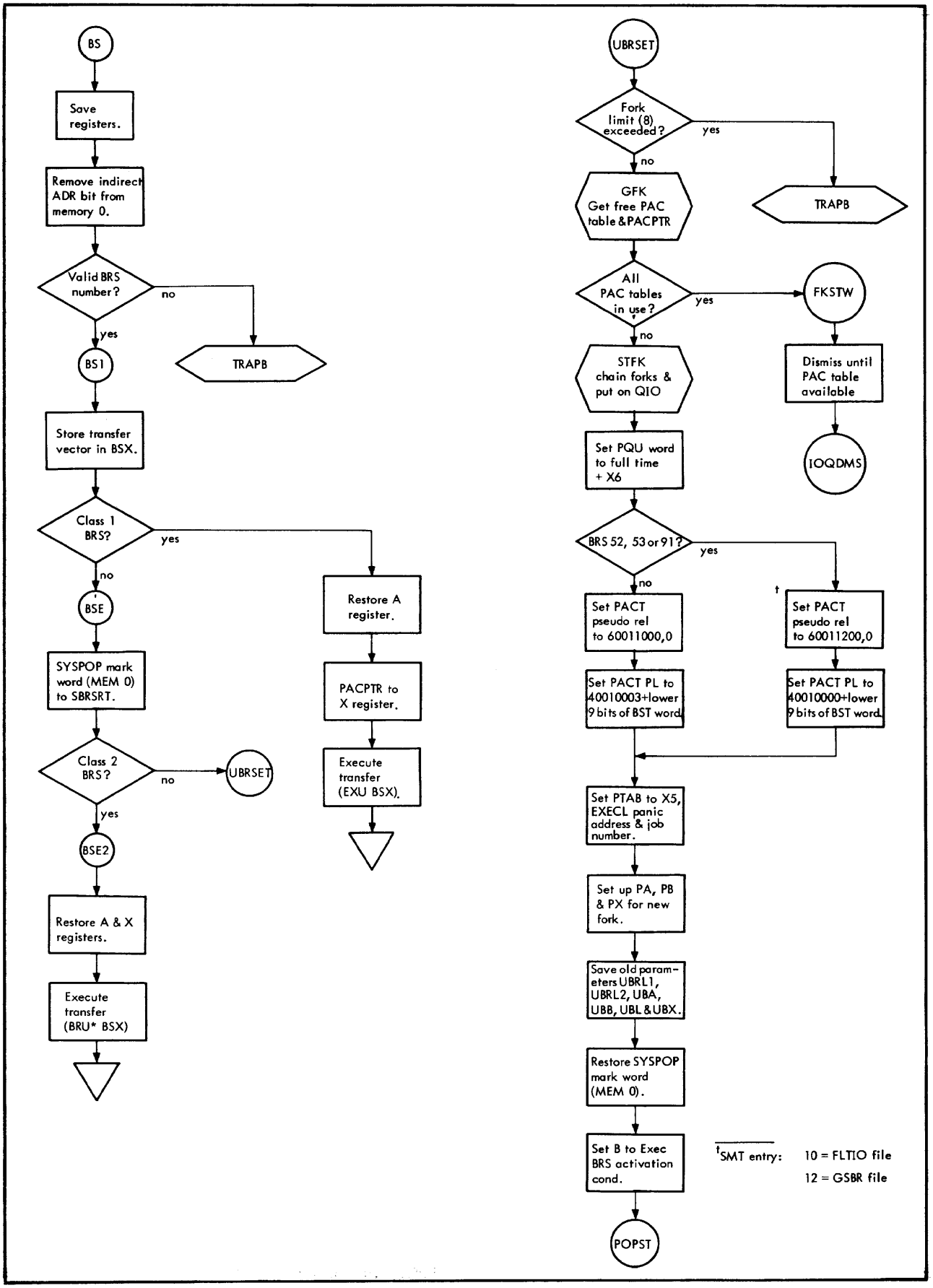
1. The I/O for the next entry (if any) on DRQ will be initiated. This entry is pointed to by IDCL.

2. The post-processing, i.e., error check, release locked pages, etc., will be done on the entry for which the interrupt was received. This entry is pointed to by IDCL1.

When IDM is entered as a subroutine call, IDCL1 will equal -1. The I/O for the first entry will be initiated and IDCL1 will point to this entry. The next entry (if any) will be set up (the POT words will be contructed), and pointed to by IDCL.

When IDM has finished executing all the requests on DRQ, IDCL1 will be set to -1.

APPENDIX M. BRS LOGIC FLOW



APPENDIX N. TRACE OF THE SUBROUTINES WHICH ARE CALLED BY THE BRS 1 (MONOPN) IN ORDER TO OPEN THE DISC

MONOPN					
	BGET				Get a TS page buffer.
	DRMOPN				Dispatch on OPNDEV table.
					The following path is taken (in DRMOPN) to open an old file.
		BSET			Get real address of buffer.
		DTC			Put command to read index block on disc queue.
		DTF			Set file number and index to error routine in disc queue entry.
			DTS		Increment count of number of commands on disc queue.
				IDM	This routine called only to start the disc if the disc is currently inactive.
				DTP	Reset DTXS1 variable.
					The following path is taken (in DRMOPN) to open a new file.
			DTA		Allocate a disc block for the Index Block.
			DTZ		Zero the area of TS buffer that the Index Block will occupy.

APPENDIX O. SUBROUTINE TRACE FOR BIO FLOW WHEN THE DEVICE IS THE DISC ON INPUT

BIO							Block input/output.
	IOI						Return buffer address and FD word.
BIS							Transfer all available words.
BIG							BIG is entered when the data block in the buffer is empty.
	GPW						Find out it is the disc.
		GPWD					Set up arguments for ED.
			ED				Get buffer pointer and drive device.
				BSET			Get buffer address.
				MPDSC			Map in the disc page.
				DRMSI			Compute disc address.
					DTC		Put command on DRQ.
					DRF		Set file number, interrupt index in DRQ.
					DTS		Update DRQ command counts.
						IDM	Start disc (if currently inactive).
						DTP	Reset command count.
	NIODMS						Calls:
				MPPACT			Map in PMT page.
	IODMS						Make up PTEST word: 11 @ FD+file nmbr.
	POPDMS						Dismiss fork.

APPENDIX P. SUBROUTINE TRACE FOR THE LOGOUT COMMAND

LOGOUT		
	KILL 6	Use BRS 121 to release memory of PMT entry 61 and following.
	BRS 8	Close all files.
	MKFD	Output updated file directory to disc.
	WRACT	Write system accounting.
	TIMER	Print elapsed time message.
	BRS 14	Wait until elapsed time message is printed.
	BRS 4	Release TS page.
	BRS 112	The BRS 112 releases the job number, resets TTYASG and WERIS, releases PAC table for the Executive, removes any tasks that this teletype has on the PU queue, and finally puts a task on PU (the same as the teletype ON interrupt) to set up an Executive for this user. The system will set up an Executive fork and allow the user to relog on provided that he has not hung up. Also, a check is made to see if the operator has issued the SHUT DOWN command. If so, the teletype will be deactivated.

APPENDIX Q. SUBROUTINE FLOW FOR THE PHANTOM USER TASK WHICH PROCESSES A TELETYPE OFF INTERRUPT

TIP			
	TFIP		
		HFK	Get PACPTR of Executive fork.
		RFK	Set panic tables for entire forking structure.
		TFK	Release all PAC table in forking structure except Executive fork. The Executive fork is put on QIO.
		CLIB	Clear input TTY buffer.
		CLOB	Clear output TTY buffer.
			TFIP gives the Executive fork an immediate activation condition and sets the PL word to location OFFINT which is in the Executive.
OFFINT			Decide if user is still logged on system.
			The following path is taken in OFFINT if the user is not logged on the system.
LGOUT3			
	BRS 14		
	BRS 4		Release TS page
	BRS 112		See subroutine trace of LOGOUT command (Appendix O).
			The following path is taken in OFFINT if the user is logged on the system.
OFDUM			Clear SWOFF.
XDUMP			Initialize for using /$/ Dump file.
OFINT2			Check to see if user has a /$/ file. If so, the dump is taken.
OFINT1			
	KILL 6		
	BRS 8		
	MKFD		
	WRACT		
	Go to LGOUT3		

APPENDIX R. AUTOMATIC RESTART

While the time sharing system is operating, some conditions can occur which cause the system to "crash". When the system crashes the MONCR (Monitor Crash) routine is entered. The Monitor will execute a BRM MONCR if system tables or pointers are altered, spurious interrupts occur, a trap occurs due to watchdog timer runout, etc. Also, the operator can force a system crash by toggling breakpoint 4.

If the system is executing in an infinite loop while in Monitor mode, no dismissals will occur, and thus no users (including the phantom user) are activated. In this event, the operator is forced to toggle breakpoint 4.

If breakpoint 3 is set when MONCR is entered, the system will execute an unconditional branch to itself (BRU *). The operator can clear this condition and allow the system to restart automatically, or, run the DSWAP utility program. Note that the DSWAP routine can store the crash (the current contents of the first 32K of core) on the system arm positions of discs zero and one. If the system is reinitialized by using DSWAP, all of the users currently logged onto the system are disconnected and their current program environment is lost.

If breakpoint 3 is reset when MONCR is entered, the system attempts an automatic restart that will save the current program environment of all the users. In the event of a crash, the system returns to each user at the Executive level and outputs the message: SYSTEM RESTARTED. The user can continue execution in the subsystem in use prior to the crash by using the CONTINUE command. He should recover the program environment that existed the last time he was in the command mode of the subsystem. In general, the restart capability allows program recovery to the same extent as that available when using a /$/ file to recover from a teletype disconnect.

The automatic restart code first disarms, disables, and clears all interrupts. If any write commands remain on the RAD queue, these are performed (using non-interrupt I/O) since the swapper has already marked these pages as being RAD resident in the SMT/PMT entries. If any read commands remain on the queue, the operation is not performed, but the RMT and SMT/PMT entries are properly adjusted.

Real page 12B is written (if necessary) to the appropriate RAD location since the initialization page (see Appendix E for a description of INIT, the initialization page) must be read into this page. Page 12B is relabeled into virtual page 7 of the Monitor for relabeling and a branch to location RESTRT, which is in INIT, occurs.

At RESTRT, any pages which are marked "not read only" in the RMT table, are written to the appropriate location on the RAD. The pages are marked as being RAD resident in the corresponding PMT entries. At this time, a copy of all user pages should be on the RAD.

The following system information is moved to high core: COMPG file, TTNO, PUCT, PMT file (includes PMT, SMT, PACT tables, and scheduler queues), and miscellaneous associated pointers.

If breakpoint 2 is set, the first 32K of core (the crash) is saved on discs zero and one. A new copy of the first 14K of the Monitor is then read from the appropriate disc. The initialization then proceeds in the normal manner except that new copies of DISC and WPAGE are not read into core. This saves the disc bit map and alleviates the requirement of running the MAP program.

The tables previously saved in high core are now moved to their appropriate locations. The contents of real page zero are moved to high core since each user's TS page must be read into these locations.

The scheduled queues are now checked and if any irregularity is detected an appropriate message is output to the console teletype. A new copy of the scheduled queues is established and the phantom user is placed onto QTI.

Next, the phantom user (PUCT) queue is scanned. If it is intact, all tasks except teletype on/off tasks are removed. If it is not intact, a message is output to the console teletype, and a new PUCT queue which contains no tasks is constructed.

Next, each PAC table is examined. A PAC table is associated with an Executive fork, a subsidiary fork, or is not in use. The first PAC table in the array should belong to the phantom user (job number zero).

When an Executive PACT is encountered, the TS page is read into core. If the user has a disc file open for output (and either the data block or index block has been changed), the checksum word is destroyed and the index block written out to the disc. This action is necessary to preserve the integrity of the disc bit map. The checksum is destroyed to insure that the user is cognizant of the fact that his file was not completely written when the crash occured. A scan is now made through the hierarchy of forks. All the subsidiary PAC tables are given an activation condition of "dead" and returned to the free PACT list. If a fork has its panic table in the TS page, the panic table entries are initialized from values in the PAC table and the PB and PX arrays in the TS page. This allows the user to successfully execute the CONTINUE or BRANCH command once the system is restarted. Next, the TS page buffers are all marked as being available and the forking structure is shown to consist of only the Executive fork. The PAC table for the Executive is initialized and placed onto QTI.

Note that once the Executive PAC table is found, the associated forking structure is entirely processed. Since the job number is available in each PAC table, an array named JOB

(indexed by job number) is maintained during restart. When a forking structure has been examined, the appropriate JOB table entry is marked to indicate that the processing of this job has been completed.

When the PAC table for a subsidiary fork is found during the PACT array scan, the job number is extracted and the table JOB is checked to see if this particular job has been processed. If so, an error has occurred since this PACT should have been marked as "dead" when the job was processed. An appropriate error message is then output. If the job has not been processed, the PACT pointer for this PAC table is stored into an array named TBL1. If the forking structure for this job is intact, the PACT pointer will be removed from TBL1 when the Executive fork is found. Thus, if any PACT pointer remains in TBL1 after the scan of all the PAC tables is completed, a PAC table has been lost and an error message is typed.

When the entire PAC table array has been examined, the system is ready to be restarted. Page zero is restored, the RAD bit map is constructed, the error counters are zeroed, the interrupt system is enabled, the real time clock is enabled, and a branch to the scheduler is taken.

The system will not attempt to restart automatically if the MONCR routine is entered while RAD queue pointers are being adjusted. These pointers are adjusted by the RAD driver and RAD interrupt routine. In this case, memory is in a state of flux and it is impossible to reconstruct the RMT and PMT/SMT tables so that they properly reflect the previous state of memory. In this case, the system will loop in an unconditional branch to itself, and the operator must use the DSWAP routine to restart.

The restart will also be unsuccessful if successive RAD errors are encountered during the attempted reinitialization. In this case, the crash was probably due to hardware malfunction.

Following is a list of the error messages which may be output to the logging teletype during restart.

A. ILLEGAL RMT XXXXXXXX. The octal contents of the RMT entry are printed. The RMT entry contained an SMT table address which was marked not read only.

B. CANT WRITE RAD, ADDRESS XXXXXXXX. The address output is the actual RAD address. Unrecoverable RAD errors occurred while attempting to restart.

C. ILLEGAL PMT ENTRY XXXXXXXX. The octal contents of the PMT entry are printed. The entry contains an illegal RAD address.

D. CHECK SCHED QUEUES. The same PACT pointer is on the queues twice, or a fork is both running and on the queues, etc.

E. CHECK PUCT LIST. The phantom user task list is improperly chained, contains duplicate entries, etc.

F. PU IS LOST. The system has deleted or in some other way destroyed the phantom user fork.

G. LOST PACTS:

XXXXXXXX

XXXXXXXX

The PACT pointers are output in octal two's complement. These are PAC tables which the system considers to be in use (i.e., the PTEST word does not have a dead activation condition) but they cannot be attached to any user's forking structure.

H. (REAL) = XXXXXXXX. Contents of cell REAL at the time of the crash. Verify that this word is not negative.

I. XXXXXXXX HAS JOB NMBR ZERO. The PACT pointer printed is associated with job number zero. This condition is illegal only if the parent fork is not the phantom user.

J. XXXXXXXX HAS NO PARENTS. Same condition as described in case G.

K. 2 TOP EXECS FOR JOB XX. Two Executive forks are associated with the same job number.

L. TS PAGE MISSING FOR JOB XX. The Exec fork for this job has subsidiary forks but the TS page is lost. This is due to an illegal RAD address in byte 60B of the PMT or to the hardware dropping the "TS page assigned bit" from the PAC table.

M. CANT READ RAD, ADDRESS: XXXXXXXX. Self explanatory.

N. RAD MAP CONFLICT AT XXXXXXXX. Two PMT entries point to the same location on the RAD.

O. CANT WRITE DISC, ADDRESS: XXXXXXXX. Self explanatory. The restart program writes the disc while attempting to destroy the checksum in user files which are currently open for output. In this case, the disc bit map may be invalid.

P. RESTARTED JOB XX. Automatic restart of this user was impossible because his TS page was lost or it was impossible to output his index block to the disc. This user will receive the "PLEASE LOGON" message since the system was unable to preserve his program environment.

Q. TN XXX

NU XXX

etc.

All the non-zero PSP counters are output. The counters are then zeroed after this message is output.