Price: $3.50

30
35

SDS 940 TIME-SHARING SYSTEM
TECHNICAL MANUAL

SDS 90 11 16A November 1967

SDS

SCIENTIFIC DATA SYSTEMS/1649 Seventeenth Street/Santa Monica, California

Printed in U.S.A.

PREFACE

This preliminary manual describes the Berkeley Time-Sharing System as modified for the SDS 940 Computer. 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 20-28 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, Phantom User Queue Entry, Job Table, Pseudo Memory Table, etc.

This publication is a reference guide for experienced programmers rather than a primer for beginners. 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 listed below.

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 FORTRAN IV Reference Manual	90 11 15

CONTENTS

APPENDIXES

ILLUSTRATIONS

TABLES

1. INTRODUCTION

The SDS 940 Time-Sharing System consists of three main parts: Monitor, Executive, and subsystems.

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 to core memory
- control of active programs

The Executive 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

Since the user cannot address the computer directly, the resident SYSPOP and BRS utility routines have been provided to allow him to perform I/O functions and control other system operations. When accessed, these routines cause a transfer to monitor mode of operation and a branch to a processing routine. The BRS instruction requires an integer in its address field that specifies the function to be performed. SYSPOP instructions require no such integer. Each SYSPOP name is unique and specifies, by itself, the function desired. Not all of these routines are accessible to every user; special user status is required to access some of the routines.

Subsystems are major processors such as FORTRAN II, CAL, QED, etc. These subsystems are programs that are permanently connected to the main system. Each subsystem is called by name through the Executive. Tables in the Executive indicate how the subsystem is to be started and where its entries are in the shared memory table.

The processors implemented as 940 subsystems have such a high rate of 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. THE SCHEDULER

FORKS

The 940 Time-Sharing System is primarily concerned with program entities called forks. A fork is a self-contained body of code for performing some process. At any point during this process it is possible to activate another fork. Forks are similar to programs and subprograms in other systems, but they differ from their non-time-sharing-system counterparts in a number of important respects.

A fork can be all or part of a program. It can have one or more forks or subprograms running concurrently under its full control.

A fork can share all, part, or none of its allocated memory with the controlling program.

Forks are similar to subroutines, except that, theoretically, all forks making up a program could be executing simultaneously in the time-sharing mode. However, forks can, and frequently do, exist in a hierarchical relationship with one another.

There is at least one fork associated with each active user in the system, but a user can have many forks under his control, each operating independently. There is also a temporary storage area called a TS block that is shared with each active fork associated with a user.

THE PROGRAM ACTIVE TABLE

A fork is defined by its entry in the program active table (PAC table or PACT). This table contains all the information required to specify the state of the computer that the user is programming, except for that contained in the user's memory or in the system's permanent tables. Each PACT entry has the format shown in Figure 1.

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 pseudo-relabeling registers for the user. A third one, that specifies the monitor map, is kept in the job tables. Pseudo-relabeling is discussed in detail in Chapter 5. 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 three pointers called PFORK, PDOWN and PPAR are discussed in Chapter 3.

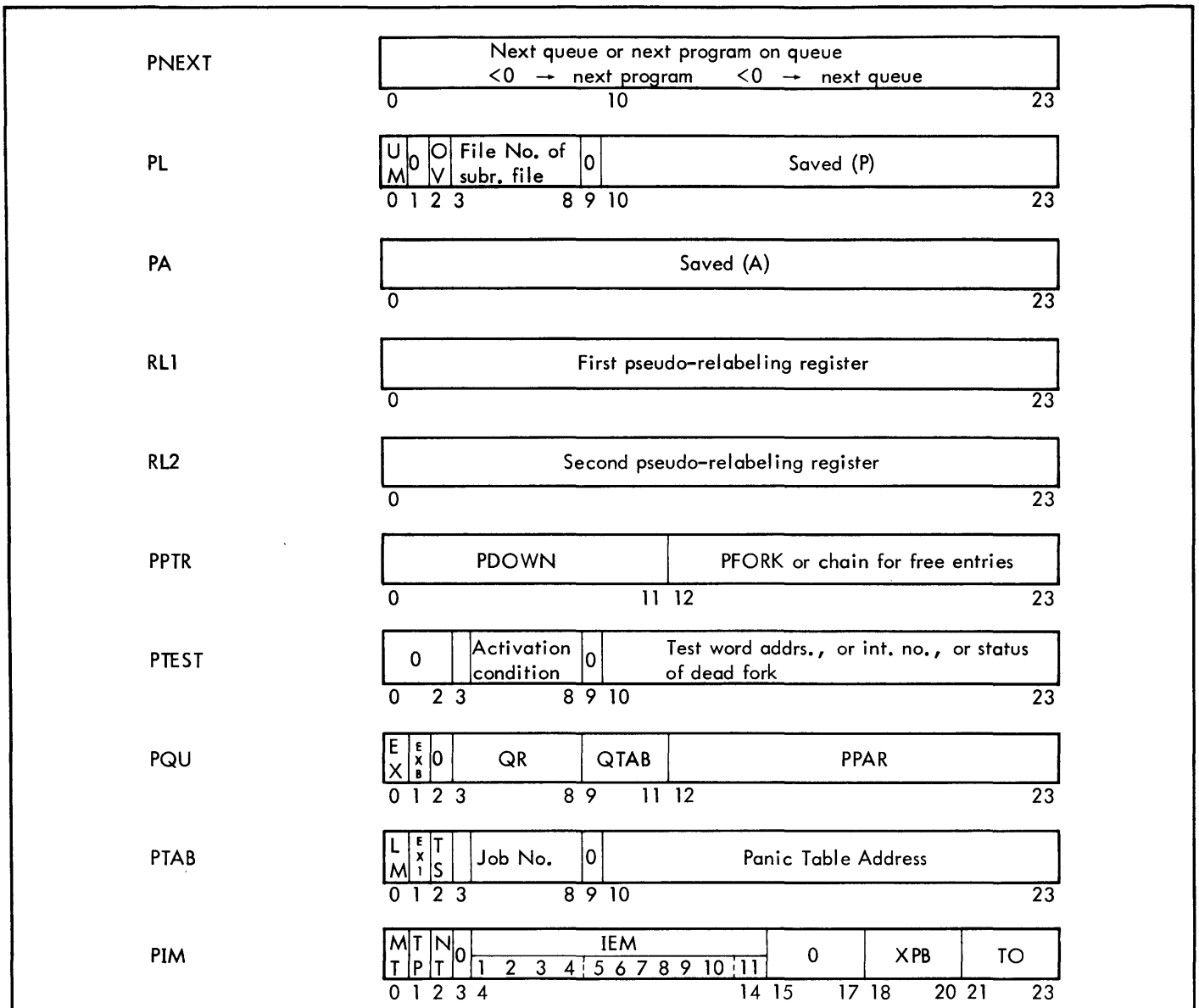


Figure 1. PAC Table – One Per Fork

The word PTAB contains the number of the job to which the fork belongs in bits 3 through 8. The top of PQU contains information about the amount of time that the fork is allowed to compute before it is dismissed. Six bits of QR count the number of clock cycles remaining before the fork is dismissed, and three bits of QTAB point to a table specifying the length of time that the fork should be allowed to run when it is activated. All times in the discussion are measured in periods of the 60-cycle computer clock.

When a fork is activated, the number in QR is put into TTIME. This number is the unused portion of the fork's long time quantum. The long time quantum is the maximum length of time a fork can run before the scheduler checks for other forks to be run. This checking is necessary because of the possibility that some other routine is in a condition to be activated. The length of a long quantum is the same for all users. Simultaneously, with the assignment of a new long quantum, the user is assigned a new short time quantum.

The short time quantum is the minimum length of time a fork will run before the scheduler checks for other forks to be run that were dismissed for I/O operations. Both TIME (short quantum) and TTIME (long quantum) are decremented at every clock cycle.

When a fork is activated, it is first allowed to run for a short quantum. During this time it cannot be dismissed except by its own request.

When TIME goes negative, a short quantum has expired, and a word called ACTR is checked to determine whether any fork that is dismissed for I/O can be run. If ACTR is still negative the fork is allowed to continue. At each subsequent clock cycle the fork may be dismissed if any fork dismissed for I/O is ready to run. It may also be dismissed when the long quantum is exhausted if any other forks are waiting to run. In this case, it is said to be dismissed for long quantum overflow. If ACTR indicates that another fork dismissed for I/O is ready to run at the end of the short quantum, the fork is dismissed for short quantum overflow.

To allow an efficient implementation of this scheme ACTR is incremented by every routine that takes any action that will allow a fork that was previously waiting for I/O to run.

Since ACTR is set to -1 when a fork is activated, this means that the clock interrupt can execute the following code in order to check both the conditions which may require further action.

```
SKR    TIME
BRU    *+3
SKN    ACTR
BRU    *+3      Ready to dismiss
SKR    TTIME
BRI             Return to program
```

If ACTR is positive or the short quantum has not run out, it is of course ignored, as noted in the above example.

When a fork is dismissed for I/O, TTIME is put into QR. When the fork is reactivated, TTIME is set from QR. TIME is reset to the full short quantum. That is, the long quantum is allowed to run down while a program computes, regardless of whether it has to wait for I/O between computations. However, a fork is always given a full short quantum. If a fork is dismissed for quantum overflow, it is given a new long quantum when it is reactivated.

There are two operations available to the user that are connected with the quantum overflow mechanism. BRS 45 causes the user to be dismissed as though he had overflowed his quantum. BRS 57 guarantees to the user upon return at least 16 msec of uninterrupted computation. This feature is implemented by dismissing the user if less than 16 msec remain in his quantum.

Ordinarily, the code that is being executed at any particular instant is that belonging to the currently active fork. This situation may be disturbed, however, by the occurrence of interrupts from I/O devices. These interrupts cause the computer to enter system mode and are processed independently of the currently running program. The interrupts never take direct action to disturb the running of the active fork, although they may set up conditions in memory that will cause some other fork to be activated when the presently running one is dismissed. Interrupt routines always run in system mode.

Other codes that may be running but not belonging to the currently active fork are the system programmed operators (SYSPOPS) or branch system routines (BRS). These routines are not reentrant and, therefore, can not be dismissed by the clock. To ensure that they will not be, the convention is established that the clock will not dismiss a program running in system mode. To guarantee that a user will not monopolize the machine by executing a large number of SYSPOPS, the user mode trap is turned on when the clock indicates that a fork is to be dismissed. The trap will occur and cause dismissal as soon as the fork returns to user mode.

The PACT word called PTEST contains the activation condition for a currently inactive fork. The condition for activation is contained in the six opcode bits of this word, while the address field normally contains the absolute address of a word to be tested for the specified condition. It is possible, however, for the address to hold a number indicating which program interrupt has occurred. Note that the value 7 given in Table 1 can be used for forks, and not all conditions pertain to activation. For example, value 71 implies that the fork is already in operation.

The following conditions are possible in PTEST.

Table 1. Activation Conditions for Currently Inactive Fork

Bits 3-8	Activation Conditions
0	Word greater than 0
1	Word less than or equal to 0
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.
5	Interrupt occurred. The address contains the number of the interrupt which occurred.
6	Word less than or equal to Real-Time.
7	Special address = 0 dead 1 running 2 BRS 31 3 BRS 106 4 Executive BRS 5 BRS 109 6 BRS 9 (User Program)
10	Do not activate
11	Word 20000000 – 0 (buffer ready)
12	Word less than 0

An Executive program can dismiss itself explicitly by putting a queue number (0 to 3) in X, a dismissal condition in B and executing BRS 72. The address of a dismissal condition must be absolute.

There is normally one running fork in the system, i.e., a fork that is executing instructions, or will be executing instructions after the currently pending interrupts have been processed. An active fork (i.e., a PACT entry) that is not running is said to be dismissed, and is kept track of in one or two ways.

If it is dismissed with BRS 9, 31, 106 or 109 (see Chapter 3) it is said to be in "limbo" and is pointed to only by the PFORK, PDOWN, and PPAR of the neighboring forks in the fork structure.

If it has been dismissed for any other reason, it is on one of the schedule queues. There are four queues of dismissed programs. In order, they are

- QTI Programs dismissed for teletype input/output
- QIO Programs dismissed for other I/O
- QSQ Programs dismissed for exceeding their short quantum
- QQE Programs dismissed for exceeding their long quantum.

Programs within the queues are chained together in PNEXT, and the last program in each queue points to the beginning of the next queue.

When it is time to activate a new program, the old program is put on the end of the appropriate queue. The schedule then begins at QTI and scans through the queue structure looking for a program whose activation condition is satisfied. When one is found, it is removed from the queue structure and turned over to the swapper to be read in and run. If there are no programs that can be activated, the scheduler simply continues scanning the queue structure.

Programs reactivated for various reasons having to do with forks (interrupts, escapes, panics) are put onto QIO with an immediate activation condition. They take priority over all programs dismissed for quantum overflow.

THE PHANTOM USER

There is a permanent entry on the teletype queue for an entity called the phantom user. The activation condition for this entry is a type 4 condition that tests for two possibilities:

1. The cell PUCTR is nonzero.
2. Three seconds have elapsed since the last activation of the phantom user for this condition.

When the phantom user is activated by (2.) it scans the system checking that everything is functioning properly. In particular, it checks that the W-buffer has not been waiting for an interrupt for an unusual length of time, and that all teletype output is proceeding normally. If the phantom user is activated by (1.), it scans the phantom user queue looking for tasks to do. A phantom user queue entry is displayed in Figure 2. It is essentially an abbreviated PAC table entry.

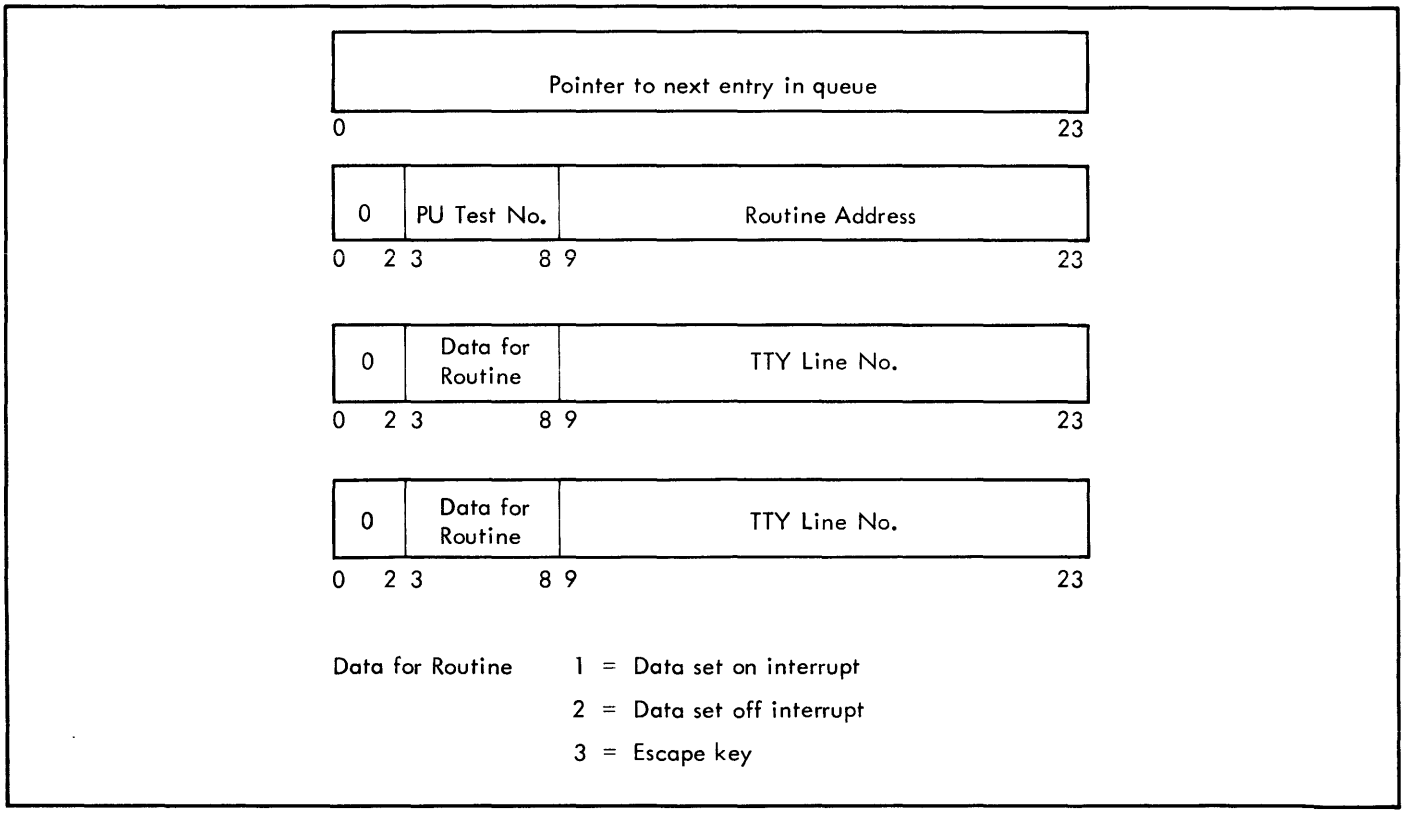


Figure 2. Phantom User Queue Entry

Such an entry is made when the system has some activity that it wants to carry out independently of any user PAC table entry, test for tape ready (on rewind), or card reader ready, and processing of escapes (an interrupt routine type of activity, but too time-consuming). The second word of the entry is the activation condition. PUCTR contains the number of entries on the phantom user queue.

The pointers or counters listed below are utilized by the phantom user to perform its tasks.

PUCT	Beginning of phantom user queue.
FPULST	First free entry in PU queue.
PUBPTR	Pointer to first active entry. Last entry points to PUBPTR.
PUCTR	Number of PU entries.
PUEPTR	Last PU entry.
PUCTR1	Entry counter during PU processing.
PUCPTR	Pointer to active entry during PU processing.
PUPAC	PACPTR of task being processed by PU.

3. FORKS AND JOBS

CREATION OF FORKS

A fork may create new, dependent, entries in the PAC table by executing BRS 9. This BRS takes its argument in the A register, which contains the address of a seven-word panic table with the format given in Table 2.

Table 2. 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 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

The panic table address must not be the same for two dependent forks of the same fork, or overlap a page boundary. If it is, BRS 9 is illegal. The first six bits of the A register have the following significance as shown in Figure 3.

Bit	Significance
0	Make fork Executive if current fork is Executive.
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 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 Exec type 1 if current fork is Exec.

Figure 3. Significance of Bits in A Register

When BRS 9 is executed, a new entry in the PAC table is created. This new fork is said to be a fork of the fork creating it. This is called the controlling fork. The fork is said to be lower in the hierarchy of forks than the controlling fork. The latter may itself be a fork of some still higher fork. A job may have a maximum of eight forks including the executive. The A, B and X registers for the fork are set up from the current contents of the panic table. The address at which execution of the fork is to be started is also taken from the panic table. The relabeling registers are set up either from the current contents of the panic table or from the relabeling registers of the currently running program. An executive fork may change the relabeling. A user fork is restricted to changing relabeling in the manner permitted by BRS 44. The status word is set to -1 by BRS 9. The fork number that is assigned is kept in PIM. This number is an index to the fork parameters kept in the TS block.

The fork structure is kept track of by pointers in PACT. For each fork PFORK points to the controlling fork, PDOWN to one of the subsidiary forks, and PPAR to a fork on the same level. All the subsidiary forks of a single fork are chained in a list.

If the fork executing a BRS 9 is a user fork, it is dismissed until the lower fork terminates. If it has Exec status, it continues execution at the instruction after the BRS 9. The fork established by the BRS 9 begins execution at the location specified in the panic table and continues independently until it is terminated by a panic, which is a signal to the system to break execution of a fork. The panic is connected to its controlling fork in three ways:

1. The controlling fork may examine its state and control its operation with the following six instructions:

   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 the 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.

   All of these instructions require the panic table address of the fork in A. They are illegal if this address is not that of a panic table for some fork.

   BRS 31 and BRS 32 return the status word in the X register, as well as leaving it in the panic table. This makes it convenient to do an indexed jump with the contents of the status word. BRS 31 returns the panic table address in A.

   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 a 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. In addition, there is some interaction when the subsidiary fork attempts to acquire memory.

MEMORY ACQUISITION

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 propagated downward 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.

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 above. When any of these conditions occur, the PACT entry for the fork being terminated is returned to the free program list. The status of the fork is read into its panic table in the controlling fork. If the fork being terminated has a subsidiary fork, it too is terminated. This process will cause the termination of all the lower forks in the hierarchy.

The panic that returns a status word of zero 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. In the absence of such a declaration the highest user fork is terminated. When a fork is terminated in this way its controlling fork becomes the one to be terminated. If a user fork is terminated by escape, the teletype input buffer is cleared. If the controlling fork of the one terminated is executive, the output buffer is also cleared.

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

If there is only one fork active, 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 occurring in the meantime will be stacked. A second one will be ignored. A program which is running with escape turned off is said to be nonterminable and cannot be terminated by a higher fork. BRS 26 skips if there is an escape pending.

If two escapes occur within about .12 seconds, the entire fork structure will be cleared and the job left executing will be the top level Executive fork. This device permits a user trapped in a malfunctioning lower fork to escape. Closely spaced escapes can be conveniently generated with the repeat button on the teletype. This type of escape will cause a user to lose memory, and should be followed by a RESET. An off interrupt from the teletype is treated like a high-speed escape.

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.

   As an extension of this system there is one way in which several forks may be terminated at once by a lower fork. This may be done by BRS 73, which 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 passed by 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.

   a. Machine instructions that are privileged.

   b. SYSPOPs that are forbidden to the user or that have been 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 that 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.

JOBS

Every complete fork structure is associated with a job. The job is the fundamental entity thought of as a user of the system, from the system's own point of view. The job number appears in the PAC table entry for every fork in the job's fork structure.

In addition, there are several tables indexed by job numbers. These are displayed in Figure 4 and indicate what is specifically associated with each job.

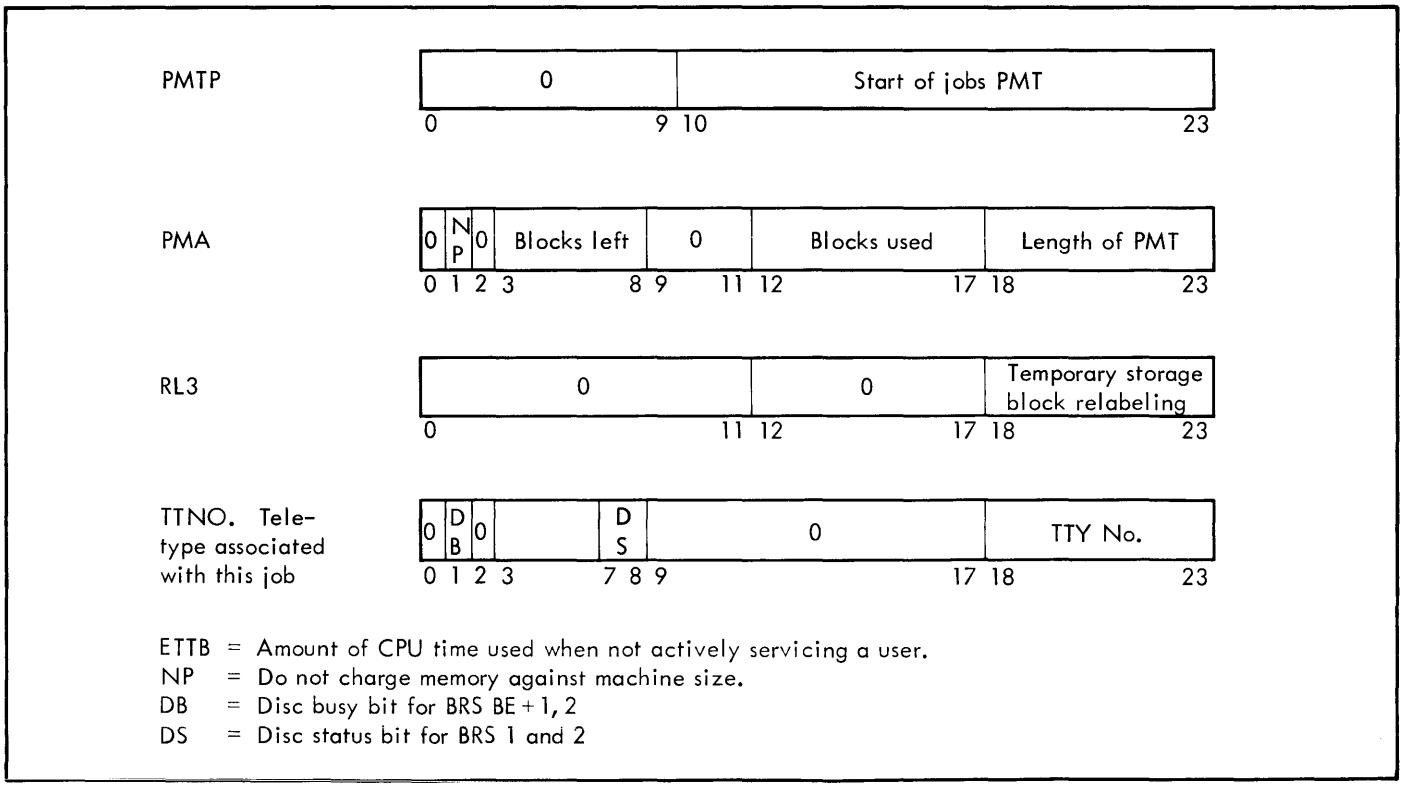


Figure 4. Job Tables

4. PROGRAM INTERRUPTS

A facility is provided in the Monitor to simulate the existence of 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 BRS 78 with an 11-bit mask in the A register. This causes the appropriate bits in PIM to be set or cleared according to whether the corresponding bit in the mask is 1 or 0. Bit 4 of A corresponds to interrupt number 1, etc. No other action is taken at this time. When an interrupt occurs (in a manner to be described) the execution of an SBRM* to location 200 plus interrupt number is simulated in the fork which armed the interrupt.

Note: The program counter which is stored in this case is the location of the instruction being executed by the fork which is interrupted, not the location in the fork which causes the interrupt. The proper return from an interrupt is a BRU to the location from which the interrupt occurred. This will cause the proper return in all cases including interrupts out of input/output instructions.

A fork may generate an interrupt by executing BRS 79 with the number of the desired interrupt in the A register. This number may not be one, two, three, four, or eleven. The effect is that the fork structure is scanned, starting with the forks parallel to the one causing the interrupt and proceeding to those above it in the hierarchy (i.e., to its ancestors). The first fork encountered during this scan with the appropriate interrupt mask bit set is interrupted. 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.

Interrupts 1 and 2 are handled in a special way. If a fork arms interrupt 1, a program panic (BRS 10 or escape key) that would normally terminate the fork which has armed interrupt 1, will instead cause interrupt 1 to occur, that is, will cause the execution of an SBRM* to location 201_8. 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 which 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, it 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 (end of record, end of file and error conditions can do this).

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.

Whenever any interrupt occurs, the corresponding bit in the interrupt mask is cleared and must be reset 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.

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 is up, the fork will be interrupted.

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

5. THE SWAPPER, MEMORY ALLOCATION AND RAD ORGANIZATION

RELABELING

Because of the necessity in various parts of the system for relabeling registers which do not change with time, the user has been denied any access to ordinary relabeling. However, he is given access to pseudo-relabeling. His pseudo-relabeling registers consist, as do the ordinary relabeling registers, of eight six-bit bytes. Each one of these bytes points to an entry in the user's pseudo-memory table (PMT) and not to a real page of memory. This table may contain up to 64 words, each one specifying a certain 2K block of memory, herein referred to as a page. The first version of the system, however, will allow access to only 14 words. The possible forms of an entry in the pseudo-memory table are shown in Figure 5.

When it is necessary to activate a user, his pseudo-relabeling registers are used to read out the proper bytes from PMT and construct a list of pages that need to be read in from the RAD. When this list is constructed, the current state of core is examined to determine whether any pages need to be written out to make room for those which must be read in. If so, a list of pages to be written out is constructed. The RAD command list is then set up with the appropriate commands to write out and read in the necessary pages. In the scan which sets up the RAD read commands, the swapper collects from PMT or SMT the actual absolute memory addresses of the page called for by the pseudo-relabeling and constructs a set of real relabeling registers which it puts in two fixed locations in the monitor (RRL1 and RRL2). It then outputs these relabeling registers to the hardware and activates the program.

There is also a system parameter called NCMEM. Pseudo-relabeling bytes with values from 1 to NCMEM-1 (0 means an unassigned page) actually refer directly to the first NCMEM-1 pages of SMT, the shared memory table, and the user's own PMT is addressed beginning at NCMEM. The "common" portion of SMT is used to hold the most common subsystems.

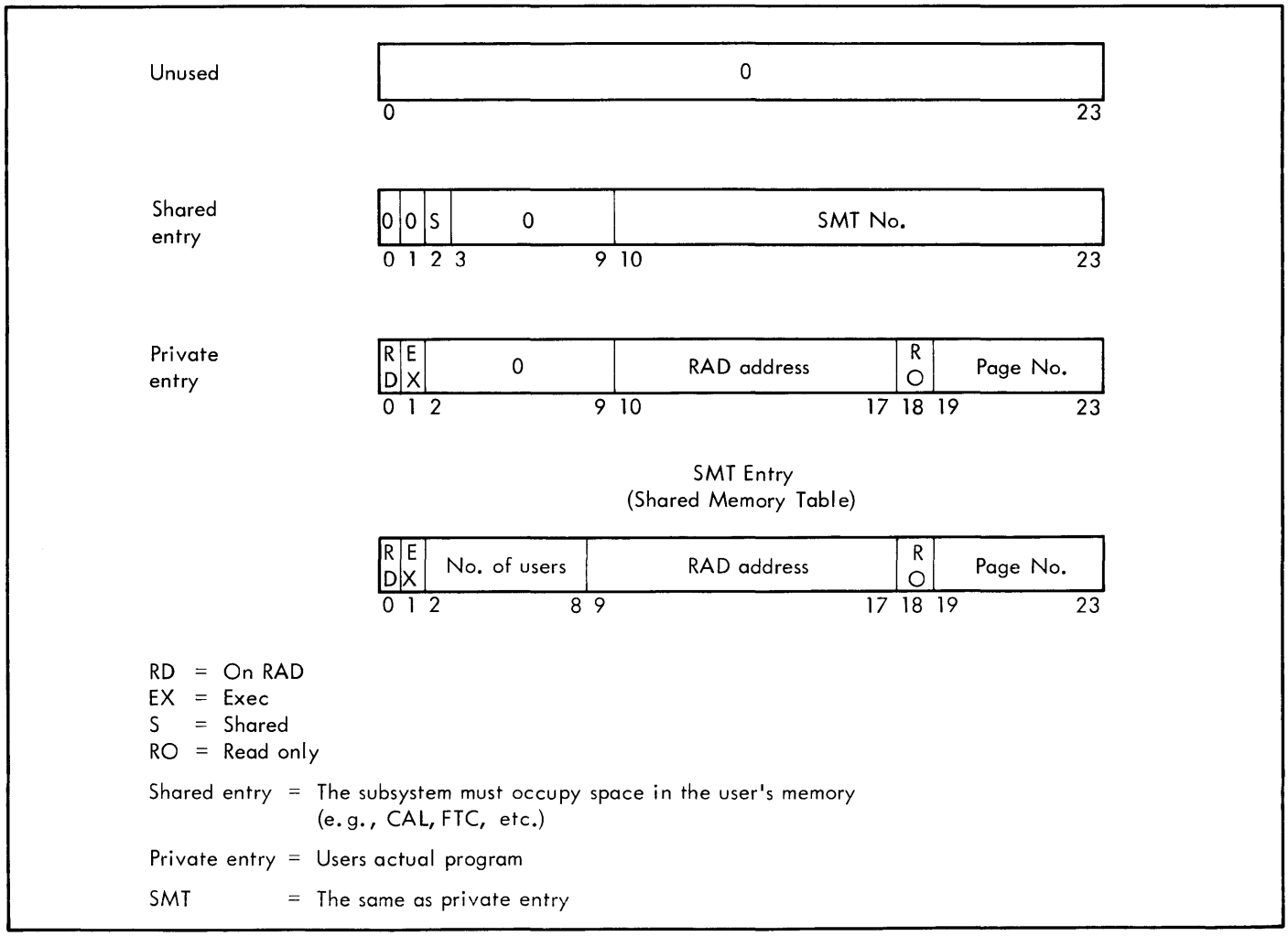


Figure 5. Pseudo Memory Table PMT Entries

There are two BRSs that permit the user to read and write his pseudo-relabeling. BRS 43 reads the current pseudo-relabeling registers into A and B. BRS 44 takes the contents of A and B and puts them into the current pseudo-relabeling registers. An executive program may set the relabeling registers in arbitrary fashion by using this instruction. A user program, however, may add or only delete pages that do not have the executive bit set in PMT. This prevents the user from gaining access to Executive pages whose destruction may cause damage to the system. Note that the user is doubly restricted in his access to real memory, because he can only access real memory that is pointed to by his pseudo-relabeling, and because he is only allowed to adjust those portions of his pseudo-relabeling that are not Executive type.

The user can also set the relabeling of a fork when he creates it (see Chapter 3). The same restrictions on manipulation of Executive pages also apply.

The system maintains a pair of relabeling registers that the Executive and various subsystems consider as the user's program relabeling. For the convenience of subsystems, an Executive program can read these registers with BRS 116 and set them with BRS 117.

The memory allocation algorithm is described in Chapter 3. A user can release a page that is in his current relabeling by putting any address in that page into A and executing BRS 4. The PMT entry for the page is removed and in any other fork which has this PMT byte in its relabeling, the byte is cleared to 0.

Equivalent to BRS 4 is BRS 121, that takes a pseudo-relabeling byte in A rather than an address. An inverse operation is BRS 120, which takes a pseudo-relabeling byte in A, generates an illegal instruction trap if the corresponding PMT entry is occupied, and otherwise obtains a new page and puts it in that entry. This is an exec-only operation.

A word of PMT whose first three bits are 001 contains a pointer to the shared memory table, SMT. An entry in SMT looks exactly like an unused or private entry in PMT. It refers to a page of memory which has a fixed location on the RAD and may be referred to by more than one program.

By putting an index in SMT in A and executing BRS 69, a pointer to the specified location in SMT is put into the first free byte of a user's PMT and the byte number is returned in A.

The user may declare a page read-only by executing BRS 80 with the pseudo-relabeling byte number of the page in A and with bit 0 of A set. To make a page read-write, bit 0 of A should be clear. Bit 0 of A will be reset if the page was formerly read-write or set if it was formerly read-only. If the program doing this is not an Executive program, then the page must not be an Executive page. Only an Executive can make a read-only PMT entry which points to SMT into a read-write entry, for obvious reasons. The significance of a read-only page to the swapper is that it need not be rewritten on the RAD when it is removed from memory.

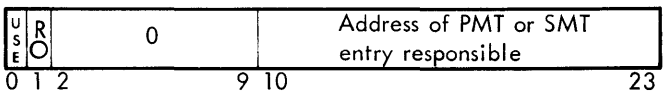
A RAD is divided into blocks of 32K. Each user is assigned a block depending on his job number. The first page in each block is always the user's TS page. Each block of 32K consists of eight bands with two pages per band. The list of swapping commands alternates pages whenever possible to minimize swap time. A bit map is kept in the TS page which maps the user's 32K. When the user requires more memory, the free page nearest the beginning of his block is taken. The first several blocks on the first RAD contain the subsystem, Exec and swappable Monitor pages.

It should be noted that whenever a user is reactivated, all of the memory in his current relabeling registers is brought in. The user does, however, have considerable control over precisely what memory will be brought in because he can read and set his own relabeling registers. Therefore, he may establish a fork with a minimal amount of memory in order to speed up the swapping process if this is convenient.

To make a page executive, execute BRS 56 with the same argument as for BRS 80, Make Page Read Only. This instruction is legal only for executive type programs.

The system keeps track of the state of real core with two tables called the real memory table (RMT) and the real memory use count table (RMC). An RMC entry is -1 if a page is not in use; otherwise, it is one less than the number of reasons why it is in use. Every occurrence of this page in the relabeling of a process which is running or about to be run counts as such a reason. In addition, other parts of the system can increment an RMC word to lock a page in core. No page with non-negative BRM can be released by the swapper.

The format of an RMT entry (one per real page) is



USE = in use
RO = read only

There is one other table indexed by real memory, called the real memory aging table. It is used by the swapper to decide what pages are to be swapped out. It does this first by right shifting one bit for every entry in the RMA. Then it sets bit one for every real page that was computed from the pseudo-relabeling from which the swapper was entered. The RMA entries with the lowest values are the ones selected for swapout.

The swapper also contains a device called the simulated associative memory or SAM, which contains pseudo-relabeling and real relabeling for the most recently used maps. It serves to reduce the amount of time needed for map changing when little swapping is taking place. It is cleared whenever a RAD read takes place, since this changes the contents of real memory and potentially invalidates all real relabeling registers.

Two BRSs exist for reading and writing pages at specified places on the RAD. They are, of course, restricted to executive programs. To read a page, put the RAD address into B and the core address in A and execute BRS 104. Use BRS 105 to write a page. RAD errors cause these instructions to generate illegal instruction panics.

6. MISCELLANEOUS FEATURES

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

He can read the real-time clock into A and the system start-up date and time into B by executing BRS 42. The number obtained increments by one every 1/60th of a second. Its absolute magnitude is not significant. An Exec fork can read the elapsed time counter for the user into A by executing a BRS 88. This number is set to 0 when he enters the system and increments by 1 at every 1/60th second clock interrupt at which his fork is running.

To obtain the date and time, he can execute BRS 91. This puts string pointers into 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 in question is terminated by executing BRS 109.

A fork can test whether it is executive or not by executing 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 the BRS skips on return.

An Executive fork can dismiss itself explicitly (see Chapter 2).

There are two operations designed for Executive BRSs which operate in user mode with a map different 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. A memory panic can occur, in which case it appears to the calling fork that it comes from the BRS instruction.

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

There are switches in the monitor which can be set by an Exec 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 Exec fork with 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 Exec fork can also force a new page to be read from the RAD with BRS BE+15. It takes an SMT pointer in A.

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

An Exec fork can crash the system with BRS BE+8.

7. TELETYPE INPUT/OUTPUT

An outline of the implementation of the teletype operations should clarify the exact disposal of the characters that are being read and written. Every teletype has attached to it information that is given below in Table 3.

As characters are output by the program, they are added to the output buffer, that can be regarded as logically independent from the input buffer in spite of the fact that it resides in the same words. The characters are then output by the teletype interrupt routine as rapidly as the teletype will accept them.

Also associated with the teletype is a buffer that contains input and output characters in the following format

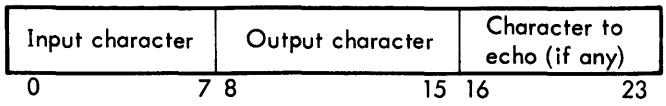


Table 3. Teletype Table

TIS2 Number of characters in input buffer

TIS4 Next available space in input buffer (pointer)

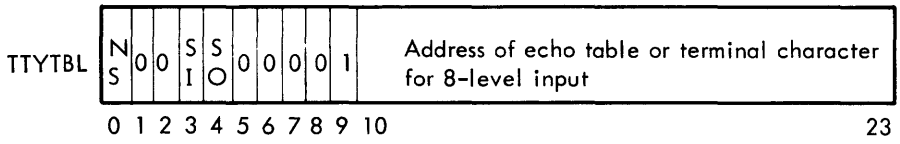
TIS5 Next filled space in input buffer (pointer)

TOS2 Number of characters in output buffer; -1 = inactive

TOS3 <0 = Not in multiple blank mode; 400 = just saw 135 (multiple blank character); other = number of blanks

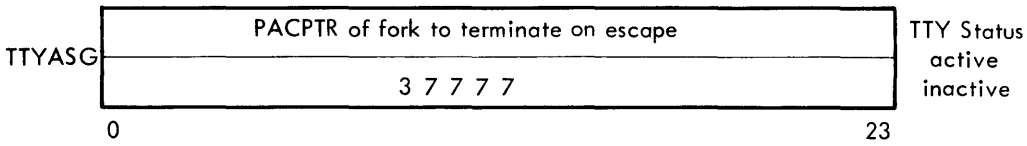
TOS4 Next filled space in output buffer (pointer)

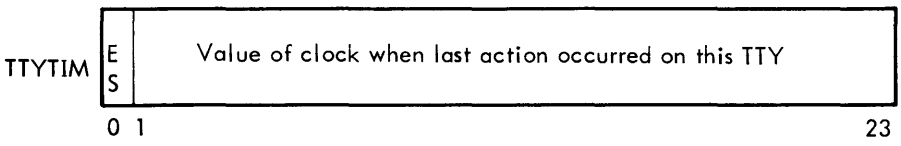
TOS5 Next available space in output buffer



TTYFLG Don't listen for input (except escape) when 0. Set when input buffer is full.

TTYBRK Waiting for break character when -1
Waiting for any character when 37777777





NS = not 8-level
SI = 8-level input
SO = 8-level output
ES = last action was input of escape

These buffers are called character ring buffers (CRBs) and they are not necessarily associated with teletypes.

When a character is typed in on a teletype, it is converted to internal form and added to the input buffer unless it is an escape on a controlling teletype. The treatment of escapes is discussed in Chapter 3. The echo table address is then obtained from TTYTBL. The echo table determines what to echo and whether or not the character is a break character. The available choices of echos and break characters are discussed later in this section. If the character is a break character, and if a user's program has been dismissed for teletype input, it will be reactivated regardless of the number of words in the input buffer. In the absence of a break character, the user's program is reactivated only when the input buffer is nearly full.

If the teletype is in the process of outputting (TOS2>-1), then the character to be echoed is put into the last byte of the buffer word that contains the input character. When the character is read from the buffer by the program, the echo, if any, will be generated. This mechanism, called deferred echoing, permits the user to type in while the teletype is outputting without having his input mixed with the teletype output.

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 wishes to compute the echo itself.

Normally a carriage return and line feed are both echoed if either is received from a teletype. However, only the first one received is sent to the program and if the other one is a also received it is ignored. A program can receive both by issuing BRS BE+11. If A is negative, both characters will be sent to the program. If A is positive, only the first character will be sent to the program.

If either line feed or carriage return is output by a program both are sent to the teletype unless the carriage is at the left margin. In this case, only a line feed is output for either a carriage return or a line feed. If a program wishes to send only one character, it should output 102B for line feed or 105B for carriage return.

To set the echo table, put the teletype number, or -1, in X and the echo table number in A and execute BRS 12. Note that BRS 12 is also used to turn on 8-level mode (see below). To read the echo table number into A, put the teletype number, or -1, in X and execute BRS 40. This operation returns the echo table number in A. If the teletype is in 8-level input mode, the sign bit of A is set and the terminal character is in A.

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.

The contents of the other internal registers are preserved by this and all the other teletype SYSPOPS and BRSs.

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. In addition to the ordinary ASCII characters, all teletype output (other than 8-level) operations will accept 135 (octal) as a multiple blank character. The next character will be taken as a blank count, and that many blanks will be typed.

The TTYTIM cell in the teletype table is set to the current value of the clock whenever any teletype activity (interrupt or output SYSPOP) occurs. The top bit is left clear unless the activity is an escape input. This cell is checked by the escape processor to determine whether the escape should reset the job to the system exec (see Chapter 3).

Every teletype in the system is in one of two states:

1. It may be the controlling teletype of some user's program. It gets into this state when a user logs in on it. Controlling teletypes are also known as attached teletypes.
2. It may be completely free.

The status of the teletype is reflected by the contents of TTYASG. If the teletype is free, TTYASG contains 3777B. If it is a controlling teletype, TTYASG contains the PACPTR of the fork to terminate on escape.

A teletype becomes a controlling teletype when an "ON" interrupt (from that line) is received by the computer. This indicates that someone has called that line. The user then has one-and-a-half minutes to log in before the system hangs up the line again. The system checks for carrier presence on a line before sending out any characters. To do this a system fork may issue BRS BE+3 with the line number to check in A.

The user may disconnect the line by hanging up the phone. BRS 112 is executed when an "OFF" interrupt is received by the system or when a user logs out. If an "OFF" interrupt has been received, BRS 112 merely makes the line available again. However, if a user has logged out without hanging up the phone, BRS 112 makes the teletype the controlling teletype for another job immediately and the next user can log in without dialing the system again. BRS 112 takes the

job number associated with the teletype in X. A job may terminate itself. This operation also releases all teletypes attached to the job. BRS 112 requires system status.

An exec fork can turn a line on or off by issuing BRS BE+6. It takes the line number in A and turns it on if B is negative or off if B is positive.

The user has considerable control over the state of the teletype buffers for the controlling teletype. In particular, he may execute the following BRSs. All these take the teletype number in X. Recall that -1 may be used for the controlling teletype.

BRS 11	clears the teletype input buffer.
BRS 29	clears the teletype output buffer.
BRS 13	skips if the teletype input buffer is empty.
BRS 14	waits until the teletype output buffer is empty, but not until the interrupt has been received for the last character.

Special provision is made for reading 8-bit codes from the teletype without sensing escape or doing the conversion from ASCII to internal which is done by TCI. 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 mode. Teletype output is not affected.

A parallel operation, BRS 85, is provided for 8-level output. BRS 86 returns matters to the normal state, as does any setting of the echo table.

To simulate teletype input, the operation

```
STI    =teletype number    or =-1
```

is available. STI puts the character in A into the input buffer of the specified teletype. Either the teletype number must be the controlling teletype or the fork issuing STI must be a system fork.

8. ORGANIZATION OF DISC AND BUFFER DEVICES

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 with 64 words per sector.

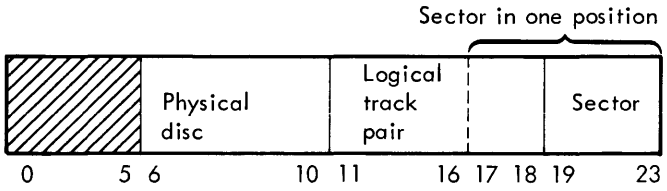
The disc files used by this system consist of 8 to 32 physical discs, with each disc having movable arms. The arms have 64 positions numbered 0 to 63 and each arm position on a disc can access 8, 192 words. Each arm position contains four pages (a page is 1/4 of an arm position) and one page contains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

For example, if the total number of arm positions are multiplied times the words per arm positions, the total number of words per disc can be calculated (e.g., 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 given in Figure 6 illustrates the disc layout. Note that the 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 one 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 an arm position 0 at disc 0 represents the file directory or the individual's user number. "Acct 1 UAD" (arm position 1 at disc 4) is the user's account directory that the system accesses for the user.

The format for the disc address word is as follows.



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 actually specify one of the 64 positions of the access arm.

Bits 17-18 actually 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.

Every file has one or more index blocks that contain pointers to the data blocks for the file. An index block is a 256 word block, as are all other physical blocks in the file storage area. Only the first 128 words of the index block are used. A couple of additional words are used to chain the index blocks for any particular file, both forward and backward. The index blocks for a file contain the addresses for all the physical blocks used to hold information for the file.

Available storage in the file area of the disc is kept track of with a bit table. If a bit in this table is set, it indicates that the corresponding block on the disc is free. The bit map is set every time the system is brought up to agree with the files in the file directories. To set the bit map, BRS BE+5 is used. It requires an index block pointer (MOD4) in A. When all files have been checked, the BRS is called with a -1 in A, 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 7.

The layout of a file buffer shows the buffer proper, and the layout of the index block buffer. The pointers associated with it are used only by disc files and are present in all cases.

The temporary storage page that is associated with each job is always the first entry in the job's PMT. This page is used to hold information about the user and for the system's temporary storage for that user. It also has room for three buffers. The pseudo-relabeling for this TS page is held in a table called RL3 which is indexed by job number, and is put into the monitor map whenever any fork belonging to that job is run. This TS page is always relabeled into 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 an input buffer. The second word points to the end of the data and is incremented as data is written into an output buffer. When the buffer is in a dormant state, both words point to the first word of the buffer. Whenever any physical

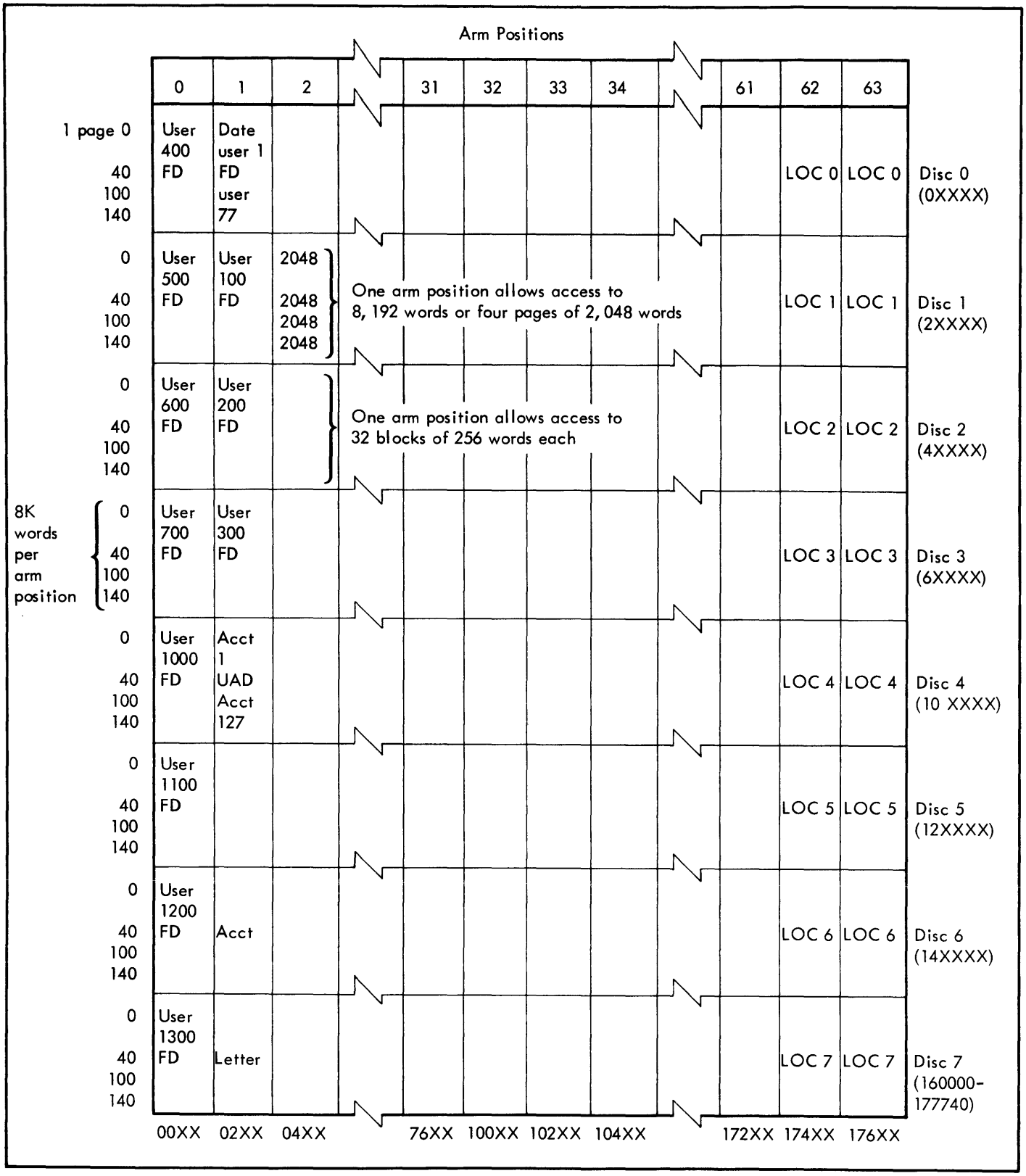
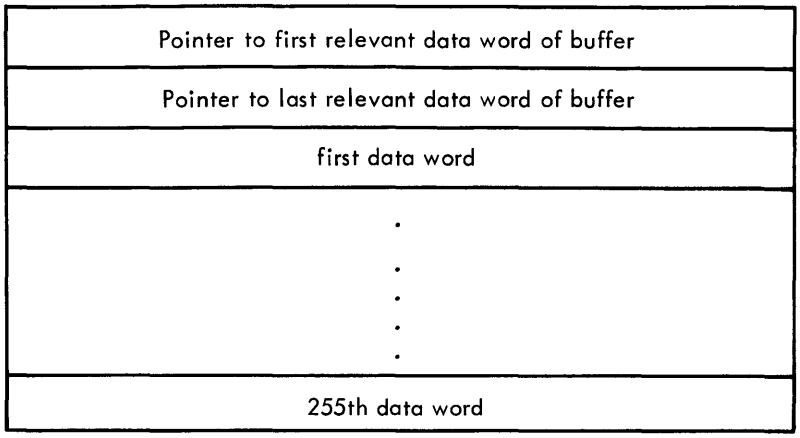
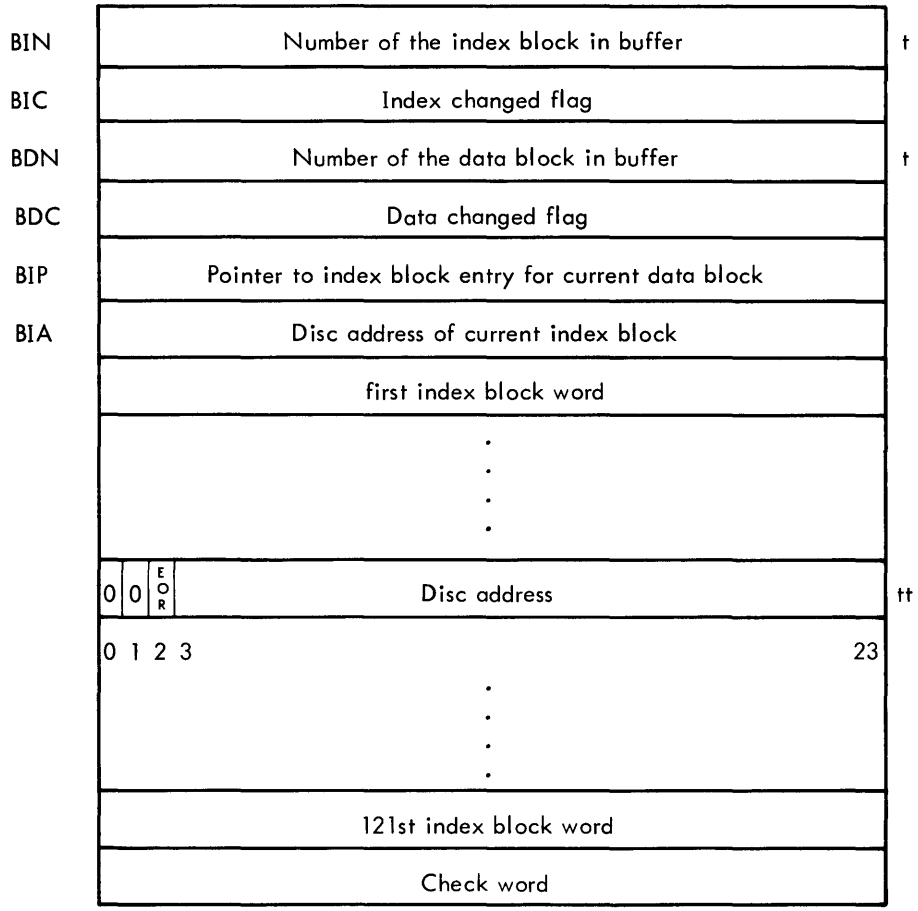


Figure 6. Disc Map

Layout of a File Buffer



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



[t]Random files only

[tt]Index block word format. EOR = end of record flag.

Figure 7. Buffers

I/O operation is completed, the first point contains the address of this word.

DEVICES

Every different kind of input/output device attached to the system has a device number. The numbers assigned to specific devices are given in Chapter 9. The various tables indexed by device numbers are described here. The entries in these tables addressed by a specific device number together with the unit number (if any) and the buffer address, completely define the file. All this information is kept in the file control block which is addressed by the file number.

The table indexed by device number are shown in Figure 8.

Note that multiplicity of bits which specify the characteristics of the device. A device may be common (shared by users, who must not access it simultaneously; e.g., tape or cards) or not common (e.g., disc); this characteristic is defined by NC (not common). It may have units; e.g., there may be multiple mag tapes. The U bit specifies this. The DIU word indicates which file is currently monopolizing the device; in the case of a device with multiple units, DIU points to a table called ADIU which contains one word for each unit.

The major parameters of a device are

the opening routine, that is responsible for the operation necessary to attach it to a file.

the GPW routine, that performs character and word I/O,

the BIO routine, that performs block I/O.

The minor parameters are

maximum legal unit number,

physical record size (determining the proper setting of buffer pointers and interlace control words for the channel),

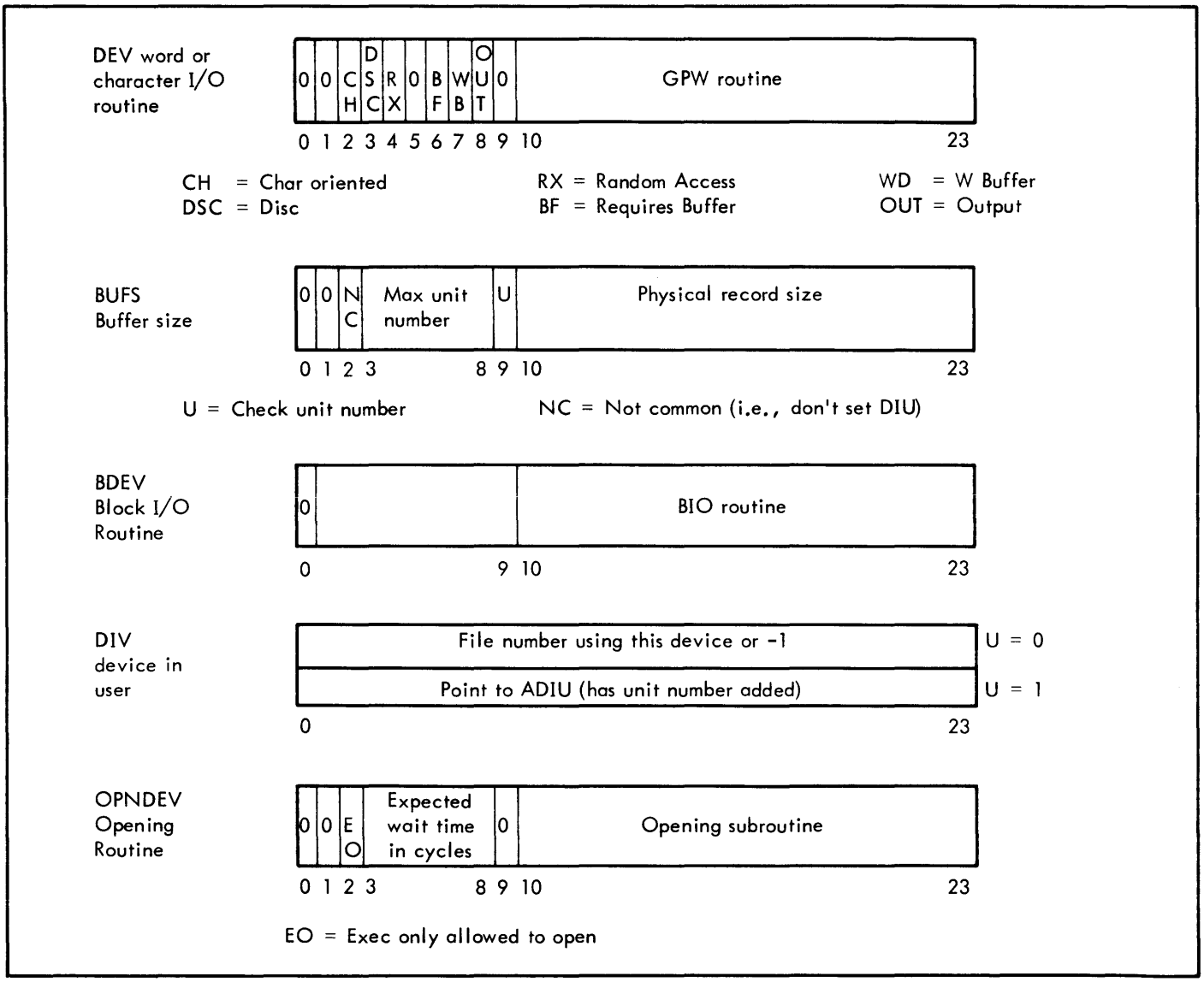


Figure 8. Device Tables

the expected time for an operation; the swapper uses this number to decide whether it is worthwhile to swap the user out while it is taking place.

SYSTEM DATA ON OUTER ARM POSITION OF DISC

Arm positions 62 and 63 contain systems which are loaded by a special routine that is kept on paper tape. 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.

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

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 take the word count in X. BRS BE+9 and BRS BE+10 always read or write a page (2K) from or to the disc.

9. SEQUENTIAL FILES

SEQUENTIAL DISC FILES

There are two basically different kinds of files that the user may write on the disc: sequential and random. A sequential file has a structure very similar to that of an ordinary magtape 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. Furthermore, the file may be instantaneously positioned to any specified logical record.

A sequential disc file may be opened by the following sequence of instructions:

```
LDX =device number, 8 (input) or 9 (output)
LDA Address of first index block
BRS  1
```

If the file is opened successfully, the BRS skips; otherwise it returns without skipping. Use of this BRS is restricted to users with system status. User programs may access disc files only through the Executive file handling machinery. BRS 1 can also be used to open other kinds of files which will be discussed later in this chapter.

If BRS 1 fails to skip, it returns in the A register for the following reasons:

- -2 too many files open – no file control blocks or no buffers available.
- -1 device already in use. For the disc, produced by an attempt to open a file for output twice.
- 0 no disc space left. This inhibits opening of output files only.

BRS 1 returns a file number for the file to the A register. This file number is a useful identification that the user has for the file. He may use it to close the file when he is done with it by putting it in the A register and executing BRS 2. This releases the file for other uses. BRS 2 is available to both user and Executive programs.

To close all his open files the user may execute BRS 8.

If the sign bit of A is set when the BRS 1 is executed, the file is made read-only. This means that it cannot be switched from input to output. If this bit is not set, then the instruction:

```
LDA =file number
LDB =1
BRS  82
```

will change the file to an output file regardless of its initial character. The instructions:

```
LDA =file number
LDB =1
BRS  82
```

are always legal and make the file an input file regardless of its initial character.

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 is specified by one SYSPOP. 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.

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.

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 or end of file

condition will set bits 0 and 8 or bits 0 and 7 in the file number (these are called flag bits) and return a 134_8 or 137_8 character, respectively. 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 input operation after the end of record, which signals the last record of the file. The user may generate an end of record while writing a file by 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_8 characters.

Mixing word and character operations will lead to peculiarities and 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 does skip. Note that a BIO cannot set both the EOR and the EOF bits, however, BIO is still implemented with considerable efficiency.

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 one 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 in that fork if it is armed.

The CTRL SYSPOP provides various control functions for sequential disc files. To use this operation execute the instructions:

```
LDA =control number
LDB =count, (if required)
CTRL=file number
```

Table 4 gives the available control numbers.

Table 4. Control Numbers

Control Number	Description
1	Write end of record on output or skip the remaining part of the logical record on input. This control does not take a record count.
2	Backspace (B) physical tape blocks.
3	Forward space (B) physical tape blocks.
4	Delete (B) tape blocks (legal on output only).
5	Space to end of file and backspace (B) physical tape blocks.
6	Space to beginning of file and forward space (B) physical tape blocks.
7	Insert logical record (legal on output file only). This control does not require record count.
8	Write end of file (output only).

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

```
LDA =file number
BRS  66
```

The index block for a sequential disc file contains one word for each physical record in the file. This word contains the address on the disc of the physical record in the bottom 21 bits. Bit 2 is set if the physical record is the last record of a logical record. A sequential file may have only one index block, or a maximum of 121 x 255 = 30, 855 words of data.

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. This also works for random files.

Three operations are available to executive programs only. They are intended for use by the system in dealing with file names and Executive commands.

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 as usual and the index block number in X. The read-only bit may be set (bit 0 of A) as usual and

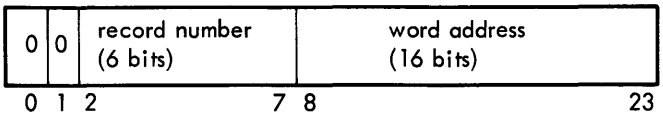
```
BRS  67
```

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

```
BRS  87
```

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

A single word of a sequential file may be directly addressed by specifying the logical record number and word number within the logical record. All the operations legal for random files (see Chapter 10) can also be used for sequential files with this convention. The format of the address is



OTHER SEQUENTIAL FILES

In addition to disc sequential files, the user has some other kinds of sequential files available to him. These are all opened with the same BRS 1:

```
LDX  =device number
LDA  =unit number
BRS   1
```

Available device numbers are given in Table 5.

Table 5. Device Numbers

Device	Device No.
Paper tape input	1
Paper tape output	2
Mag-tape input	4
Mag-tape output	5
Card punch Hollerith	6
Card punch binary	7
Line printer output	11
Card input Hollerith	12
Card input binary	13

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 as described in Chapter 9. Three error conditions apply to mag-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. The mag-tape is not available to the user as a physical device.

CTRL =1 (end of record)

is available for physical sequential files 3 and 5 (paper tape and mag-tape output). Several other controls are also available for mag-tape files only. These are given in Table 6..

Table 6. File Control for Mag-tape

Operational Control No.	Mag-tape File Control
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 type programs. I/O operations to the mag-tape may, of course, be executed by user programs if they have the correct file number.

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 already attached to some other job. BRS 119 releases a tape so attached.

It is possible for mag-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 mag-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 look more like other files in the system, the following transformations are made by the system on card input:

1. All non-trailing strings of 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 on the card are not transmitted to the program.
3. The card is not regarded as a logical record. However, the system generates the character 155 (carriage return) at the end of each card.

The result of this configuration is that the string of characters obtained by reading in a card deck 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 becomes not ready.

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

The phantom user's ten second routine checks to see whether a W-buffer interrupt has been pending for more than ten seconds. If so it takes drastic and ill-defined action to clear the W-buffer. BRS 114 also takes this drastic action; it can be used if a program is aware that the W-buffer is malfunctioning.

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 9.

CHARACTER BUFFERS

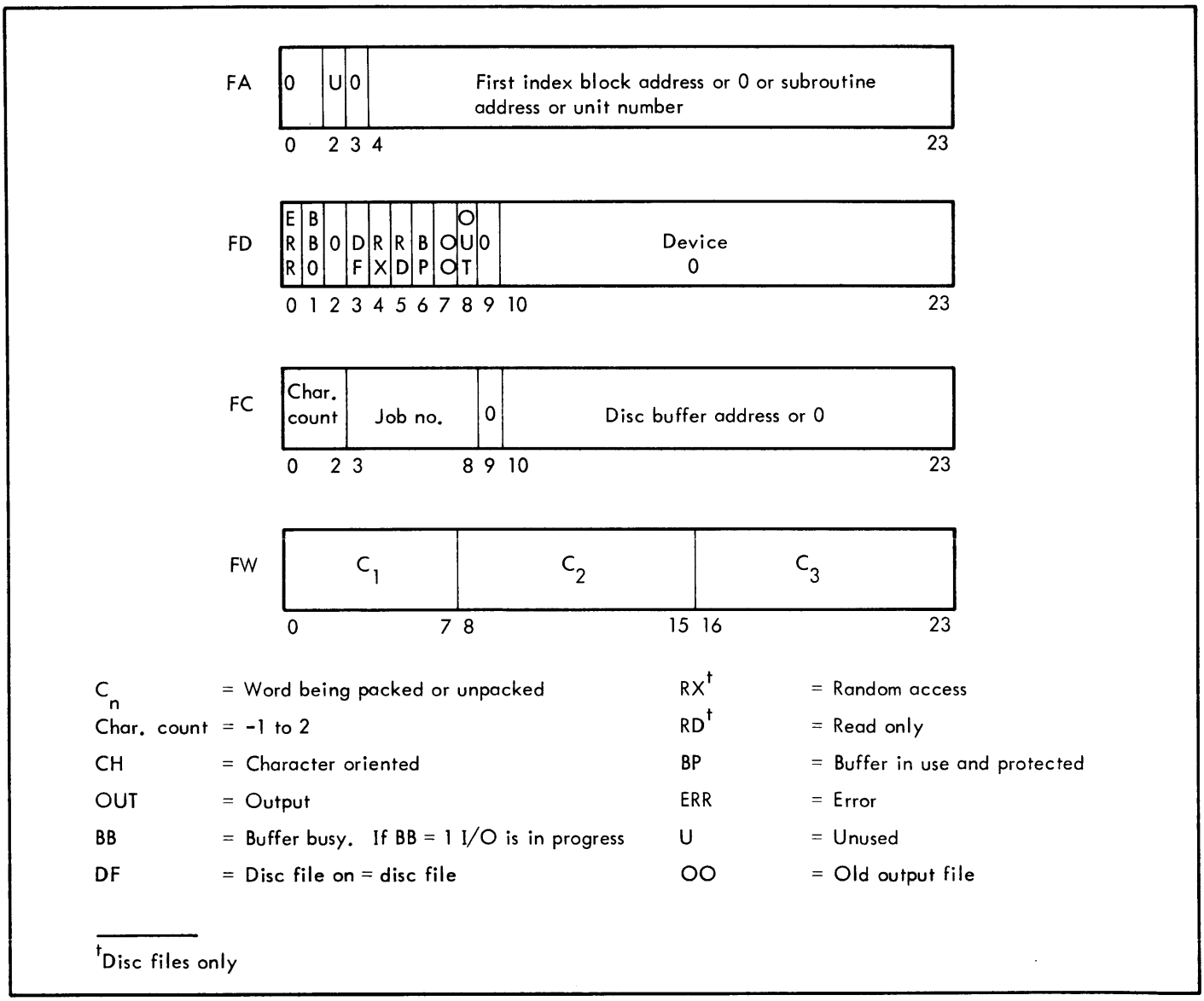
Chapter 7 describes the format of a teletype buffer. These buffers not only deal with the teletype but are capable of dealing with any character-oriented device. For this reason the character ring buffers are not directly indexed by the physical number of the teletypes to which they are associated. Instead, a table indexed by physical teletype number is used to obtain the buffer number.

PERMANENTLY OPEN FILES

There are a few 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

Figure 10. Fixed File Numbers



C_n	= Word being packed or unpacked	RX^t	= Random access
Char. count	= -1 to 2	RD^t	= Read only
CH	= Character oriented	BP	= Buffer in use and protected
OUT	= Output	ERR	= Error
BB	= Buffer busy. If BB = 1 I/O is in progress	U	= Unused
DF	= Disc file on = disc file	OO	= Old output file

[t]Disc files only

Figure 9. File Control Block

10. SUBROUTINE FILES

In addition to the previously mentioned operations for performing input-output through physical files, a facility is provided in 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, simply by 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 opcode field of the parameter word indicates the characteristics of the file. It may be one of the following combinations:

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 or a character oriented subroutine. The system will take care of the necessary packing and unpacking of characters. BIO is also acceptable.

The opening of a subroutine file does nothing except to create a file control block and return a file number in the A register. When an I/O operation on the file is performed, the subroutine will be 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 should execute 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 BRS 2 like any other file.

In order to implement BRS 41, it is necessary to keep track of which I/O subroutine is open. This information is kept in 6 bits of the PAC table. The contents of these 6 bits is transferred into the opcode field of the return address when an I/O subroutine is called, and is recovered from there when the BRS 41 is executed.

11. EXECUTIVE TREATMENT OF FILES

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

The first part of this chapter contains a description of the file naming system as it appears to the user, and continues with a description of the Executive tables that implement the various features.

A user may give his files arbitrary names containing any characters other than ' or /. The names of new disc files may be surrounded by /, and the names of new tapes files must be surrounded by '. When a file is created it's name must be enclosed within one or the other of these characters.

When a user types a file name not enclosed within slashes or quotes, he need only type enough characters of the name to determine it uniquely. 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 of the characters, line feed or carriage return, the name will be regarded as correct. Any other character will be regarded as an indication that the name was incorrect. This machinery is intended to make it more difficult for the user to destroy old files or create new ones inadvertently.

When a new slashed output file name is given to the system, a new entry in the file directory and a new index block on the disc are created for it. If the name is being given to an executive command, it will be assumed that the file is a sequential one.

It is possible for the user to reference files belonging to users other than himself if the file name contains at least one control character or an @. He does this by preceding the file name with the account number and user name enclosed in parentheses. Thus, to get at file /@PROGRAM/ belonging to user JONES, he might type

(A1JONES)/@PROGRAM/

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

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

"PROGRAM".

The previous paragraphs have described the behavior of the system's file naming logic when it is recognizing names typed in on a teletype. The BRSs that recognize file names are capable, however, of accepting them in many other ways. Essentially, they accept a string pointer to the portion of the name already known (which may be null) and file numbers for the input file to be used in obtaining the rest of the name, and the output file on which the name should be completed. In most cases the first or the second of these items will be irrelevant.

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 expect to get the file name from the teletype. If the name is known to the program, they may be replaced by BRS 48. These BRSs are used in the following way.

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

The normal return leaves a file directory pointer in A, and BRS 18 leaves the character typed after OLD FILE/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.

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)

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
```

A string pointer is a character address found by multiplying the word address by three and adding 0, 1 or 2. The string pointer in A points to the character before the beginning of the file name. The pointer in B points to the last character of the name.

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

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

The string pointers point to the file name to be looked up in the file directory. The normal return leaves a file directory pointer in A. All other registers are modified. If the file name cannot be located in the file directory, the BRS 48 takes the exception return, while the BRS 60 will attempt to place the new name in the file directory; if it is unable to do so because the file directory is full, it will take the exception return.

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

```
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.

The file directory consists of an SPS hash table together with a table of equal length, called the description table (DBT), which has a three-word entry corresponding to each three-word entry in the hash table. In addition, there is a string storage area for storing file names and a few words of miscellaneous information. The parameters of a file directory are shown in Figure 11, and the format of a single hash table entry and matching DBT entry are shown in Figure 12. Executive commands for examining the file directory and setting various bits are described in Chapter 12. In addition, a number of BRSs are provided which permit the user's program to affect the contents of the file directory.

The creation date of file is set to the current date 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. There are five file names built into the system. They are:

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 and have the obvious significance. 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 from the paper tape and deliver it as the type.

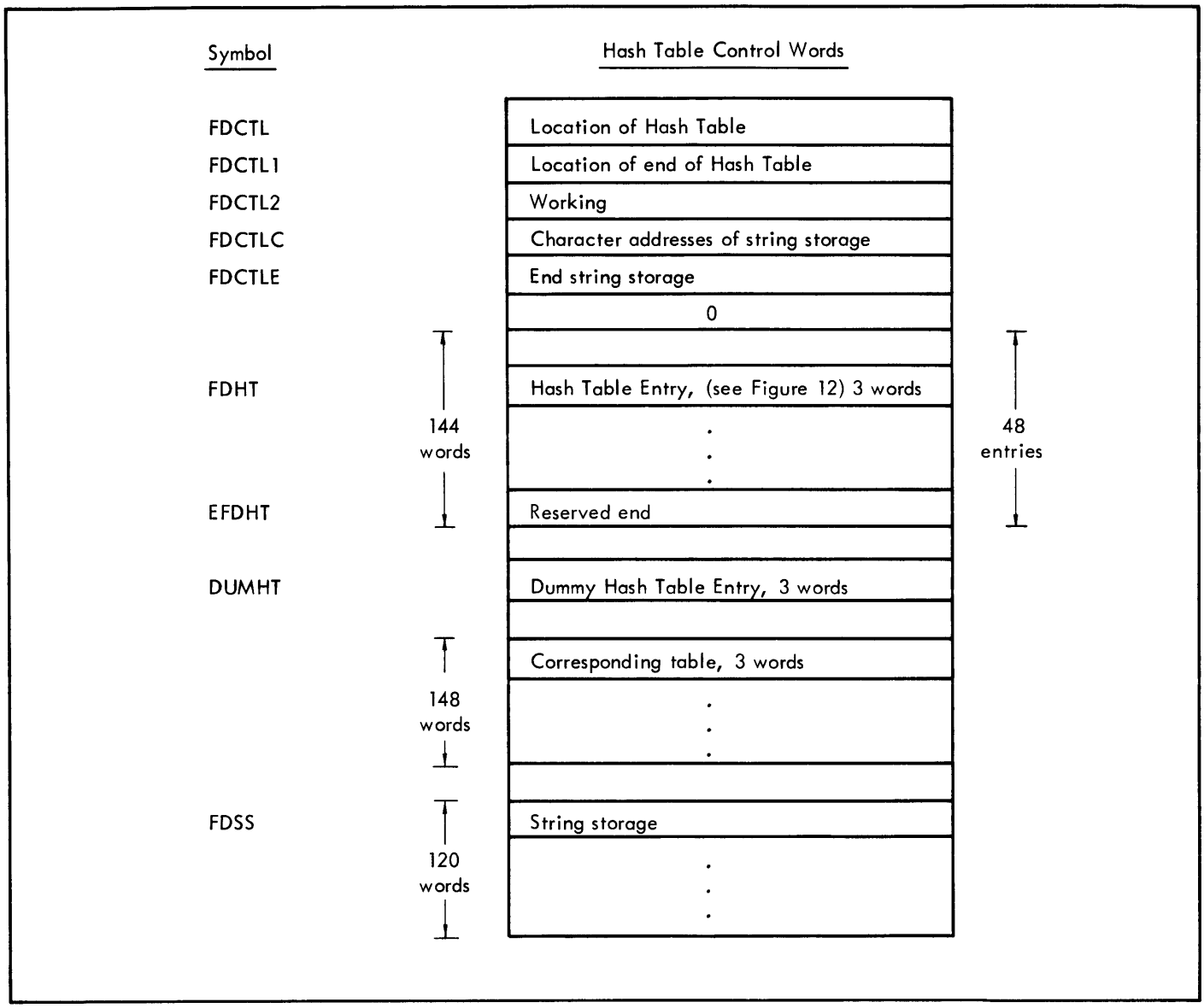
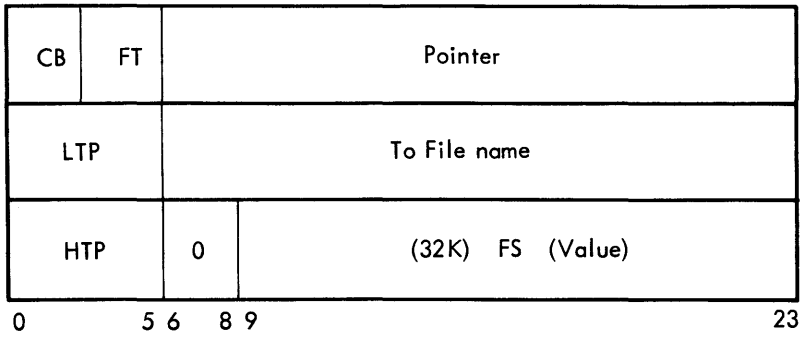
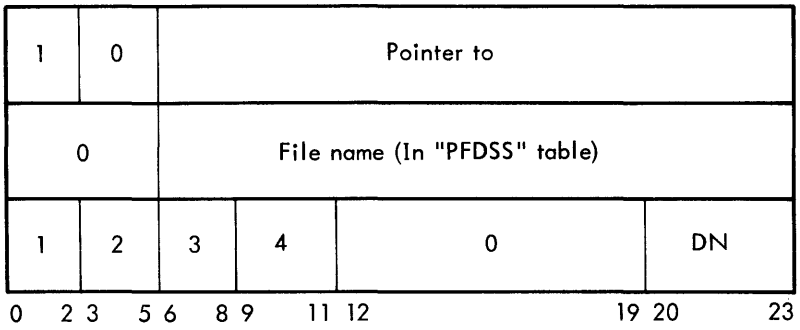


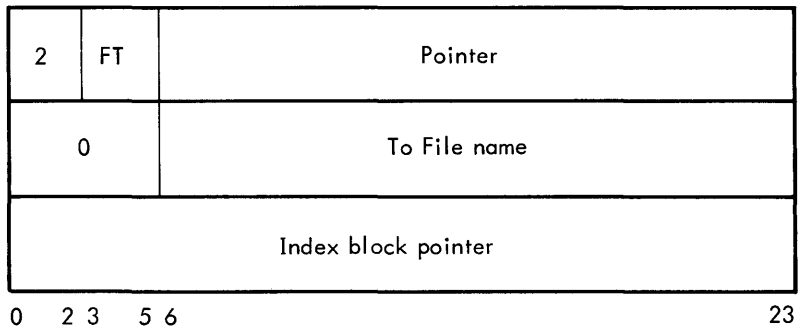
Figure 11. File Directory Arrangement



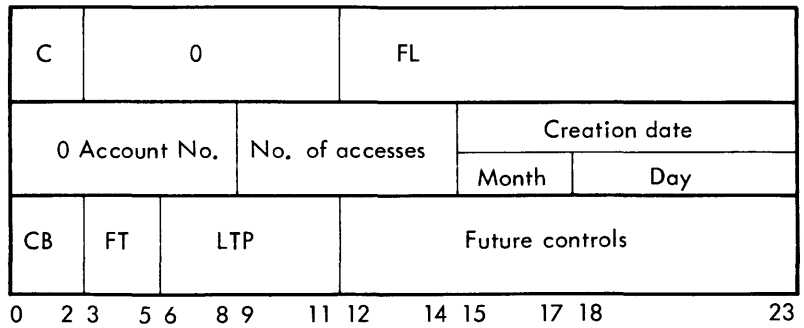
Physical Device



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

F = End of entry flag (1)

Figure 12. Hash Table Entry

12. EXECUTIVE COMMANDS RELATED TO FILES

When a user logs in to the system, his complete file directory is read in from the disc and placed in the file directory hash table along with the name of the physical devices. The "LOGIN" procedure is described in the SDS Terminal Users Guide, Publication No. 90 11 18A.

The following executive commands are related to the users 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 cause the complete directory to be typed while FD 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)	0
d	Blank	2
t	File type (1 through 4)	File type (1 through 4) (see Chapter 11)
s	File size	Index block pointer

A colon typed after either of the above commands, will cause the length (in numbers 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

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

0, 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

where the key to the parameters is the same as described above. 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 0, 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 Appendix A.

13. EXECUTIVE COMMANDS

The following commands are accepted by the Executive and are explained in detail in the SDS Terminal Users Guide, Publication No. 90 11 18A. The commands listed below would be applicable for all users.

LOGOUT	Allows user to logout
WRITE FD	Writes file directory on disc
RENAME	Renames a file
DATE	Types date and time
KILL PROGRAM	Kills program relabelling only
RESET	Clears all of user's memory
COPY	Copies file to file
FILES	Types file directory
FD FOR	Types selected file directory entry
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	Deletes a file
TIME	Types real time used
STATUS	Types user's relabelling status
MEMORY	Types user's unused memory
"	Causes typing to be ignored by EXEC
DUMP	Dumps all program, saves status
RECOVER	Recovers from a DUMP file (type 4)
CONTINUE	Returns to subsystem being used before the return to Exec
RELEASE	Releases a subsystem
EXIT	Allows a user to LOGOUT without writing file directory

The following commands are also accepted by the Executive but are for users with operator or system status only.

SHUT DOWN	Starts system shut down
UP	Cancels shut down
HANG UP	"Hangs up" selected teletype phone lines (DSS)
ANSWER	Answers (or enables) data subset
ACCOUNTING	Controls accounting to paper tape
LETTER	Types broadcast letters
ABT	Aborts tape operation (halts runaway)
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 available only for users with system status and are accepted by the Executive.

RSMT	Reads in from RAD a SMT Page
SYSDP	Allows core to be dumped directly on disc
SIZE	Sets user's machine size
MAIL	Types all mail in user's mail box
SEND TO	Allows user to put letter in mail box

The following commands are accepted by the Executive for users with system status or operator status.

USERS	Types number of users on system
WHERE IS	Gives teletype number for a user
WHO IS ON	Types users on system by account and name
REWIND	Rewinds tape, resets tape logic
RLT	Releases tape
STN	Sets tape number
PTN	Types tape number
SETEXEC	Sets user status
POSITION TAPE	Positions tape
TAPE POSITION	Types current tape position
DF	Allows a file directory entry to be set up
REMOVE FILE	Removes file from directory (without deleting)
PSP	Types error counters, etc.
CREATION	Types file directory with creation date and access count
LFCRE	Types creation date and access count of selected file
STORE	Stores a file on magnetic tape (in back-up format)
RETRIEVE	Retrieves a file from magnetic tape
DIRECTORY	Types file directory for files in backup format

14. OPERATOR EXECUTIVE ROUTINES

FUNCTION

The general function of the operator Executive 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.

DESCRIPTION OF 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 7.

Table 7. 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.

OPERATING PROCEDURES

This section describes the general operating instructions, program loading, assembly and a description of the operator executive routines given in Table 7.

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.

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 exec 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 240. Normally a program identifier is placed at location 237 so that the program is saved from 237 to the final address (as typed by DDT) with the starting address as 240.

DESCRIPTION OF OPERATOR EXECUTIVE ROUTINES

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 linefeed 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 file. 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.

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

EXAMPLES:

*ACCOUNT (RET)
B1XYZ (RET)
0 0 (RET)
*

where

"B1" is the account number,

"XYZ" is the password,

0 0 set the account parameters to zero.

CANCEL ACCOUNT

COMMAND: CANCEL ACCOUNT (RET)

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

Operating Instructions. 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.

EXAMPLE:

*CANCEL ACCOUNT (RET)
B1 (RET)
*

CANCEL NAME

COMMAND: CANCEL NAME (RET)

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

Operating Instructions. 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
*CANCEL NAME (RET)
B1JONES (RET)
NEW
INVALID USER
*
```

CLEAR FILE

COMMAND: CLEAR FILE (LF)

FUNCTION: Clears a selected file directory.

Operating Instructions. 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.
```

EXAMPLE:

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

CLEAR RECORDS

COMMAND: CLEAR RECORDS (RET)

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

Operating Instructions. 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.

Operating Instructions. 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.

EXAMPLES:

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

COUNT LETTER

COMMAND: COUNT LETTER (RET)

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

Operating Instructions. 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    975
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.

Operating Instructions:

1. Command followed by a carriage return

   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.

2. Command followed by a line feed

   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 with key to fields:

1.	116	7.	154
2.	0:03.41	8.	600000
3.	0:59	9.	21
4.	77777777	10.	23000000
5.	41	11.	25020
6.	65	12.	/NAME OF FILE/

Key Breakdown:

1. User number.
2. Hours, minutes, and 1/100 minutes of computer time.
3. Hours and minutes of real time used (since reset time).
4. Valid on-time where each bit represents an hour of the day. The left-most bit represents 00:00 to 01:00.
5. Account number (where 41 would be account D1).
6. Number of times the file was accessed since last disc re-ordering. Reaches a maximum of 77_8 and stays there.
7. Creation date , where high order 4 bits is month less one and low order 5 bits is day of month less one. Example of 154 is April 13.
8. Flag bits indicating file was written on.[t]
9. File size where each bit represents one data block of 255 words.
10. File type (23 means symbolic disc file).
11. Index block pointer.
12. Name of file (control characters are preceded by an & on the teletype or by a Δ on the printer).

EXAMPLES OF FILES

```
-GO  /OPER/

*FILES  Ⓛ


7/3  13:39
300
300  0:00.01  0:03  77777777
   314301  60000000  24000000    62250 /D/
   310301  60000000  22000000    15562 /BS/
   300233        17  23000000    12550 /MAP3/
   300233         6  21000000      501 /OMAP/
   305301  60000000  23000000    46547 /MAPC/
   302301  60000000  21000000    46571 /OMAPC/
   342301  60000000  23000000    25571 /SYM3/
   301155        13  23000000    75102 /DD1/
   305301  60000000  21000000    32156 /OSYMA/
   321301  60000000  23000000    74753 /MAPB/
   303301  60000000  21000000    41572 /OMAPB/
   340301  60000000  23000000    72250 /SYM2/
   320301  60000000  23000000    15147 /DDUMP2/
   313301  60000000  23000000    42073 /TCOP19/
   300155        62  22000000    75115 /DDT/
   301205         2  22000000    30625 /OBCRDUM/
   303160        10  23000000    67306 /DDUMP/
   314301  60000000  22000000    41563 /B/
   300171         6  23000000    73303 /CRDTP/


OVERFLOW:  2140

2
 2   0:00.00  0:00  77777777
   700107         2  23000000    34223 /PRIMES/
   700564         2  23000000    57645 /DATA LA/
   700570        36  23000000    67672 /A1/
   700545         6  23000000    33672 /PROPOST/
   700562         3  23000000    53666 /COMSW 2/
   700557         3  23000000    53657 /COMSW 1/
   700564         4  23000000    57647 /DATA SF/


7777


TOTAL:  0:00.01  0:03


END JOB
```

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

EXAMPLES OF FILES (Cont.)

```
-GO  /OPER/

*FILES  (RET)

OUTPUT TO:  TELETYPE

7/3  13:34
 1    0:00.00  0:00  77777777
      300251         14  21000000  50512 /@OP8/
      304301   60000000  22000000  52606 /A/
      311301   60000000  23000000  55701 /OP1/
      300243          2  21000000  47121 /@HEAD/
      300152          4  21000000  23613 /@SYSD/
      300235          3  23000000  50715 /DSWAP/
      300232         14  21000000   1071 /@OPB/
      300216         40  23000000  37205 /NOP1/
      300243          4  23000000  31061 /HEAD/
      300220          2  21000000  35603 /TIME/
      300264          3  21000000  46154 /CBI/
      300243          4  23000000  34652 /HEAD1/
      300220         46  23000000  54440 /COP1A/
      300174          2  21000000  23436 /@NT/
      300251          5  23000000  51073 /ZD/
      300233         50  23000000   5077 /OP1B/
      304301   60000000  21000000  66264 /OP-SDS/
      306301   60000000  21000000  35414 /@OP/
      300232         23  23000000  63064 /OP2B/

OVERFLOW:  2123

 2    0:00.00  0:00  77777777
      700107          2  23000000  34223 /PRIMES/
      700564          2  23000000  57645 /DATA LA/
      700570         36  23000000  67672 /A1/
      700545          6  23000000  33672 /PROPOST/
      700562          3  23000000  53666 /COMSW 2/
      700557          3  23000000  53657 /COMSW 1/
      700564          4  23000000  57647 /DATA SF/

 5    0:03.40  0:29  77777777
      300174          6  23000000  27362 /NMPL/
      304302   60000000  22000000  71140 /B/
      300173         32  24000000  73334 /DEBIT1/
      303302   60000000  23000000  65030 /GOM/
      300267         32  24000000  16767 /6-23/
      300171         32  24000000  54313 /DEBIT/
      303212         20  23000000  17362 /MOK/
      304302   60000000  21000000  55625 /@GO/
      300167          4  23000000  54303 /FT/
      300231         32  24000000  74423 /DEB 5-26/
      300171          5  23000000  44132 /F2/
      300267          6  23000000   5363 /S7/
      300231         32  24000000  72646 /5-26/
      300242          6  23000000  31046 /S6/
      300205          4  23000000  20341 /OMPLOT/

 6    0:00.20  0:03  77777777
      202274   60000000  23000000  75671 /INV2/
      201273
```

GARBAGE

COMMAND: GARBAGE (RET)

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

Operating Instructions. 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 (RET) 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 (RET)

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

LENGTH

COMMAND: LENGTH (RET)/(LF)

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

Operating Instructions:

1. Command 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 gone through, and the output file has been closed.

2. Command 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 ⏎

FUNCTION: Allows the operator to create a broadcast letter.

Operating Instructions. Before giving this command, the operator must first set the EXEC letter switch to OFF. This is done by giving the EXEC command LETTER ⏎. 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 ⏎	Giving the EXEC command to turn the letter switch off.
LETTER OFF	
-GO /OPER/ ⏎	Calling the OPER program
*LETTER ⏎	Giving the command
LETTER NO.: 2 ⏎	The program asks for a letter number
⏎ ⏎ ⏎	
TEXT OF LETTER ⏎⏎	The letter; maximum size is 189 characters
D^C	
*	
-LETTER 2 ⏎	EXEC command to type a letter
TEXT OF LETTER	EXEC types the letter
-LETTER ⏎	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 ⏎

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

Operating Instructions. 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 ⏎

B1JONES ⏎

OLD

60000023 ⏎

POINTER

COMMAND: POINTER ⏎

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

Operating Instructions. 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 pointer.

REMOVE LETTER

COMMAND: REMOVE LETTER (RET)

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.

Operating Instructions. 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 (RET)
LETTER NO.: 2 (RET)
*
_

RESET TIME

COMMAND: RESET TIME (RET)/(LF)

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.

Operating Instructions. 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 (RET)/(LF)

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

Operating Instructions

1. Command 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.

2. Command terminated by line feed

   The 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/ (RET)
*SET DAY (RET)
END OF JOB
_
-GO /OPER/ (RET)
*SET DAY (LF)
121 (RET)
23 (RET)
7777 (RET)
END OF JOB

SET HOUR

COMMAND: SET HOUR (RET)/(LF)

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

Operating Instructions

1. Command 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.

2. Command 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 parameter:

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 77777777 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 0:02.34 10.54 7777 17777 (RET) (Changes the valid hours from "1200 to 2359" to "1100 to 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.

Operating Instructions. 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.

EXAMPLE OF SIZE ACCOUNT

```
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
```

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.

Operating Instructions

1. Command 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.

2. Command 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.

EXAMPLE OF TIME

```
-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
```

UAD

COMMAND: UAD (RET)/(LF)

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

Operating Instructions

1. Command 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.

2. Command 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.

EXAMPLES OF USER OUTPUT

```
-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
```

USERS

COMMAND: USERS ⏎

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

Operating Instructions. 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.

15. SUBSYSTEMS

The Time-Sharing System software is organized into a monitor, a system executive, and a number of subsystems which perform specialized functions. Each of these subsystems is called by giving its name to the executive as a command. The result of this operation is to bring the subsystem off the RAD and to to transfer to its starting point. The system will thereafter remember the subsystem which is in use and will cccept the CONTINUE command as an instruction to re-enter the subsystem without any initialization. Thus, for example, the command

-DDT

would call the debugging subsystem. The line

-CONTINUE ⏎

DDT

would re-enter DDT without initializing. Most of the subsystems are permanently present in the shared memory table, and may be called on by a user program.

Subsystems presently available in the Time-Sharing System are:

TAP: Symbolic macro-assembler

DDT: Debugging system

QED: Symbolic text editor

FTC: FORTRAN II compiler

FOS: FORTRAN II loader and operating system

FORTRAN: FORTRAN IV system

CAL: Conversational Algebraic Language

BASIC: Conversational Algebraic Language

16. MISCELLANEOUS EXECUTIVE FEATURES

The Executive provides a number of BRSs that are services for the user. The BRSs all declare a fork to execute. This group of BRSs are run in user mode and are called class 3 BRSs in the Monitor.

To get the date and time into a string, the operations

```
LDP     PTR
BRS     91
```

may be executed. The current date and time are appended to the string provided in A and B and the resulting string is returned. The characters appended are

```
mm/dd   hh:mm
```

where

```
1st mm  = month     (2 digits)
dd      = day or month   (2 digits)
hh      = hour of day    (24 hour clock)
2nd mm  = minutes
```

Hours are counted from 0 to 23.

All other system Executive BRSs have been described in previous chapters.

17. MISCELLANEOUS MONITOR BRS'S

The Monitor provides a number of BRSs which are services for the user. Many of these are incorporated in the string processing system or in the floating point package and are described in the next two chapters. These are called class 2 BRSs in the Monitor.

To put an integer to any radix the instructions

```
LDB     =radix
LDX     =file
BRS     38
```

may be executed. The number that may be preceded by a plus or minus sign, is returned in the A register and the non-numeric character which terminated the number in the B register. The number is computed by multiplying the number obtained at each stage by the radix and adding the new digit. It is, therefore, unlikely that the return value will be correct if the number of digits is too large.

To output a number to arbitrary radix the instructions

```
LDB     =radix
LDX     =file
LDA     number
BRS     36
```

may be executed. The number will be output as an unsigned 24-bit integer. If the radix is less than 2, an error will be indicated.

18. STRING PROCESSING SYSTEM

A resident part of the system is a package of string handling routines. These are discussed in detail in the second half of this manual and will only be listed here.

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 is 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 SYSPOP's which are formally part of SPS but which are useful in floating point operations and in general programming are:

LDP	Load pointer
STP	Store pointer

These are double word 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, respectively.

19. FLOATING POINT

Floating point arithmetic and input-output operations have been incorporated into the 940 system through the use of programmed operators. This allows the user to perform useful arithmetic and I/O operations in a single instruction. A brief summary of the most commonly used arithmetic and I/O POPs is outlined in this chapter.

The floating point numbers referenced in this chapter are normalized double word values. The first word is a sign bit followed by the high order 23 bits of the mantissa bits followed by a 9-bit exponent field which, like the mantissa, is always represented in two's complement form.

Unless otherwise specified, the POPs do not make a skip return.

The remaining floating point SYSPOPs and BRSs use a format word in register X which contains the information shown in Figure 13.

Floating Point Load/Store Instructions

Example 1

NAME: LDP

FUNCTION: Load pointer

CALLING SEQUENCE: LDP Memory

DESCRIPTION: Loads A, B with MEMORY, MEMORY+1.

LDP is a single instruction that is equivalent to

```
LDA     MEMORY
LDB     MEMORY+1
```

NAME: STP

FUNCTION: Store pointer

CALLING SEQUENCE: STP MEMORY

DESCRIPTION: Replaces MEMORY, MEMORY+1 with the contents of A, B. STP MEMORY is a single instruction that is equivalent to

```
STA     MEMORY, STB     MEMORY+1
```

Example 2

NAME: FAD

FUNCTION: Floating add

CALLING SEQUENCE: FAD MEMORY

DESCRIPTION: The floating point value at MEMORY, MEMORY+1 is added to the floating point value in A,B. The sum replaces the value in A, B. Memory is unaffected.

NAME: FSB

FUNCTION: Floating subtract

CALLING SEQUENCE: MEMORY

DESCRIPTION: The floating point value at MEMORY, MEMORY+1 is subtracted from the floating point value in A, B. The difference replaces the value in A, B. Memory is unaffected.

NAME: FNA

FUNCTION: Floating negate

CALLING SEQUENCE: BRS 21

DESCRIPTION: The floating point value in A, B is negated. The result is left in A, B.

NAME: FMP

FUNCTION: Floating multiply

CALLING SEQUENCE: FMP MEMORY

DESCRIPTION: The floating point value at MEMORY, MEMORY+1 is multiplied by the floating point value in A, B. The product replaces the value in A, B. Memory is unaffected.

NAME: FDV

FUNCTION: Floating divide

CALLING SEQUENCE: FDV MEMORY

DESCRIPTION: The floating point value in A, B is divided by the floating point value at MEMORY, MEMORY+1. The quotient replaces the dividend in A, B. Memory is unaffected. Division by zero causes an overflow.

NAME: FIX

FUNCTION: Conversion from floating point to fixed point

CALLING SEQUENCE: BRS 50

DESCRIPTION: The floating point value in A, B is converted to fixed point. A is replaced by the integer part of the original value, the fractional part is left adjusted in B. If the integer is too large, the most significant bits are lost.

NAME: FLOAT

FUNCTION: Conversion from fixed point to floating point

CALLING SEQUENCE: BRS 51

DESCRIPTION: The integer in A is floated. The floating point result is left in A, B.

Bits	Field Name	Significance
0-2	T	Format types: 0 – Octal 1 – Integer 2 – E format with the number right justified in the specified field on output. 3 – F format with the number right justified in the specified field on output. 4 – J format with the number left justified in the specified field on output. 5 – F format with the number left justified in the specified field on output. 6 – Double precision format. Same as 2 on input. On output same as 2 except a D will be output for the exponent if bit 16 is 1. 7 – Free form (output left justified).
3-8	D	Number of digits following the decimal point.
9-14	W	Total field width. In J format that is the number of digits before the decimal point.
15	0	Overflow action. If the field width is too small on output and this bit is 1, the first character of the output field will be a star and characters to the right will be lost. If this bit is zero and overflow occurs, characters on the right will be lost.
16	E	If this bit is 1, E format of output will be used to represent the exponent. If this bit is 0 the @ symbol will be output. Either the E or @ is always acceptable on input.
18		If this bit is 0 on input the symbol @ will be treated as a legal exponent identifier; i.e., 1.0@+2 will be legal input. If this bit is 1 the symbol @ will be treated as an illegal character. This bit has no effect on output.
19		If this bit is 0, illegal characters in the input string will be ignored. The error flag will be set when one is read. If this bit is 1 and an illegal character is read, the scan will be terminated, the error flag will be set and the string pointer will be set to the character read. The conversion will take palce on the characters read to that point. This bit has no effect on output.
20		If this bit is 0, the input string ±N±M is legal. N is treated as the mantissa and M the exponent of a floating, real number. If this bit is 1, the second occurrence of a sign will be treated as an illegal character. This bit has no effect on output.
21		Must be zero.
22		Must be zero.
23		If a 1, the double precision accumulator will be used for numeric input-output. Significance is extended to 18+ digits. Applies to all format types.

Figure 13. Format Word for Floating Point

OPERATING CHARACTERISTICS

On input the D file is overridden by the presence of a decimal point. If a decimal point and/or E are present, any form of the number is acceptable to any input format. It is only in the absence of these characters that the format specifications determine the interpretation of the field. Illegal characters appearing anywhere in the field may be ignored depending on bit 19 of the format word. Blanks will be converted to zero.

The maximum allowable number of input digits is twelve. If more than twelve digits are input the most significant twelve will be used. If twelve digits are used, care must be taken as overflow can occur during the conversion process. Insignificant leading or trailing zeros will be ignored.

The maximum allowable integer on input is $\pm 2^{38} - 1$ or ±274,877,906,913. Floating point numbers must lie in the range:

$$9.9999999999E-78 \leq \text{number} \leq 5.7896044625E+76$$

Free form output will be output using an F17 if the exponent lies in the range $-1 \leq \text{exponent} \leq 10$ (X = 10 -number of digits to left of decimal point). If the number is outside this range an E17.11 will be used. Free form output always assumes a floating point number. If an integer is input it will be normalized prior to conversion.

For the E format on output, the E (@ if bit 16 of the format word is 0) is always followed by a + sign or - sign. On all output the sign of the number is printed only if it is negative.

Error Conditions:

If an error is detected during the conversion process a positive integer indicating the error type will be returned in the index register as given in Table 8.

Table 8. 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.

String Conversion

Example 3

NAME: SIC

FUNCTION: String to internal conversion

CALLING SEQUENCE:

```
LDX     FORMAT
SIC     POINTER
BRU     INTEGER
BRU     FLOATING
```

DESCRIPTION: FORMAT describes the type of conversion to be done (see the SDS 940 FORTRAN IV Manual, Pub. No. 90 11 15, for the FORMAT word specifications). The string of input characters starts at the character following the character pointed to by the character address in POINTER. The character address in POINTER+1 points to the last character of the input string.

NAME: ISC

FUNCTION: Internal to String Conversion

CALLING SEQUENCE:

```
LDP     VALUE
LDX     FORMAT
ISC     POINTER
```

DESCRIPTION: FORMAT describes the type of conversion to be done (see the SDS 940 FORTRAN IV Manual, Pub. No. 90 11 16 for the FORMAT word specifications). POINTER+1 contains the character address of the character immediately preceding the position where the first character of output is to go. POINTER+1 is incremented by one for each character of output added to the character string. VALUE is the double word floating point value to be converted.

NAME: FFI

FUNCTION: Formatted input

CALLING SEQUENCE:
```
LDX     FORMAT
BRS     52
```

DESCRIPTION: Characters are read for a file and converted to internal form. Either a floating point value is left in A, B or an integer is left in A. A skip return is generated if a floating point value is read and the input mode is free format.

NAME: FFO

FUNCTION: Formatted output

CALLING SEQUENCE:
```
LDP     VALUE
LDX     FORMAT
BRS     53
```

DESCRIPTION: The floating point value in A, B or the integer in A is output to the file specified in FORMAT.

20. BRS AND SYSPOP INDEXES

INDEX OF BRS'S AND SYSPOP'S BY NUMBER

*BRSs marked with an asterisk are executive BRSs, and all others are monitor BRSs.

*BRSs marked with an asterisk are executive BRSs, and all others are monitor BRSs.

SYSPOPs	Function	Page
BIO	Block input/output	82
CIO	Character input/output	81
CIT	Character input and test	105
CTRL	Input/output control	83
EXS	Execute instruction in system mode	126
FAD	Floating point addition	117
FDV	Floating point division	119
FMP	Floating point multiplication	118
FSB	Floating point subtract	118
GCD	Get character from end of string and decrement end pointer	111
GCI	Get character from beginning of string and increment beginning pointer	110
ISC	Internal to string conversion	115
IST	Input from specific TTY	94
LDP	Load string pointer	109
OST	Output to specific TTY	94
SKSE	Skip if string equal	109
SKSG	Skip if string greater	110
SIC	String to internal conversion	115
STI	Simulate TTY input	95
STP	Store string pointer	108
TCI	Teletype character input	93
TCO	Teletype character output	93
WCD	Put character on beginning of string and decrement beginning pointer	112
WCH	Write character to memory by table	112
WCI	Put character on end of string and increment end pointer	111
WIO	Word input/output	81

INDEX OF BRS'S AND SYSPOP'S BY TYPE

21. SCHEDULING, FORKS AND PROGRAM INTERACTION

Program Interrupts

BRSs or SYSPOPs	Function	Page
78	Arm/disarm software interrupts	52
79	Cause specified software interrupts	53
49	Determines which software interrupts are armed	53
BE+12	Arm timing interrupt	54

Control of the Escape Key

BRSs or SYSPOPs	Function	Page
90	Declare a fork for escape	54
46	Turn escape off	55
47	Turn escape on	55
26	Skip if escape waiting	56

Activation of Forks

BRSs or SYSPOPs	Function	Page
9	Open fork	57
57	Guarantee 16ms computing	58

Interrogation of a Fork

BRSs or SYSPOPs	Function	Page
30	Read status of a lower fork	58
107	Read status of all lower forks	59

Temporary Suspension of Forks

BRSs or SYSPOPs	Function	Page
45	Dismiss on quantum overflow	59
72	Executive dismissal	60
81	Dismiss for specified amount of time	60
31	Wait for specific fork to cause a panic	61
106	Wait for any fork to terminate	61
109	Dismiss calling fork	62

Termination of a Fork

BRSs or SYSPOPs	Function	Page
10	Terminates the calling fork	62
32	Terminates a specified lower fork	63
73	Terminates a specified number of lower forks	63
108	Terminate all lower forks	64

22. INPUT/OUTPUT

Direct Control of Peripherals

BRSs or SYSPOPs	Function	Page
1	Open a file of a specific device	65
110	Read device and unit	66
2	Close a file	66
20	Close a tape file	67
8	Close all files	67
66	Delete DSU file data	68
67	Delete DSU file index block	68
82	Switch sequential file type	69
87	Read DSU file index block	69
104	Read a page (2048 words) from RAD	70
105	Write a page (2048 words) to RAD	70
113	Compute file size of a disc file	71
114	Turn off run-away magnetic tape	71
118	Allocate magnetic tape unit	72
119	Deallocate magnetic tape unit	72
BE+9	Read DSU page	73
BE+10	Write DSU page	73
BE+7	Test a breakpoint switch	74
BE+1	Read DSU	74
BE+2	Write DSU	75

Control of Files Via File Names

BRSs or SYSPOPs	Function	Page
15	Read input file name	76
16	Open input file in file directory	77
17	Close all files	77
18	Read a file name and look it up in the file directory	78
19	Open output file located in file directory	79
48	Look up input/output file name	80
60	Look up I/O file name and insert in file directory if not found	80

I/O Operations

BRSs or SYSPOPs	Function	Page
CIO	Character input/output	81
WIO	Word input/output	81
BIO	Block input/output	82
CTRL	Input/output control (tape)	83

23. TELETYPE OPERATIONS

Linking and Attaching

Input/Output Operations

24. MEMORY OPERATIONS

Private Memory

Shared Memory

25. STRING PROCESS

String I/O

String I/O (cont.)

Hash Table Search

String Manipulation

Character Manipulation

26. NUMBER OPERATION

Number I/O

Arithmetic Operations

27. EXECUTIVE COMMAND OPERATIONS

28. MISCELLANEOUS OPERATIONS

BRSs or SYSPOPs	Function	Page
42	Read real-time clock	121
91	Read date and time into a string	121
88	Read execution time	122
41	Return from I/O subroutine	122
111	Return from exec BRS (exec only)	123
112	Turn off teletype station (exec only)	123
152	Execute instructions in system mode	126
71	Skip if executive	122
BE+5	Set disc bit map	124
BE+8	To crash the system for error diagnostic	125
BE+13	Sets exec switches in SYMS	125
EXS	Execute instruction in system mode	126

21. SCHEDULING, FORKS AND PROGRAM INTERACTION

NUMBER: 78

NAME: SAIR

FUNCTION: Arm/Disarm Software Interrupts

STATUS: User

CALLING SEQUENCE:
```
LDA     MD
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)

11 Interrupt if DSU error

5 through 10 interrupts on condition set by user

Location 200 octal plus the interrupt number must be set to point to a routine to process the interrupt. When the interrupt occurs an SBRM* is executed to the location pointed to. If it is desired to return to the point in the program interrupted, the user must BRR to the location where the return was saved.

Example:

```
      SET           INTERRUPT ROUTINE   RETURN
LDA = ESCAPE        ESCAPE ZRO ESCRTN   BRR ESC RTN
STA 201B            :
:                   :
```

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 5 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 5 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 and etc. There are 11 programmable interrupts. See also, BRS 78.

REGISTERS 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 is the new interrupt mask.
T is the time in milliseconds after which the fork will be interrupted.
N is the 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 also, BRS 81.

REGISTERS AFFECTED: None

NUMBER: 90

NAME: DFR

FUNCTION: Declare a Fork for "Escape"

STATUS: User

CALLING SEQUENCE: BRS 90

DESCRIPTION: In case the user types "Escape" or a fork panics, this fork will be activated. A fork panic is a fork status of 0, 1, or 2. See also, BRS 10.

REGISTERS AFFECTED: None

NUMBER: 46

NAME: NROUT

FUNCTION: Turn Escape Off

STATUS: System

CALLING SEQUENCE: BRS 46

DESCRIPTION: This BRS will set up to remember an escape interrupt, but not allow the program to be interrupted. It will stack the first escape occurring and ignore any subsequent ones.

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 interrupt was stacked (remembered) while in an Off condition, the interrupt will occur.

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 to the "exception return" if there is not an escape stacked. 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".
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.

5 = Make fork privileged if current fork is privileged.

DESCRIPTION: BRS 9 is used to create dependent entries in the PAC table. 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 16ms 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 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.

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 also, 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

BRS 10 terminates the fork that issues it and returns control to the higher fork. It is just like typing "escape" on the teletype.

DESCRIPTION: Terminates a lower fork. 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 only 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 system 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

22. INPUT/OUTPUT

NUMBER: 1

NAME: MONOPN

FUNCTION: Open a File of a Specific Device

STATUS: System

CALLING SEQUENCE:
```
LDA    ±I
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.
- = Make the file read only.
+ = Make the file read/write.
D = Device number.

Available device numbers are as follows:

1. Paper tape input.
2. Paper tape output.
3. Card input.
4. Magnetic tape input.
5. Magnetic tape output.
7. Printer.
8. Sequential DSU input.
9. Sequential DSU output.
10. Random DSU

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; otherwise control is transferred to the exception return. Exception conditions are as follows:

1. Device in use or not available.
2. File in use.

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 cannot be opened again until it is closed. 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 also BRSs 1, 8, and 82.

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 Mag 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 clobbered.

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 to the system 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. BRS 8 is always executed when control returns to the system. This BRS will not close magnetic tape files correctly. See also, BRS 1, 2, 82, 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.

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.

REGISTERS AFFECTED: None

NUMBER: 82

NAME: SWSF

FUNCTION: Switch Sequential File Type

STATUS: User

CALLING SEQUENCE:
```
LDA     N
LDB     C
BRS     82
```

N = File number
C = 0 will make the file an input file.
C = 1 will make the file an output file.

DESCRIPTION: This BRS sets the file type to input or output depending on the contents of register B regardless of its current file type and transfers control to the normal return.

RESTRICTION: If the sign bit of register A was set when the BRS 1 was executed to open the file, it cannot be switched from input to output. A violation results in an instruction trap.

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     DFCD
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: 114

NAME: MTDI

FUNCTION: Turn Off Run-away Magnetic Tape

STATUS: System

CALLING SEQUENCE:
```
BRS     MIDI
NORMAL RETURN
```

DESCRIPTION: Issues commands to try to stop the tape.

REGISTERS AFFECTED: None

NUMBER; 118

NAME: TGET

FUNCTION: Allocate Magnetic Tape Unit

STATUS: System

CALLING SEQUENCE:
```
LDA     =Tape Number
BRS     118
EXCEPTION RETURN
NORMAL RETURN
```

DESCRIPTION: Assigns tape requested to the user. If 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. Releases regardless of who had it.

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 104. Can only be called by the system exec. BE+1 can be used to perform this function.

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 105. Can only be called by the system exec. BE+2 should be used to perform this function.

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

CALLING SEQUENCE:
```
LDA     =Core Address
LDB     =Disc Address
LDX     =Number of Words
BRS     BE+1
NORMAL RETURN
```

DESCRIPTION: Reads up to 2K words from disc. Transfer must not cross a page boundary and must be in multiples of 64 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 BE+1. The number of words must be a multiple of 64 and greater than 0.

REGISTERS AFFECTED: None

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 the 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:	Pointer to the string pointers of the desired file in the file directory hash table.
	B:	The value word of the hash table entry.
	X:	Clobbered.

Note: The information contained in the registers cannot be used directly by the user since the addresses are in the T. S. Block; this BRS is normally follwed 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 = File Directory Pointer Address

DESCRIPTION: Opens an input file located in the user's file directory. The BRS 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 a physical device that cannot be an input file. The file directory pointer may be obtained from a BRS 15 or a BRS 18.

Exception Return:		All registers clobbered.
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 Mag Tape)

STATUS: User

CALLING SEQUENCE: BRS 17

DESCRIPTION: If mag 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 also, BRS 8. All registers are clobbered.

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 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 conditionally 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.

Exception Return:		All clobbered.
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:	Clobbered.

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 (as supplied in X by BRS 16) for tape files only.
N3 – File Type (as supplied in B by BRS 16).

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 DSU file and it cannot be opened, the message "NO ROOM" is typed. The message "FILE TYPE WRONG" is typed as appropriate.

Exception Return: All clobbered.
Normal Return: A – File Number.
B & X – clobbered.

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:	Clobbered.
Normal Return:	A & B:	Location in file directory hash table. Can be used by BRS 16 or BRS 19.
	X:	Clobbered.

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:	Clobbered.
Normal Return:	A & B:	String pointer to location in file directory hash table.
	X:	Clobbered.

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 134_8 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 137_8 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 134_8 or 137_8 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 not read into or out of 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 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

1. End of Record
2. End of File
3. Bad Record

If bit 1 is on 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 (only tape is implemented)

STATUS: System

CALLING SEQUENCE:
```
LDA     C
CTRL    N
```

C = Control number
N = File number

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 physical block.
4	Backspace file.
5	Erase tape (output only) (3 inches).
6	Rewind.
7	Write EOF. Output only.
8	Long erase. Output only.

REGISTERS AFFECTED: None

23. TELETYPES

NUMBER: 23

NAME: LNKS

FUNCTION: Link/Unlink TTY – Not implemented

STATUS: User

CALLING SEQUENCE:
```
LDX    T
LDA    A
LDB    C
BRS    23
```

T = Teletype number
A = Address of a list of teletype numbers terminated with -2.
C = Control word. The bits of this word are as follows:

Bit 0 = 0 = Output LCW, 1 = Input LCW.
Bit 1 = 0 = Clear all links first, 1 = Do not clear links first.
Bit 2 = 0 = Set link bits for TTY whose numbers are in the table.
Bit 2 = 1 , Clear link bits for TTY whose numbers are in the table.

DESCRIPTION: This BRS is used to set the link bit for TTY T in the LCW. Associated with each TTY are two words called the absolute input and absolute output link control words (LCW's). Each of these words contains one bit for each TTY in the system (maximum of 24). Also associated with each TTY are relative LCW's for input and output. The bits in these LCW's are set by this BRS. From the old relative LCW and the information supplied in the calling sequence a new relative LCW is created. Each time any relative LCW is changed, the absolute LCW's are all recomputed.

Link bits set in the input LCW cause input characters to be stored in the buffer of all TTY's linked to the controlling TTY. Link bits set in the output LCW cause output characters, including echoes, to be output to all TTY's linked to the controlling TTY.

REGISTERS AFFECTED: None

NUMBER: 24

NAME: LNKC

FUNCTION: Unlink – not implemented

STATUS: System

CALLING SEQUENCE:
```
LDX    T
BRS    24
```

T = Teletype Number

DESCRIPTION: This BRS is used to clear all links, input and output, to or from TTY T.

REGISTERS AFFECTED: None

NUMBER: 25

NAME: MSGS – not implemented

FUNCTION: Set Accept Messages and Set Accept Input Indicators.

STATUS: System

CALLING SEQUENCE:
```
LDX    T
LDA    I
BRS    25
```

T = Teletype number (must be controlling TTY or an attached TTY).
I = Bit 23 on to set "Accept Messages" Indicator.
I = Bit 24 on to set "Accept Input" Indicator.

DESCRIPTION: This BRS allows the user to specify whether or not messages from outside will be accepted, and whether or not input from outside will be accepted from his controlling teletype or for one which he has attached. The accept message indicator controls execution of OST instructions and the setting of teletype output links. The accept input indicator controls execution of STI instructions and the setting of teletype input links. Setting or clearing of these indicators will not affect any TTY links currently set.

REGISTERS AFFECTED: None

NUMBER: 27

NAME: ASTT – not implemented

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 TTY to the requesting program. If the indicated TTY 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

NUMBER: 28

NAME: RSTT – not implemented

FUNCTION: Release TTY

STATUS: System

CALLING SEQUENCE:
```
LDA     T
BRS     28
```

T = Teletype Number

DESCRIPTION: Returns to a free status the TTY indicated by the A register. If the TTY 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

NUMBER: BE+3

NAME: CARRY

FUNCTION: Test for Carrier Presence

STATUS: System

CALLING SEQUENCE:
```
LDA     =LINE #
BRS     BE+3
EXCEPTION RETURN – No Carrier
NORMAL RETURN – Carrier Present
```

DESCRIPTION: This BRS gives a skip return, if the carrier signal is present on the line identical in A. No carrier signal – no skip.

REGISTERS AFFECTED: None

NUMBER: BE+6

NAME: TTYON

FUNCTION: Turns a Teletype Line On or Off.

STATUS: System

CALLING SEQUENCE:
```
LDA     =TTY #
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: 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 TTY input buffer.

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 TTY output buffer.

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 = ± 1, 2, or 3 to indicate the proper echo table. If the sign bit of R is set, each 8 bit character read from the teletype is transmitted unchanged to the user's program. No echoes are generated while in this special 8-level mode. Teletype output is not affected.

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.

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.

REGISTERS 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 TTY).

DESCRIPTION: Dismiss this fork until the teletype output buffer indicated is empty. It is dismissed only until the last character is transmitted. This fork might be restarted before the last character interrupt has occurred, therefore, caution should be exercised.

REGISTERS AFFECTED: None

NUMBER: 85

NAME: SET8P

FUNCTION: Set Special Teletype Output

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     85
```

T = Teletype number (-1 is used to indicate controlling TTY).

DESCRIPTION: Sets teletype to 8-level output mode. The teletype specified must either be the controlling teletype or an attached teletype. 8-level is transmitted to the teletype exactly as it is received from the user program.

REGISTERS AFFECTED: None

NUMBER: 86

NAME: CLR8P

FUNCTION: Clear Special Teletype Output

STATUS: User

CALLING SEQUENCE:
```
LDX     T
BRS     86
```

T = Teletype number (-1 is used to indicate controlling TTY).

DESCRIPTION: Puts the teletype output back into normal mode. The teletype specified must either be the controlling teletype or attached.

REGISTERS AFFECTED: None

NUMBER: BE+11

NAME: CRSW

FUNCTION: To Allow the User to Ignore Line Feed or Carriage Return when it Follows a Carriage Return or Line Feed.

STATUS: User

CALLING SEQUENCE:
```
LDA     =0 (ignore) = -1 (do not ignore)
BRS     BE+11
NORMAL RETURN
```

DESCRIPTION: The contents of the A register will give the following results. If A is negative, all line feeds and carriage returns received from the TTY will be sent to the program and echoed. If A is positive, a line feed after a carriage return received from the TTY will be ignored (not sent to the program and not echoed) and a carriage return after a line feed will also be ignored (not sent to the program and not echoed). In all cases the first line feed or carriage return received will be sent to the program and echoed plus echo its complement.

REGISTERS 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 the location M right justified. The remainder of location M is cleared. The character is also placed in the A register right justified.

REGISTERS 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 135_8 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: IST – not implemented

FUNCTION: Input from Specified Teletype

STATUS: User

CALLING SEQUENCE: IST T

T = Teletype number

DESCRIPTION: IST is used to input a character from an attached teletype. The character will be right justified in the A register upon return.

REGISTERS AFFECTED: None

NAME: OST – not implemented

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

NAME: STI

FUNCTION: Simulate Teletype Input

STATUS: User

CALLING SEQUENCE: STI T

T = Teletype number

DESCRIPTION: This BRS is used to simulate teletype input. It puts the character in the A register into the input buffer of the specified teletype. It is legal for a user fork only if T equals the controlling TTY or -1.

REGISTERS AFFECTED: None

24. 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 in any other fork which has this PMT byte in its relabeling, the byte is cleared to 0.

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 exactly like 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 EXEC.

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 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 like it came from the 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.

REGISTERS 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.

REGISTERS 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.

REGISTERS AFFECTED: A

NUMBER: 68

NAME: EBSM

FUNCTION: Enter Block in SMT — Not implemented.

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.

REGISTERS AFFECTED: A

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.

REGISTERS AFFECTED: A

25. 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 on = 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.

Exception Return: A – The next character in the input buffer
B & X – No change

Normal Return: A – The character supplied remains in A (the character is removed from the input buffer).

REGISTERS AFFECTED: A

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 of the form:

```
ZRO  Hash Table Beginning Address
ZRO  Hash Table End Address
ZRO  0
```

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, and in register A, the string "value" 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 is pointed to by register X. (Word 3 will be -1 if the table is full.) The "value" is the hash image for the string.

See BRS 6

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 none is found but 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 the former case, more characters are taken from input until an exact match is obtained or no match is possible; in this latter case, the match is still valid, and the last character (which caused the mis-match) is left in the input file.

Exits are as follows: (1) The exception return is taken of the no-match condition with a string pointer in A, B 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 string "value" in B. X is undisturbed.

The "value" is the hash image for the string.

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: STP

FUNCTION: Store Pointers

STATUS: User

CALLING SEQUENCE: STP A

A = Address of a string pointer pair.

DESCRIPTION: This SYSPOP is generally used in conjunction with LDP. It stores the contents of the A and B registers into the string pointers indicated in the calling sequence.

REGISTERS AFFECTED: None

NAME: LDP

FUNCTION: Load Pointers

STATUS: User

CALLING SEQUENCE: LDP A

A = Address of a string pointer pair.

DESCRIPTION: This SYSPOP loads the string pointers indicated in the calling sequence into the A & B registers.

REGISTERS AFFECTED: None

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

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.

REGISTERS 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.

REGISTERS 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.

REGISTERS 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

26. 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.

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.

REGISTERS AFFECTED: None

NAME: SIC

FUNCTION: String to Internal Conversion

STATUS: User

CALLING SEQUENCE:
```
LDX     FORMAT
SIC     POINTER
BRU     INTEGER
BRU     FLOATING
```

DESCRIPTION: See String Processing System documents. FORMAT describes the type of conversion to be done.

The contents of POINTER point to the character immediately preceding the character string. POINTER+1 contains the character address of the last character of the string.

INTEGER and FLOATING are routines that handle the converted input. Error flags, if applicable, are in the index register A, double word value corresponding to the string is in A, B upon return.

REGISTERS AFFECTED: A, B, X

NAME: ISC

FUNCTION: Converts Internal Numbers to Formatted Output Strings

STATUS: User

CALLING SEQUENCE:
```
LDP     M
LDX     FORMAT
ISC     POINTER
```

DESCRIPTION: See String Processing Documents. FORMAT describes the type of conversion to be done. The contents of POINTER point to the character immediately preceding the character string. POINTER+1 contains the character address of the character immediately preceding the position where the first character of output is to go. M, M+1 contain the floating point word to be converted. Pointer+1 is incremented once for each character added to the string.

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 double word floating point value in the A and B registers is negated.

REGISTERS AFFECTED: A, B

NAME: FAD

FUNCTION: Floating Point addition

STATUS: User

CALLING SEQUENCE: FAD N

DESCRIPTION: SYSPOP FAD (A, B) + (M, M+1)
A floating addition is performed to the contents of memory location M and M+1 and the A and B registers. The results are left in the A and B registers.

REGISTERS AFFECTED: A, B

NAME: FSB

FUNCTION: Floating Point Subtraction

STATUS: User

CALLING SEQUENCE: FSB N

DESCRIPTION: (A, B) – (M, M+1)
The contents of memory locations M and M+1 are subtracted (floating subtraction) from the contents of the A and B registers. The results are left in the A and B registers.

REGISTERS AFFECTED: A, B

NAME: FMP

FUNCTION: Floating Point Multiplication

STATUS: User

CALLING SEQUENCE: FMP M

DESCRIPTION: (A, B) * (M, M+1)
The contents of memory locations M and M+1 are multiplied (floating multiplication) by the A and B registers and the results left in the A and B registers.

REGISTERS AFFECTED: A, B

NAME: FDV

FUNCTION: Floating Point Divide

STATUS: User

CALLING SEQUENCE: FDV M

DESCRIPTION: (A, B)/(M, M+1)
The contents of the A and B registers are divided (floating divide) by the contents of memory locations M and M+1 with the quotient left in the A and B registers.

REGISTERS AFFECTED: A, B

27. 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

28. 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 for the job in A.

REGISTERS 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.

REGISTERS 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: LDA 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.

REGISTERS 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.

REGISTERS 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.

REGISTERS AFFECTED: None

NUMBER: BE+13

NAME: SETSW

FUNCTION: Sets System Exec Switches in SYMS

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.

REGISTERS AFFECTED: A

NUMBER: 152

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

APPENDIX A. GLOSSARY OF TERMS

B

breakpoint switch: Refers to the four toggle switches physically located on the computer console.

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.

customer file directory: The names of all files for a particular user are recorded in this directory.

D

device table:

Device	Number
Paper Tape Input	1
Paper Tape Output	2
Magnetic Tape Input	4
Magnetic Tape Output	5
Hollerith Card Output	6
Binary Card Output	7
High Speed Printer Output	11
Hollerith Card Input	12
Binary Card Input	13

DSU block: Four consecutive sectors on the disc whose beginning addresses are MOD 4. A block consists of 256 words.

DSU data block: A DSU block with pointers in the first and second words. The first word points to the first relevant data word. The second word points to the last relevant data word.

DSU file: A file stored on the disc storage unit. Each file consists of at least an index block, and if the file contains data, then a sufficient number of DSU blocks to record the data.

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.

fork: A fork is all or part of a program. A program may consist of many forks and these forks may be in a heirarchy one to another. Forks are different from subroutines in that all forks making up a program could be theoretically executing simultaneously. At least one fork is associated with each active user in the system.

fork states:

-2 Dismissed for I/O.
-1 Running.
0 Dismissed on escape key or programmed panic
1 Dismissed on illegal instruction panic.
2 Dismissed on memory panic.

I

index block: A DSU block (256 words) which contains the DSU addresses for all data blocks of a file. Words 0 through 120 contain a DSU address which is MOD 4 in bits 6 to 23. Bits 0 and 5 of these words are unused. Bit 2 indicates an end of record data block. Words 121 and 122 are link pointers, and 123 is a hash total. Words 124 through 130 contain the file name, and word 131 contains user numbers.

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).

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 break 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

panic table (cont.):

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.

Q

quantum, long time: The maximum length of time a fork can run before the schedule 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

Relabeling, pseudo: See relabeling registers.

relabeling registers: The relabeling registers are used to indicate a page number which has been assigned to a user 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

S

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.

string, null: A pair of string pointers whose character addresses are the same.

APPENDIX B. 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 -2 indicates that there are additional entries in the overflow block.

The last four words of the file directory block contain the following information:

Last word	Valid on-time for this user (1 bit per hour of the day).
Last word -1	Accumulated computer time used.
Last word -2	Accumulated real-time used.
Last word -3	Overflow block pointer

In the case of an overflow block, the last three words are zero, and the overflow block pointer is a backward pointer to the first file directory block.

FT = File type

LTP = Low order tape position

HTP = High order tape position

FS = Tape file size

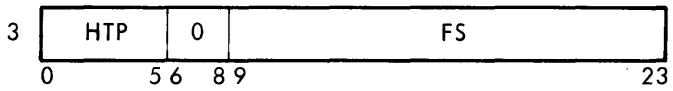
FL = File length for disc files

C = Change in file length (file length no longer valid)

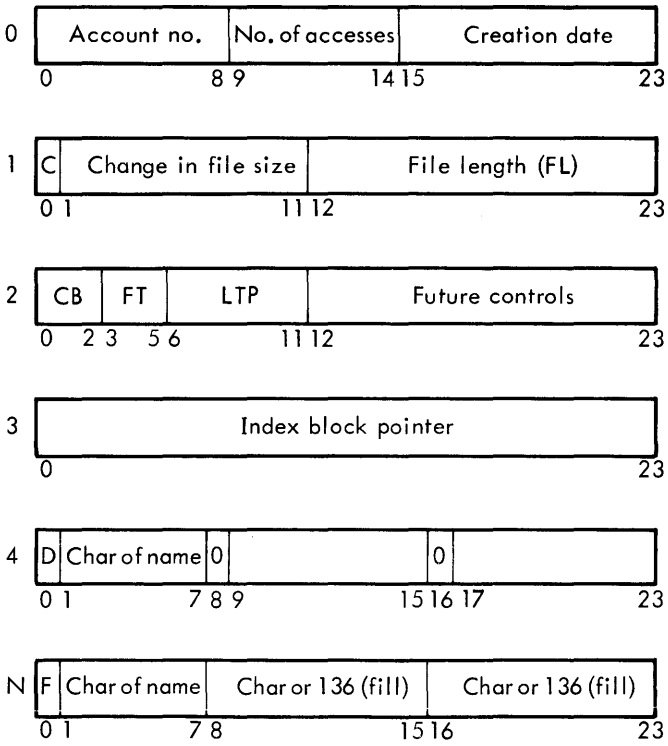
CB = File control bits, 0 = Tape file
2 = Disc file

F = End of Entry Flag (1)

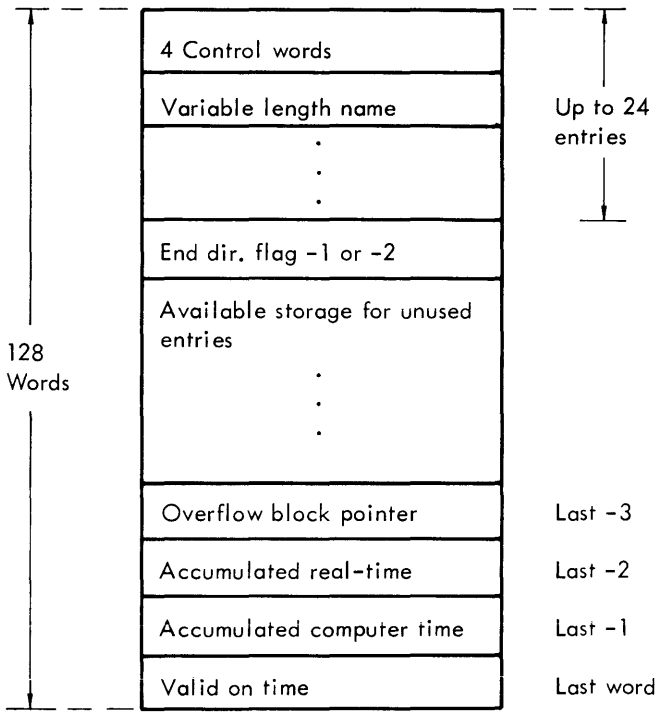
If Tape File, word #3 =



FILE DIRECTORY FORMAT ON DISC, 1 ENTRY (DISC FILE)



FILE DIRECTORY BLOCK



USER ACCOUNT DIRECTORY ON DISC

Words	0	1	2	3	4	5	6	7
	Acct. password				na	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

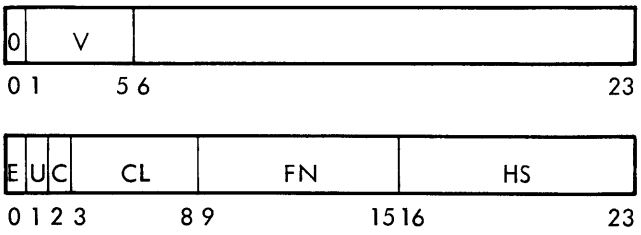
na	is not assigned
C	is a control parameter
N	is a user number
p	is reserved for an overflow pointer and not presently used.

The control parameter bits are assigned as follows:

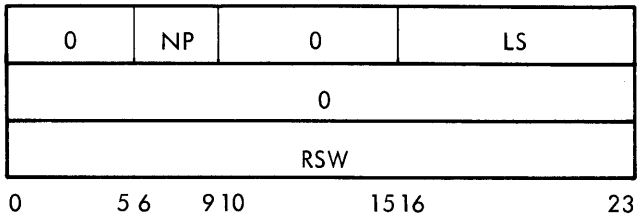
Bit	Use
0	System status
1	Control
2	Operator status
3	Subsystem status
4,5	Not assigned
6,11	Subsystem classes

SUBSYSTEM TABLE

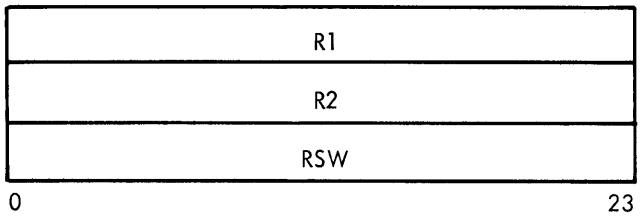
HASH TABLE ENTRY



CORRESPONDING TABLE (NOT COMMON SUBSYSTEM)



CORRESPONDING TABLE (COMMON SUBSYSTEM)



V	=	Subsystem verify number
LS	=	Low-order starting address
E	=	Propogate exec status
U	=	Co-exist with users memory (cannot if on)
C	=	Common subsystem
CL	=	Class (must agree with user's control parameters)
FN	=	File number (location on RAD for noncommon subsystem)
HS	=	High-order starting address
NP	=	Number of pages for noncommon subsystem
R1	=	First-half SMT relabeling (4 bytes)
R2	=	Second-half SMT relabeling (4 bytes)
RSW	=	Relabeling status word (8 bytes)