FST-1

DISC OPERATING SYSTEM

(DOPSY)

MANUAL

December, 1970

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

FIGURES

FST-1 DOPSY MANUAL

1.1 INTRODUCTION

The DOPSY programming system consists of a collection of programs operating under control of the Monitor. This is illustrated in the following diagram.

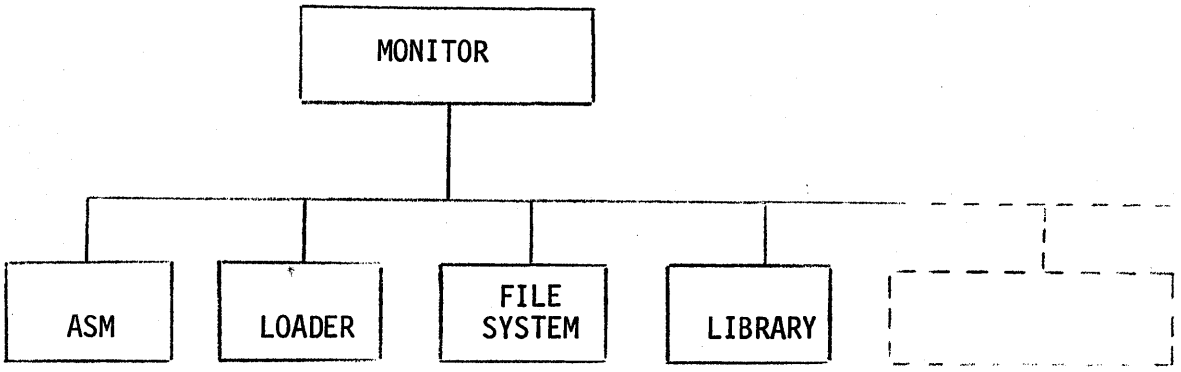


Figure 1

The broken lines indicate that this set of programs is not closed. As new translators become available, for example, they can easily be added to the system.

These related, but independent programs form a simple on-line, batched job processor. This manual attempts to provide enough material for a person to use the system commands effectively. The two major areas discussed are: 1) the details of systems organization that affect or are related to the user's program and 2) the commands by which a user communicates with the system. As a user becomes more familiar with the system and wants to utilize more of its resources he may wish to consult other manuals in the library of manuals for this computer. (See References at the end of this manual).

The minimum configuration that the system will run on is:

- 8K main memory (Note: 12K memory is minimum for running TOPSY--Tester OPerating SYstem).
- Disc file.
- Teletype with paper tape reader and punch.

GENERAL SYSTEM DESCRIPTION

2.1 INTRODUCTION

This section discusses in general terms the organization of DOPSY. This information is included so the user may better understand how his program interacts with the system.

2.2 INFORMATION FLOW

Figure 2.2 is a block diagram of a typical system. Input and output to the system are directed to a very large degree by the user explicitly selecting particular I/O devices when he issues a command to the system. For this purpose he may use the symbolic references TTP, TTK, CR, and LP. In many cases, failure to select a particular device will result in the system using the device assigned to PID or POD, Primary Input Device and Primary Output Device, respectively. In other cases, however, omission of the I/O device specification indicates to the system that working storage is to be used for the operation. This use of the PID or POD devices as default I/O devices is of secondary importance, however. The primary purposes of these devices are 1) a source of system commands, and 2) a receiver of standard systems output. These points are discussed in more detail throughout the manual.

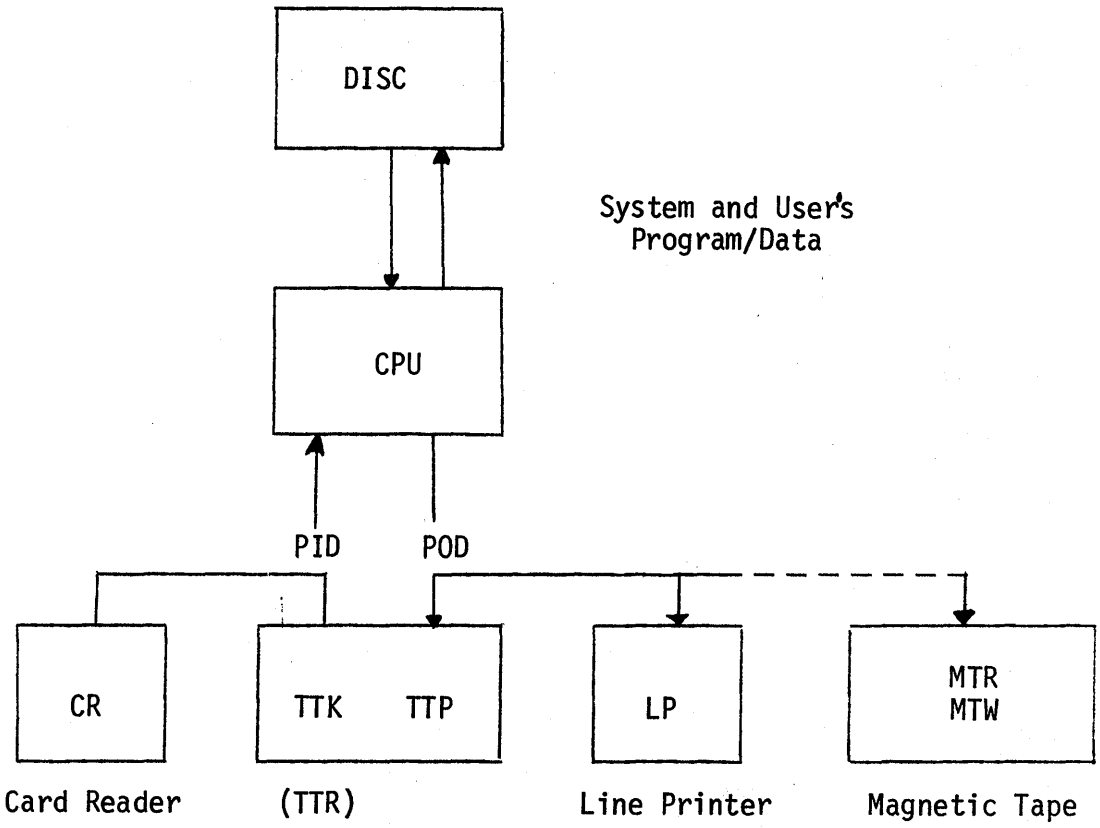
Note that the teletype is considered to be three devices; paper tape and keyboard input are differentiated, but not paper tape and printer output. The disc holds the programs and data required by DOPSY and also acts as a storage device for user programs.

2.3 DISC ALLOCATION

Figure 2.3 illustrates how space is allocated on the disc. Most of the facilities used by DOPSY for maintaining the disc and file processing are available to users.

Of the five areas on the disc only the first two, the core buffer (CB) and the skeleton monitor, are fixed in size. The size of the file directory can change only by issuing the appropriate ASSIGN command. The size of the file area and WS, however, are continually changing as a result of the ASSIGN, DELETE and CREATE commands and any increase in the size of one results in a decrease in the size of the other.

The kinds of files that the system processes are called STRING, DATA, OBJECT and COREIMAGE. The records that comprise these files are characterized by record length, fixed or varying, and record contents, character or word.



TeleType Keyboard
TeleType Paper Tape Reader
TeleType Printer

Figure 2.2

Information Flow

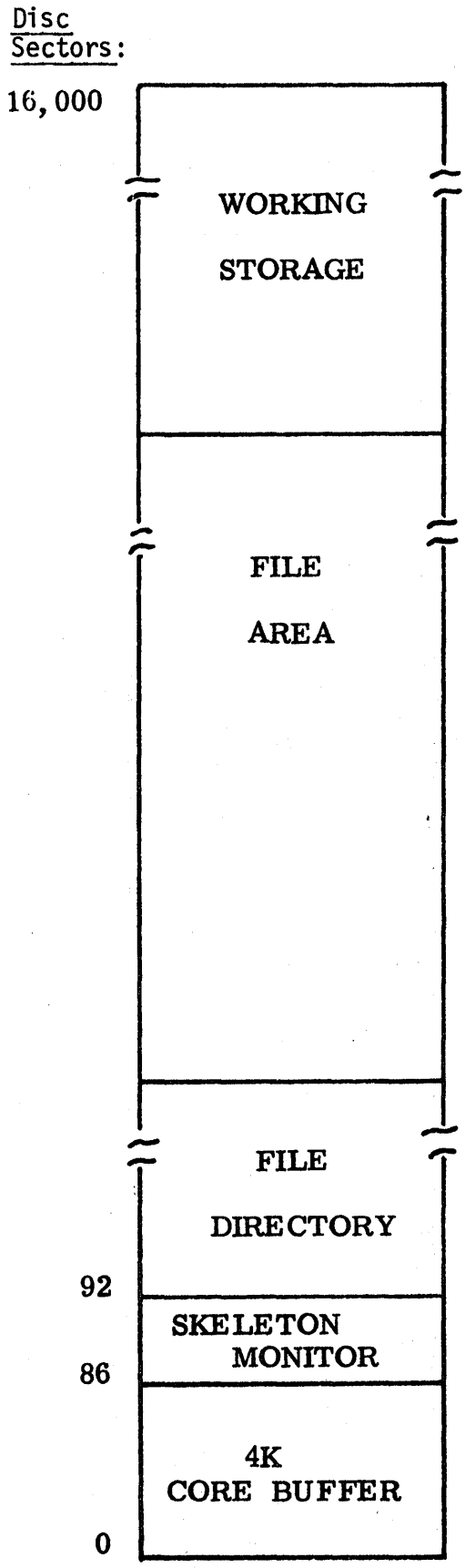


Figure 2.3
DISC ALLOCATION

STRING files have variable length, character records. OBJECT records are also variable in length, but are word-oriented; the length of these records is determined from the first word in the record. DATA records are word-oriented and fixed length; the record size, however, is declared at the time the file is created and can never change. COREIMAGE files have a very limited usage with regard to I/O and are treated as type DATA for this purpose.

The word-oriented files do not have any record separators. The first word of a record follows immediately after the last word of the previous record. STRING files, however, have each record terminated with a 77B character, or '←'. Records of STRING files should not, therefore, contain the '←' character or the record will appear as two or more records to the system's input routine.

2.4 MEMORY ALLOCATION

Figure 2.4 illustrates the allocation of the main memory when the monitor has control and when a user's program has control. User programs are not allowed to load below 220B and they should not use locations below 220B as data storage. No other restrictions are placed on such programs regarding utilization of main memory.

2.4.1 Skeleton Monitor

Locations 100B through 507B are occupied by the SKELETON MONITOR. This body of code consists of three pieces: the COMREC, Automatic Restart Routine and the Core-Image Loader. The COMREC and ARR occupy locations 100B through 217B and should not be altered except for the following case.

The variable M1RSTRT, location 120B, in the COMREC may be changed by the user to cause DOPSY to accept a different set of command records. This is discussed in more detail in 2.5.

ARR is used to automatically reload DOPSY at the completion of a user's program execution. It first saves the lower 4K of main memory in CB and then loads the core-image loader CIEXEC. CIEXEC, in turn, is used to load the main monitor.

The ARR can obtain control in either of the following ways:

- Executing a BRU 125B in any program.
- Executing a BRU 125B from the CPU console.

The core-image loader is used by DOPSY for loading overlays and can also be used by the user providing his program loads above 511B.

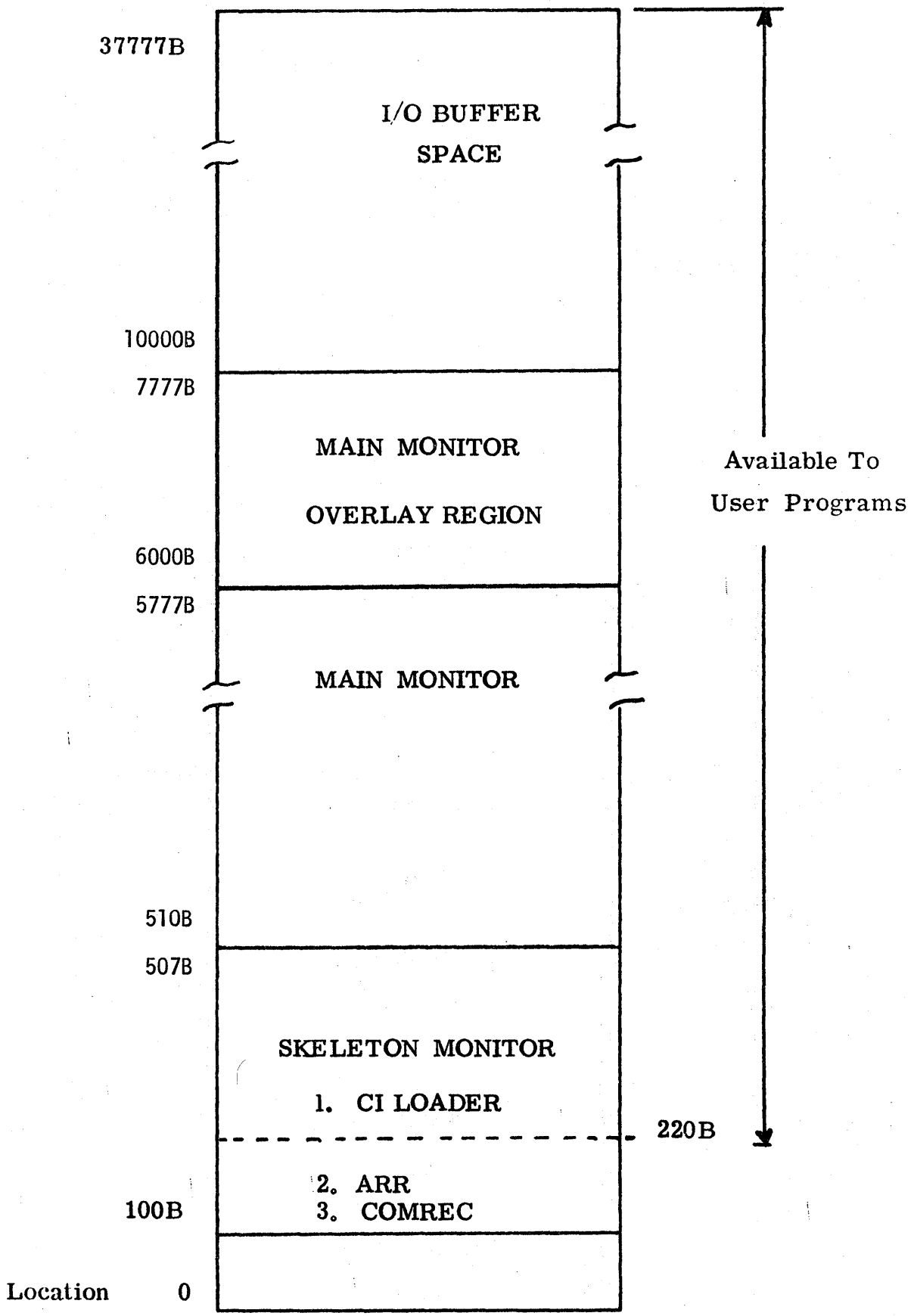

37777B
I/O BUFFER
SPACE
10000B
7777B
MAIN MONITOR
OVERLAY REGION
6000B
5777B
MAIN MONITOR
510B
507B
SKELETON MONITOR
1. CI LOADER
220B
2. ARR
3. COMREC
100B
Location 0
Available To
User Programs

Figure 2.4

The coding required is:

```
LDA  = 'SYMB'
LDE  = 'OL'
BSM*   510B
---            ERROR RETURN
---            NORMAL RETURN
```

The A and E registers must be loaded with the six-character name of the core-image program to be loaded. The A register should never be zero and the low order twelve bits of the E register should always be zero. The error return is taken if the file entry is not in the directory or if its type is not core-image. The normal return is taken only if the overlay executes a BRU* to its entry point.

CIEXEC uses and does not restore index registers 7 and 6 and state switch 7. It also loads into index register 0 the address plus one of the last location loaded.

The ARR facilities are used to load object as well as core-image programs and the user should note that the disc I/O that resides in ARR executes a BAH 100B if it is unable to read the disc after 80 retries. This halt is also executed if ARR is unable to reload the MONITOR because its directory entry cannot be located. In either case, the contents of the program counter will be 101B when the halt occurs.

2.5 COMMAND RECORDS

2.5.1 Introduction

The commands recognized by the monitor may be provided from the card reader, magnetic tape, or the teletype. They are received as input via the I/O procedures CRIO, MTIO, and TTRIO. When entering commands from the keyboard, one should be cognizant of the limited editing facilities provided by TTRIO. TTRIO interprets the character produced by pressing CTRL B as a backspace of a single character and that produced by CTRL L as a line delete. The former is indicated by echoing a '←' for each occurence while the latter is indicated by a carriage return, line feed and '*'.

Non-commands read from the card reader are simply bypassed, but a non-command record read from the teletype keyboard will produce the message '// RECORD NEEDED'.

2.5.2 Command Format

The format of the commands that the system recognizes is:

```
COL.  1  2  3  4 ____________________________________________ 80

      /  /  x  name                    <operand parameters>
```

Columns one and two must contain '//'. The x in column three is a restart character. Not all command records received by the monitor are accepted for processing. A command is accepted for processing only if column three is blank or equal to the last restart class given to the system.

Every time DOPSY is reloaded by ARR a particular class code is assigned to the restart. This class code is obtained from M1RSTRT, location 120B in the COMREC, and will have the value '@', 40B, unless the user's program changes M1RSTRT. This class code is then saved in the COMREC entry M1LRST and is used by DOPSY in determining which command records will be processed and which will be ignored. In general, cards that have a blank or an '@' in column 3 will be accepted if M1RSTRT is not changed by the user.

As an example of using the restart class consider a two phase program. The first phase reads data and does preliminary editing on the data. The resulting data is left in working storage for subsequent processing by phase two, which should be executed only if phase one is successful. This success is indicated by having phase one alter M1RSTRT to give a restart of '!', or 01. By using the following set of commands, phase two will execute only if phase one terminates successfully.

```
// EXEC 'PHASE1' CLEAR
        •
        •
   DATA USED BY PHASE1
        •
        •
// (eof record)
// @ DUMP 6000B to 7000B
// ! EXEC 'PHASE2' CLEAR
```

Only one of the last two commands is accepted. Failure of PHASE1 to alter M1RSTRT causes the DUMP command to be accepted and the second, EXEC, ignored, but the reverse will happen if PHASE1 is able to alter M1RSTRT to '!'.

The remainder of the command record contains the command name followed by the operand parameters. The command name may start anywhere after column three and terminates with the first special character, anything other than a letter or digit, which is ignored.

The operand parameters occur after the command name and must be separated from each other by a special character such as space or comma.

2.5.3 Operand Parameter Types

The types of parameters that may occur in an operand are:

- string — A sequence of characters, other than a single quote, enclosed in single quotes. These are used to reference files, job numbers, etc. Only the first six characters are retained.

Example: 'FILE.1'
'CER*'
'ABIGSTRING'

- name — A sequence of letters, including $ or digits; the first character must be a letter. Like strings, only the first six are retained.

  Example: TTK
  CREATE
  TEST34

- integer — A sequence of digits, which if terminated by the letter B is assumed to be octal, otherwise, decimal. Only the low order 24 bits are retained.

  Example: 15
  40B
  77777777B

  Note: 4095 is equivalent to 7777B

- special characters — These are generally ignored but must be present to separate the parameters. Characters other than letters or digits are special characters.

  Example: ,
  ;
  +

The meaning of each parameter type is a function of the command. These are discussed in detail in section 3.0. One point to note, however, is that unless stated otherwise in a particular command description, the order of these parameters is irrelevant.

2.5.4 Noise Words

There are certain names that are used by the command to specify certain options: STRING, TTR, LP, CLEAR and MAP are a few examples. Words, or names, other than those 'reserved' names recognized by the command may be inserted freely to improve read-ability. Some examples of this are shown below. The underlined items are the significant ones; other words, characters, etc. are considered noise and ignored.

// DUMP ONTO LP LOCATIONS 200B TO 1500B.

// CREATE A STRING FILE NAMED 'FILE1' FROM WS?

// RENAME FILE 'TEST3' AS 'TEST4'

// EXEC LOAD FROM CR AND 'PRG1' AT 500B. CLEAR AND MAP.

// CREATE A DATA FILE NAMED 'SAMPL1'. RECORD SIZE IS 15.

// ASSIGN 100 SECTORS TO FILE 'DATALG'

2.6 ERROR RECOVERY

Whenever the system encounters an error that requires user intervention, it types the message 'ERROR--text' on the teletype. The text following the two dashes describes the error condition. If PID is the TTK the system will type an '*' and expect a command to be typed. This command may be the same one with the error corrected or a new one. If PID is the CR, the '*' is typed and one of the words ABRT, CR, TTK or MTR must be typed as a response. If the response is none of these, the 'ERROR--test' message is again typed and the response must be entered again.

ABRT causes the job to be skipped while CR, MTR and TTK indicate that the command will be re-entered from the card reader, magnetic tape, or teletype keyboard respectively. If the user wants to ignore the command, he should type CR and not alter the card deck in the card reader. When the response is CR, a halt occurs to allow the input deck to be altered if necessary. Pressing START will cause processing to continue. When accepting commands from magnetic tape, the corresponding procedure may be used.

Below is a list of the monitor and system error messages. User messages produced by the command processors are described in Section 3.0 with the commands.

Error Messages:

TEXT	DESCRIPTION
// RECORD NEEDED	The record is not properly identified as a command.
IN COMMAND NAME	The command name is not one recognized by the monitor.
FUNCTION NOT IMPLEMENTED	The command processing program could not be located on the disc.
SYSTEM-1 • • SYSTEM-12	These messages indicate some malfunction or unusual condition encountered by the system itself. They are described in Appendix A.

Other messages that come out are of a warning nature only and do not require any intervention.

3.0 COMMAND DESCRIPTIONS

3.1 Introduction

This section describes all of the commands recognized by the system. The general form of each command is given along with specific examples; the following syntax notation is used:

- Parentheses '(',')' are used to enclose items that are optional.
- Where a choice is to be made from a set, the items in the set will be enclosed in angled brackets, '<','>' and separated by a slash.

  Example:

  ```
  <TTP/TTK/CR>
  ```

- 0 indicates that none of the elements of the set needs to be chosen.
- An underlined item in a set indicates that it is assumed if none are selected.
- Constant-names are shown in upper-case. They must always be written exactly as shown.

  Example:

  ```
  STRING
  TTP
  CLEAR
  ```

- Variable names or quantities are shown in lower-case. Their value changes with usage and must be supplied by the user.

  Example:

  ```
  string
  integer
  file-name
  ```

3.2 JOB

Generic form:

```
// JOB string
```

Examples:

```
// JOB 'CER'
// JOB 'TSTR'
```

Description:

JOB causes the system to initialize for processing a new job by restoring the COMREC from the permanent copy kept on the disc. The net result of this is 1) any information in working storage is lost, and 2) PID and POD are reset to the standard systems value. After boot-strapping the monitor system

from the card reader or magnetic tape, a JOB command with a non-blank string in the operand must be accepted by the system before it will accept any other commands. This is required in order to permit proper file referencing.

Warning Message:

DISC WRITE DISABLED	The disc cannot be written upon. Either the DCU or disc write-disable switch is set.

Error Messages:

TEXT	DESCRIPTION
MISSING PARAMETER	No non-blank string was present.

3.3 ASM

Generic form:

```
// ASM<TTK/TTR/CR/MTR/'file name'/O><TTP/LP/MTW/O>(SPASS) (LIST) (SYM) (OBJ) (INSEQ)
```

Example:

```
// ASM
```

The source program is read from PID and assembled. Unless there are SYM or OBJ directives in the source program, the only output that will be produced is that for statements or symbols that produce errors or warning flags. This output will be listed on POD.

```
// ASM 'TSTPR' LIST OBJ SYM INSEQ
```

The source program in the file TSTPR is assembled. The assembly listing and symbol table listing are listed on POD. The object program will also be produced in working storage. Columns 73-80 of the source records are checked for ascending sequence. Sequence blanks are skipped.

Description:

The Assembler Manual gives the information necessary for producing programs that are acceptable to the assembler.

The following points should be noted:

- ASM treats the operand parameters OBJ and OBJECT as equivalent. (Note: OBJECT is not recognized as a valid opcode in the source program.)

- ASM always leaves the generated object program in working storage.
- ASM will accept disc file input and has more I/O options.
- ASM produces the listing for the LIST option on the first pass if the assembly is a single pass (SPASS) assembly.

The operand parameters are used to specify the source of the input, the destination of the output and what output is to be produced. PID and POD are assumed if no input or output file is specified. Only statements and/or symbols that produce error or warning flags are listed if SYM or LIST are omitted and the corresponding directives are not in the source deck. LIST/NO LIST directives in the source file are not honored if 'LIST' is not an operand parameter.

Error Messages:

TEXT	DESCRIPTION
SYMBOL TABLE OVERLFOW	The symbol table is not large enough for the assembly. The program must be segmented.
DISC OVERFLOW	The capacity of the disc makes it impossible to assemble. The user should delete some files to make more space available.

Warning Messages:

IN FILE SPECIFICATION	Source file cannot be located or its type is not STRING or DATA.
EOF-END ASSUMED	The source program file has been exhausted, but no END card has been read.
MONITOR RECORD IGNORED	Source contained a monitor record.
MAGTAPE WRITE ERROR	Listing has a bad record. Assembly is continued.

3.4 EXEC

Generic forms:

```
// EXEC MTR/CR/TTR/TTK/O ('file name') (integer) (CLEAR)(DEBUG)(MAP)(CTRL)(XPND)
        // EXEC 'file name' (CLEAR)
```

Example:

```
// EXEC
```

The program in working storage will be loaded beginning at location 220B and executed.

```
// EXEC 'TEST1' 500B MAP CLEAR TTR CR
```

Object programs are to be loaded from paper tape, the card reader and the disc file TEST1. Prior to loading, core memory is set to zero from the loading point up. The program is to be loaded beginning at location 500_8 and when loading is complete a memory map will be listed on POD. If there is an object program in working storage, it will be loaded also.

```
// EXEC 'TEST2' CLEAR
```

The object file TEST2 is loaded after setting the core memory to zero from the loading point up. The program is loaded beginning at location 220_8. Working storage will also be loaded if it contains an object program.

```
// EXEC 'MAINPR' CLEAR CTRL
NOLOAD DISCIO, CPIO, GET, PUT
LOAD '*TEST', CLOSE, TVECT
NOLOAD TTPIO, OPEN
//
```

The object file 'MAINPR' is loaded after setting the core memory to zero from the loading point up. The files listed in the second and fourth (NOLOAD) records are not to be loaded from the corresponding disc files even though there are CALL records for them. The files listed in the third (LOAD) record are to be loaded even though there are no CALL records for them. (See Note 2, Pg. 18.) File names in CTRL records need not be put in quotes unless they contain special characters or start with a number. Working storage will also be loaded if it contains an object program.

Example:

```
// EXEC 'OVR1' CLEAR
```

The core image file OVR1 is loaded after setting core memory to zero from 500B up. (See page 17). No other options are required. (See Note 1, page 18.)

Description:

The first form of the EXEC command is used to load and execute object programs. The source of these programs may be any combination of the peripheral files from the TTK, CR, TTR, MTR, or disc file. Working storage is always loaded if it contains an object program; it will not be deleted and is unaffected by the loading. Input from a peripheral file must always terminate with an EOF record. (// in columns 1 and 2: rest of the 'card' blank).

When there is more than one source of input, the order of loading is as follows: TTK, CR, TTR, MTR, disc file and working storage.

When all of the programs at these sources have been loaded, there may be CALL statements whose corresponding PROC statements have not been loaded. If this is the case, EXEC will attempt to load a disc file whose name is the same as that of the missing PROC statement. In trying to find this file, EXEC will first look in the user's directory and then the system's directory. If no such file can be found, the same procedure will then be done for the remaining missing PROC statements. When this has finally been done for all such PROC statements and some are still missing, the missing ones will be flagged with a 'U' in the MAP listing. Executing a CALL for an undefined entry results in a HALT.

Note that a disc file can be loaded automatically only if there is a CALL statement whose name matches that of the file.

The automatic search facility can be suppressed and/or supplemented by placing the word CTRL in the operand. This option requires additional records to specify the names of files that are not to be loaded (searched for) even though they are CALLed. The source of these records is the same as that of the EXEC command. The series is terminated by an EOF record.

The format of these records is:

<LOAD/NOLOAD><name/'name'>(,<name/'name'>)

Example:

```
LOAD '*TEST', DISCIO, LPIO, '469X'
NOLOAD GET, PUT
LOAD CLOSE, 'XBQ&', H46T
```

The word LOAD or the word NOLOAD must appear first in each record, and all names in any one record are treated as specified by the first word.

If a file name appears in both a LOAD and a NOLOAD record, the NOLOAD option takes precedence. The file-names need not be put in quotes unless they contain special characters or start with a number.

The files listed in LOAD records are loaded in the order specified after all programs from TTK, CR, TTR, MTR, the named disc file, and WS; programs still missing and not appearing in NOLOAD records will be searched for and loaded after the programs specified by LOAD records. Note that a desired order of loading can be forced by specifying all the programs to be loaded in CTRL records.

The object programs that are to be loaded load one after the other into ascending memory locations. The first location to be loaded is location 220B, but this can be changed by placing the address of the first location to be loaded in the command operand field. Due to double precision operations, this number must be even and it will be increased by one if it is not. ABS assemblies load independently of the relocation process and should not load into the same area of memory that relocatable programs are occupying; under no circumstances should any program load below 220B.

The word CLEAR in the operand field causes core memory to be set to zero before loading the program. The area cleared is from the load point to the top of memory. (See Note 3.) The word DEBUG causes the debug procedure DEBUG to be loaded and given control when loading is complete; the user will then need to use a DEBUG command to transfer control to his program.

The word XPND causes EXEC's internal buffers to be shortened to allocate more space for the symbol table. This slows down the loading process noticeably, but should enable programs which overflow the standard symbol table to be loaded. (See Note 8.)

The MAP option will list on POD the address plus one of the highest location loaded followed by the names and addresses of all PROC statements. The address of the highest location loaded is also transmitted to the user's program in index register 0. Following the name is a code whose meaning is described in the following table:

CHARACTER	DESCRIPTION
space	Linkage established correctly.
N	PROC has no corresponding CALL.
U	CALL has no corresponding PROC.
D	More than one PROC with same name.
?	Combination of N and D.
0, 1, 2	SYSTEMS-ERRORS (See Note 2 & Appendix A.)

For 'D', the CALLs will be linked to the first PROC entry; executing a CALL for a 'U' entry will cause the machine to halt. Note that 'N' entries are frequently produced by PROC statements that establish interrupt linkages and by MAINPR PROC statements and are not errors. All PROC statements with codes other than space will be listed on POD even if the MAP option was not requested.

Example:

```
01754
MAINPR N   00250
LPIO       01500
LPIOIN N   01535
TTRIO      01557
TTRINT N   01600
TSUB1  U
TTPIO      01643
TTPINT N   01665
```

When loading is complete, EXEC attempts to transfer control to DEBUG if it was loaded; to the MAINPR PROC statement if one was loaded; or to the first PROC statement that was loaded. If DEBUG or MAINPR are not present, the name of the first PROC statement is listed on POD followed by 'IS ENTRY PNT'.

Example:

```
TEST1  IS ENTRY PNT
```

If the user's program executes a BRU* to its entry point when finished, the DOPSY system will be reloaded automatically provided the core area below 6100B has not been disturbed.

The second form of the EXEC command is used for loading core-image programs. It is the user's responsibility not to EXECute a core-image file that does not have an entry point! If CLEAR is specified, core memory between 500_8 and the end of memory will be set to zero (see notes 1 and 3), the program will then be loaded by the core-image loader, CIEXEC. The address plus one of the highest location loaded is transmitted to the user's program in index register '0'. If the user's program executes a BRU* to its entry point when finished, the DOPSY system will be reloaded automatically (see restriction above).

Error Messages:

TEXT	DESCRIPTION
IN FILE SPEC	An output device is specified, a disc file cannot be opened or its file type is not correct.
C.I. FILE ILLEGAL	CI files cannot be loaded with object files. (See Note 4)
IN CTRL RECORD	The format of a CTRL record is incorrect.
IN REC SEQ - XXXXXX	The object file XXXXXX has been scrambled. (See Note 5).
PROGRAM TOO BIG	The program is too large for available memory.
SYMBOL-TABLE OVERFLOW	There are too many PROC records in the object programs being loaded, i.e. $\sim$ 1000 PROC records in an 8K system (very unlikely). (See Note 8).
LOADING BELOW 220B	Programs cannot load between 0 and 220 octal.
NO ENTRY POINT	EXEC cannot determine where to transfer control. (See Note 6).
NO OBJECT PROGRAM GENERATED	No machine-instructions were actually loaded.
INV CKSUM XXXXXX	Object records have bad parity. (See Note 5).

Warning Messages:

XXXXXX IS ENTRY PNT

Neither DEBUG nor MAINPR PROC statements were loaded; control will be transferred to the PROC statement XXXXXX. (See Note 7).

EXEC NOTES

1. The other options should not appear. Most will be ignored (MAP, DEBUG, loadpoint, XPND); CTRL records will be processed but no programs will be LOADed or NOLOADed; requesting loading from a peripheral device (TTK, CR, TTR) will result in an error condition. (See Note 4.) Note that WS will never be loaded with a core-image program.

2. If a file named in a LOAD record has no corresponding PROC or CALL statements, it will be flagged with 0. This may or may not indicate that the file was not loaded; if an object file with that name exists in the user's directory or the system's directory, the file was loaded.

3. For core-image files, the area cleared is from 500B to the top of memory. The forced CLEAR caused by CREATE using EXEC clears the image of core from 0 to the top of memory.

4. This error will result from an attempt to CREATE a core-image file from an existing core-image file even if no object programs are loaded, or if any peripherals are specified when EXEC loads a core-image file (even if no programs are loaded from the peripherals).

5. See the ASSEMBLER MANUAL for this computer for a description of object-record format and sequence.

6. This check is purposely suppressed when creating core-image files.

7. If no PROC statements were loaded, the warning message XXXXXX IS ENTRY PNT with XXXXXX all blanks will be generated. If EXEC was called directly a NO ENTRY POINT error will occur, but if CREATE called EXEC to generate a core-image file, the operation will complete successfully. It is the user's responsibility not to attempt to EXECute a core-image file that does not have an entry point.

8. The option XPND can be used to allow loading a set of programs which overflow the normal symbol table. It expands the symbol table but slows down the loading process appreciably. If an overflow occurs with the XPND option, the number of PROC statements MUST be reduced.

3.5 CREATE

Generic forms:

// CREATE ('file name')<CR/TTR/TTK/MTR/0><STRING/OBJECT/OBJ/DATA(integer)/0>

<OVLY/0>

// CREATE 'file name' COREIMAGE<parameters required by EXEC>

Examples:

// CREATE 'TSTB12' CR STRING

The string file TSTB12 is to be created from a card deck.

// CREATE 'MATB' DATA 18

The file MATB is to be loaded from working storage. The file type is DATA and each record is 18 words in length.

// CREATE 'BINDEC' TTK STRING

The string file BINDEC is to be typed in from the teletype keyboard.

OVLY

// CREATE 'OVRLY1'^COREIMAGE 4000B '=OVR'

The core-image file OVRLY1 is to be created from object file '=OVR'. The entire image of the core memory will automatically be set to zero prior to loading. The loading origin for the program is 4000_8; a loading map will be produced.

// CREATE 'DEMONA' OBJECT OVLY

The object file DEMONA is to be created from working storage. If there is an existing file by that name, it will be updated.

Description:

CREATE is used to place data in a disc file. The first of the two forms is used for file types STRING, OBJECT, and DATA. The operand specifies the file type, the file name, and the source of data that is to be entered into the file. The parameters OBJ and OBJECT are equivalent. If no file type is specified, type STRING is assumed. If no file name is specified, the data will be loaded into working storage. If no data source is specified, the data will be obtained from working storage. (The case of working storage to working storage is essentially an NOP.) The size of DATA file records may be specified by placing an integer value ($2 \leq n \leq 20$) in the operand; if this value is omitted 20 words (80 characters) is assumed.

The source data is always assumed to be of the type specified in the operand. This condition is checked only when the data source is working storage.

There are two levels of protection provided against inadvertent overlaying of an existing file. If the user is intentionally updating an existing file, he may insert OVLY into the command telling CREATE to go ahead and overlay any existing file. If he does not do so and the file specified currently exists under his job number, he will be queried:

UPDATE OF EXISTING FILE?(Y/N)

Insertion of Y will cause the program to continue while insertion of any other letter will cause a return to Monitor and cancellation of the current request. For the purpose of these tests, a file to which space has been assigned, but which contains no data will be considered a new file.

If an old file is being rewritten, it must be large enough to accommodate the new data. If it is not, the old information may or may not have been destroyed; it depends on whether CREATE was able to determine the size of the new data before it was moved. (See Note 1.) When CREATE has completed all data transfers, the new file type is set in the directory entry. WS will be empty if it was the data source and/or if the file created had not existed previously.

The second form is used when a core-image file is to be generated. The operand parameters must occur in the order shown. Note that a file name must be specified. After the word COREIMAGE must come the parameters required by EXEC to load the program; see section 3.4.

The EXEC options MAP and CLEAR are forced when generating core-image files. The forced CLEAR option clears the entire image of core memory. A core-image file cannot be generated from an existing core-image file.

Error Messages

TEXT	DESCRIPTION
IN FILE SPECIFICATION	The source data file specified is an output device. (See Note 2.)
IN RECORD SIZE	DATA records cannot be larger than 20 words, or smaller than 2 words.
IN FILE TYPE	WS type does not agree with the type specified in the operand.
DIRECTORY FULL	The file directory is full and has no room for the new entry. (688 entry limit).
FILE TOO SMALL	The old file is not large enough for the new data. The old data may or may not be destroyed. (See Note 1.)

INVALID FILE NAME	The file name specified failed to start with a non-blank character or was a reserved name ($DIRCT or $ARR).
MISSING PARAMETER	The file name before the word COREIMAGE was omitted.
DISC OVERFLOW	There is not enough space available on the disc for the new core-image file.

CREATE NOTES

1. If the input was from working storage or the file type is COREIMAGE, the old data is not destroyed.

2. This error can also be generated by various file referencing errors in EXEC when generating core-image files. (See Note 3.)

3. EXEC may generate additional error-messages when generating core-image files (see 3.4) since CREATE calls EXEC to relocate the program.

3.6 ASSIGN

Generic form:

// ASSIGN 'file name' integer<WORDS/SECTORS/0><STRING/DATA(integer)/0>

Examples:

```
// ASSIGN 'TEST4' 30 SECTORS

// ASSIGN 'DATAF1' 10000 WORDS STRING

// ASSIGN 'DATA' 8000 WORDS DATA 12
```

Description:

ASSIGN is used to save space for a new file or to change the amount of space allocated to an old file. The first integer in the operand specifies the amount of space to be assigned to the file. The units may be SECTORS or WORDS; in the latter case, this value will be rounded up to the nearest multiple of 48.

If the file is an old one, enough space will be added or removed from the end of the file to get its allocated space to agree with the amount specified. In no case, however, will space be removed if it contains useable information. Attempting such an operation causes a warning message to be issued; the size of the file will be decreased as much as possible (to that required to retain all of the useable information).

If the file is a new one, its type can be specified as STRING or DATA. If the type specification is omitted, type STRING is assumed. If the type is DATA, the record size can be specified by placing a second integer ($2 \leq n \leq 20$) in the operand. If the record size specification is omitted, 20 words (80 characters) is assumed.

Error Messages:

TEXT	DESCRIPTION
MISSING PARAMETER	The file name was omitted.
DIRECTORY FULL	The file directory is full and has no room for the new entry.
IN RECORD SIZE	DATA records cannot be larger than 20 words or smaller than 2 words.
INVALID FILE NAME	The file name specified failed to start with a non-blank character or was a reserved name ($DIRCT or $ARR).
DISC OVERFLOW	The requested extra space is not available on the disc. The operation is suppressed.

Warning Messages:

DELETION OF USEABLE INFORMATION SUPPRESSED	The requested reduction of file size would destroy useable data; the file size is reduced to the minimum size required to retain all the data.

3.7 FDUMP

Generic form:

```
// FDUMP<'file name'/O><TTP/LP/CR/MTW/O>(integer)(LEADER)(DIRECTORY)(OCTAL)
// FDUMP WS
```

EXample:

```
// FDUMP DIRECTORY
```

This option will provide the user with a listing of the files in his directory, in this case on POD.

```
// FDUMP LP
```

This command provides a listing of working storage on the line printer. The contents of working storage is destroyed.

```
// FDUMP 'BPRGA' TTP
```

The file BPRGA is punched onto paper tape.

```
// FDUMP 'TSTDAT' LP 10
```

The first 10 records of file TSTDAT are listed on the line printer.

```
// FDUMP DIRECTORY 'BTEST'
```

This option allows the user to dump the directory entry of a specified file, in this case to POD.

```
// FDUMP '=XYZ' OCTAL MTW
```

This command will cause the object file '=XYZ' to be dumped in octal format to the magnetic tape. Such a magnetic tape could then be listed at a later time on the line printer.

Description:

The FDUMP command with the DIRECTORY option allows the user to dump the contents of his directory to one of the output devices listed above. If no device is specified, the output will be to POD.

The output of a typical directory listing is shown below.

NAME	TYPE	ASSIGNED	USED
OVRLY1	C	001056	000987
TEST4	D	000960	000480
OBJ5	O	000480	000443
PRGA	S	019200	011000

If a file name is also specified with the word DIRECTORY, only the information for the specified file will be listed. When operating under the system job number, if the file name ' /↑↑' is specified, the entire disc directory is dumped and the job number for each file is included.

The units for the amount of space assigned and space used is words (in decimal). If the file is core-image, the number of words is the sum of the program load and the sectors required to retain interrupt loading information. The type entries C, D, O, and S stand for COREIMAGE, DATA, OBJECT and STRING, respectively. See parameter WS description (below).

The second form is used to dump a disc file to a peripheral device, If, however, no disc file is specified working storage is dumped, while omission of a peripheral device specification causes information to be dumped to WS. The case of WS to WS is an NOP. Generally, an entire file will be dumped, but an abbreviated dump can be obtained by placing an integer in the operand. This specifies the number of records to be dumped, so output terminates on an EOF or the record count being satisfied.

The word LEADER can be used if output to TTP is also going to be punched. This will cause FDUMP to halt while the paper tape punch is turned on. When execution resumes, a ten inch leader will be punched, then the file, and lastly a ten inch trailer.

The parameter OCTAL allows the user to request a listing of a file, typically DATA, OBJECT, or COREIMAGE, in octal format. Such a printout would be directly readable without the necessity of other conversions.

The parameter WS (included in the second format) tells the user the extent of the current disc working storage area. This information is automatically added to any directory information request. The output format is:

WORKING STORAGE HAS XXXX WORDS

XXXX is a decimal number.

Error Messages:

TEXT	DESCRIPTION
IN FILE SPECIFICATION	An input device is used where an output device was expected or disc file cannot be located.

DISC OVERFLOW	Working storage is not large enough to contain the file.

3.8 DELETE

Generic Form:

```
// DELETE<'file'( ,'file name')/O>
```

Examples:

```
// DELETE 'OBJ5A'

// DELETE 'TEST1','TEST2','TEST3'
```

Description:

DELETE is used to remove a file from the disc. For each file specified in the operand the following actions occur.

- The name of the file is removed from the directory.
- The space allocated to the file is made available for reassignment.

If no file names are specified, working storage is deleted; that is, it is made to look empty.

Users' Note: For maximum efficiency, the files must appear in the operand in reverse order of their occurrence on the disc (from the bottom of a directory-listing upward).

Error Messages:

TEXT	DESCRIPTION
INVALID FILE NAME	A file name specified failed to start with a non-blank character or was a reserved name ($DIRCT or $ARR). The entire command is ignored.

Warning Messages:

XXXXXX MISSING	A named file was not located in the directory. The remainder of the command is processed normally.

3.9 RENAME

Generic forms:

```
// RENAME 'file name' 'file name'

// RENAME JOBNUMBER string
```

Examples:

```
// RENAME 'TEST3' AS 'NUTEST'
```

The file TEST3 will be renamed NUTEST.

```
// RENAME JOBNUMBER 'JPQ#'
```

The files belonging to the current job number will be assigned to the job number JPQ#.

Description:

The first form of RENAME is used to change the name of a file. The first operand file name is used to select the file whose name is to be changed while the second is the new name of the selected file.

The second form is used to change the job number required to access the files belonging to the current job; the new job number is specified by the high-order four characters of the string. The current job number will also be changed to the new value.

Error Messages:

TEXT	DESCRIPTION
MISSING PARAMETER	One or both of the two required file names are missing.
IN FILE SPECIFICATION	The old file cannot be located.
INVALID FILE NAME	The file name specified failed to start with a non-blank character or was a reserved name ($DIRCT or $ARR).
DUPLICATE FILE	A file already exists with the new file name.

3.10 DUMP

Generic form:

// DUMP <integer/integer ξ integer> <TTP/LP/MTW/O>

where ξ is - or * or , .

Examples:

// DUMP ONTO LP LOCATION 200B-300B AND 5000B, 6000B

The values in locations 200 through 300 octal and 5000 through 6000 octal are listed on the line printer.

// DUMP 100B TTP 400B, 420B

The values of location 100B and 400 through 420 octal are listed on the teletype.

Description:

DUMP is used to output the contents of core memory. The destination of the output may be any of the devices shown. If none is specified POD is assumed. The area(s) of core to be dumped are specified by an integer or integer pairs separated by a non-numeric character. Any number of these may be specified and a device specification may occur anywhere in the operand. The extent of zero fields will be issued as a message: Zero Field NNN-PPP where NNN and PPP are in octal format.

Note: DUMP reads the first 4K of core from the coreimage buffer. To have the latest version, the disc must be enabled and a branch to 125B must be done before executing DUMP.

3.11 NOTE

Generic form:

// NOTE '(any characters other than quote)'<TTP/LP/MTW/O>(HALT)

Examples:

// NOTE 'PLACE PINK CARDS IN CARD READER' TTP HALT

// NOTE 'PHASE 1 ABORTED'

Description:

Note outputs the text between the quotes to the specified device. If the device specification is omitted, the text will be outputted to POD, unless POD is TTP in which case nothing is printed. The word HALT will cause the CPU to stop executing instructions until START is pressed.

The message text must occur before the other parameters.

Error Messages:

TEXT	DESCRIPTION
MISSING DELIMITER	A quote is missing.
IN FILE SPECIFICATION	An input device is specified instead of an output device.

3.12 SET

Generic form:

```
// SET<CR/TTK/TTR/MTR/0><LP/TTP/MTW/0>
          (input)          (output)
```

Examples:

```
// SET INPUT TO TTK
```

This command will cause the system to look for future commands at the teletype keyboard.

```
// SET CR TTP
```

The primary input device is set to CR and the primary output device is set to TTP.

Description:

SET is used to temporarily alter the devices assigned to PID and POD. These assignments are effective only until the next JOB or SET command; in the former case, PID and POD are reset to their standard devices while in the latter they are set to the devices specified by the user.

3.13 PATCH

Generic form:

```
// PATCH
```

Description:

PATCH may be used to modify string, object, data, or coreimage files. For string, object, and data files, addresses given to PATCH must start relative to the first word of the file, e.g. (0). For coreimage files, PATCH will accept only those addresses relative to the first absolute core location of the program being patched. PATCH will not allow alteration of interrupt locations which are part of coreimage files. If there is not enough space at the end of a file, ASSIGN can be used to increase the file size. Attempting to address locations below the lowest file word or beyond the last available word in the file will result in an error response '??'.

To execute PATCH , type // PATCH under any job name with the DOPSY monitor system and disc enabled.

PATCH is now waiting for one of the following commands which can be abbreviated to first character if desired. A space must delimit the command.

Note: Decimal values are assumed if not terminated with a B, which symbolizes octal.

1. OPEN 'File Name'

2. READ NNNB (single entry)
   or
   READ NNNB - QQQB (for string)

3. WRITE NNNB: QQQB (single entry)
   or
   WRITE NNNB: QQQB, RRRB, etc. (for string)

4. CALC A + B + C = NNNB

   A, B, C are any numbers in a simple add/subtract operation. The answer is always in octal.

5. DOPSY

The OPEN command must precede all others except CALC. Its purpose is to locate the user's file and set up pertinent information for executing the READ and WRITE commands.

The READ command will list the current values of the addresses of the user's file provided the addresses exist. They are output in octal, eight (8) to a line starting at the first entered address (value) of the command.

The WRITE command will enter the data following a colon sequentially from the address supplied after the word WRITE . Values should be suffixed with B for octal values.

CALC is self-explanatory.

An input error which PATCH finds will be terminated with a '??'. It is for the user to figure these out.

3.14 MTAP

Generic forms:

```
// MTAP   TMARK

// MTAP   SKIP   n   <FILE/FILES/REC/RECS> <FWD/BACK> (WAIT)


// MTAP   REW (WAIT)
```

Description:

MTAP is a utility tape handler for use by the user in formatting and processing magnetic tape.

Option 1 of the command will cause a tape mark to be written at the current position of the magnetic tape.

Option 2 may be used to position the tape for processing by SKIPping n files or records forward or backward. If WAIT is specified, the next command (in the job string) will not be executed until the SKIP is completed.

Option 3 will cause the rewinding of the tape. WAIT serves the same purpose as for SKIP.

Error Messages:

TEXT	DESCRIPTION
UNRECOGNIZABLE FIELD	MTAP does not understand the command. Re-enter.

3.15 COMPILE

Generic form:

// COMPILE<TTK/TTR/CR/'file name'><TTP/LP><LIST/OBJ/LISTOBJ>

Description:

The first group of enclosed options in the above command indicates that the compiler input may be specified as coming from the Teletype Keyboard, from paper tape via the teletype paper tape reader, from cards via the card reader, or from a file previously created.

For further details consult the FACTOR manual.

4.0 MONITOR OPERATION

Providing ARR has not been destroyed, DOPSY is easily reloaded by transferring control to location 125B. If ARR is not available, however, it is necessary to execute the following program in order to load DOPSY.

```
DCBP    DATA    *+1,96,60B,406B
RETRY   LDA     DCBP
        RD      70B
        BOI     3,*-1
        BOI     4,100B      ENTER ARR
        BRU     RETRY
```

If a card reader is available, this program can be loaded via the LOAD switch on the CPU; otherwise, it must be entered manually each time ARR is destroyed. This load program cannot be executed between 60B and 220B.

SECTION V

REFERENCES

1. FST-1 SYSTEMS MANUAL

2. FST-1 ASSEMBLER USER'S MANUAL

3. FST-1 SUBROUTINE LIBRARY MANUAL

4. FST-1 UTILITIES MANUAL

5. SENTRY-400 FACTOR MANUAL

6. SENTRY-400 TOPSY MANUAL

APPENDIX A

DOPSY System Error Messages

SYSTEM - 1	$UPDAT overlay missing (called by $CREAT, $CREAT2, $DELET, or $ASSIG).
SYSTEM - 2	"Never-happen" error-returns (e.g., PUTW E-O-F).
SYSTEM - 3	$DIRCT cannot be opened or closed.
SYSTEM - 4	Working storage cannot be opened or closed.
SYSTEM - 5	$ASM1 overlay missing.
SYSTEM - 6	$CREAT or $CREA2 can't close a file just created. (ENTRFN or DISCIO or hardware: Disc Write Inhibited).
SYSTEM - 7	$CREA2 overlay missing (called by $EXEC).
SYSTEM - 8	$EXEC overlay missing (called by $CREAT).
SYSTEM - 9	Working storage is exhausted or INPUT START ADDRESS >TOP OF WS+1 (see $ASSIG and $DELET logic writeups).
SYSTEM - 10	Core-buffer area of disc cannot be written correctly after 10 attempts (usually bad parity).
SYSTEM - 11	SENTRY 400 software error: Missing TOPSY overlay (CPMAIN, DATALOG, DLGIOF, FCFAIL, DCFAIL, ERROR or MEASURE).
SYSTEM - 12	$COMP1 overlay missing.
SYSTEM - 20	FCOMP1 overlay missing.
"DISC ERROR.	PRESS START TO RE-TRY. n"

n	Routine
1	$EXEC
2	$UPDAT
3	$DELET
4	$ASSIG