HEWLETT hp PACKARD

MOVING-HEAD DISC OPERATING SYSTEM

MOVING-HEAD
DISC OPERATING SYSTEM

HEWLETT hp PACKARD
11000 Wolfe Road
Cupertino, California 95014

02116-91779

August 1970

PREFACE

MOVING-HEAD DISC OPERATING SYSTEM is the programmer's and operator's guide for the Hewlett-Packard Moving-Head Disc Operating System (DOS-M). DOS-M is a batch processing system that executes complete jobs without operator intervention. For a full understanding of DOS-M the reader should be familiar with one of the Hewlett-Packard programming languages, as presented in the *FORTRAN* (02116-9015), *ALGOL* (02116-9072) and *ASSEMBLER* (02116-9014) programmer's reference manuals.

The Introduction of this manual explains the software and hardware elements of the system. Section I presents the system organization, while Sections II and III cover the complete set of batch and keyboard directives and program calls to the system. All facets of DOS-M programming --FORTRAN, ALGOL, Assembler, Loader, DEGUB, and Library -- are presented in Section IV. Section V assembles all the necessary information on input/output, including the planning of I/O drivers. Procedures for installing and initiating the software appear in Section VI. The appendices provide tables, summaries, a complete listing of error messages, and sample job decks.

CONTENTS

SECTION II (cont.)
DIRECTIVES

ILLUSTRATIONS

TABLES

INTRODUCTION

In the Moving-Head Disc Operating System (DOS-M), software modules are stored permanently on the disc for high-speed batch processing, eliminating slow and inefficient paper tape loading. Input can be set up and executed in serial order to automatically edit, translate, load and execute a set of source programs written in HP FORTRAN (an extension of ASA BASIC FORTRAN), HP ALGOL, or HP Assembly Language. A variety of files can be stored, edited, listed, dumped and used as input to programs.

FEATURES OF DOS-M

DOS-M contains the following highlights and features:

- Keyboard and batch processing modes,
- Software programming aids: FORTRAN Compiler, Assembler, Relocating Loader, Relocatable Library, Debug Routine, Source File Editor, and ALGOL.
- Jobs executed in a queue with no operator intervention,
- Symbolic disc files, with relative addressing,
- Centralized and device-independent I/O processing,
- Modular structure,
- Custom configuration to optimize available memory and I/O,
- Cyclic error checking on disc read & write operations,
- Exchangeable discs packs, and
- Optional search of the entire system for file names.

DOS-M HARDWARE CONFIGURATION

DOS-M will operate on either an HP 2114 computer or an HP 2116 computer. This causes several variations in hardware configuration.

1. No MEMORY PROTECT on the 2114. DOS-M operates either with memory protect, as on the 2116 (recommended) or without, as on the 2114. (Without memory protect, user programs can destroy the system area of core.)
2. No EAU instructions on the 2114. EAU is available on the 2116, but DOS-M does not require these instructions.
3. Direct Memory Access (DMA): DOS-M uses one channel DMA on the 2114 and two-channel DMA on the 2116.
4. Memory Size: Minimum memory for DOS-M is 8,192 words (2114 or 2116). Larger memories can be used on the 2116 only.
5. Input/Output Channels: The minimum 2114 has 7 channels versus 16 on the 2116.

MINIMUM HARDWARE

The minimum hardware requirements for the Moving-Head Disc Operating System are:

1. Computer (2114B or 2116B) 8,192 words of memory, Central Interrupt Processor, 1 channel DMA, halt on memory parity error.
2. HP 2870 Moving-Head Disc Drive with fixed disc and a removable cartridge.
3. System Input Device: Teleprinter (HP 2752).
4. Batch I/O Device: Second Teleprinter (HP 2754).

In place of the HP 2754B teleprinter, the user can select one of the following combinations instead for batch operations:

Batch List Device	Batch Input Device	Batch Punch Device
HP 2752A Teleprinter	Punched Tape Reader	Punch Unit
HP 2752A Teleprinter	Mark Sense Card Reader	Punch Unit
Line Printer	Punched Tape Reader	Punch Unit

The following hardware options are available:

1. Time Base Generator (provides accounting times).
2. Extended Arithmetic Unit (EAU) on 2116 only (provides hardware multiply, divide, etc. for user programs).
3. Additional memory: 16,384 or 32,768 words on 2116.
4. Additional I/O channels: extenders are available on the 2114 and 2116.
5. Memory Protect (2116 only).
6. Photoreader.
7. Paper Tape Punch.
8. Line Printer.
9. Mark Sense Card Reader.
10. HP 3030 Magnetic Tape Unit
11. Up to three additional Disc Drives (all four operate on one controller).
12. Plotter

DOS-M SOFTWARE MODULES

DOS-M consists of the following programs:

DOS-M Supervisor and sub-modules
DOS-M Assembler
DOS-M FORTRAN Compiler
DOS-M Relocating Loader
DOS-M Moving Head Disc Driver (DVR 31) (uses DMA)
DOS-M Special Teleprinter Driver (DVR 05)
DOS-M DSGEN (the system generator)

In addition, the following programs can be included when DOS-M is generated:

RTE/DOS ALGOL Compiler (16K memory required)

RTE/DOS Relocatable Library (EAU or Non-EAU)

RTE/DOS FORTRAN IV Library (extended precision arithmetic)

DOS I/O Drivers (either core- or disc-resident):

- Teleprinter (DVR ØØ)
- Photoreader (DVR Ø1)
- Tape Punch (DVR Ø2)
- Line Printer (DVR 12)
- Mark Sense Card Reader (DVR 15) (uses DMA)
- 3030 Magnetic Tape (DVR 22) (uses two-channel DMA)
- Plotter (DVR 1Ø)

DOS-M Supervisor

The DOS-M supervisory software consists of a monitor (DISCM) that is partly core-resident and partly (optionally) disc-resident and a disc-resident job processor (JOBPR):

DISCM	JOBPR
Interrupt Processor	Job Processor
Executive Processor	File Manager
I/O Processor	
Executive modules:	
$EXØ1 through $EX2Ø	

NOTE: Exec modules can be made either core- or disc-resident when DOS-M is generated.

NOTE: JOBPR is always made disc-resident when DOS-M is generated. DISCM brings it into core when needed.

SECTION I
SYSTEM ORGANIZATION

An operating system is an organized collection of programs which increases the productivity of a computer by providing common functions for all user programs.

An operating system's function is to aid in the preparation, translation, loading, and execution of programs. This is accomplished by an auxiliary, quick access memory, usually a disc storage unit. The various translators, loaders, and other software are stored permanently on the disc for use only when needed. Since the programmer requests a compiler from the disc instead of loading it by hand from paper tape, the overhead time can be significantly reduced.

DOS-M

The Moving-Head Disc Operating System is composed of user disc files and the DOS-M Supervisor. The Supervisor consists of two parts: a Disc Monitor (DISCM) and a Job Processor (JOBPR). DISCM consists of modules which are either core- or disc-resident and handle I/O transfers, requests from programs, and other supervisory tasks. The disc-resident JOBPR handles operator and programmer directions from the batch or keyboard device.

The Moving-Head Disc Operating System affords speed and convenience. Programs can be input to DOS-M for automatic translation, loading, and execution. For example, simple punched cards carry out load-and-go operations in DOS-M as follows:

a. DOS-M reads the FORTRAN Compiler into core from the disc.

b. The Compiler reads the source program from an external device, such as a card reader, and stores the relocatable binary instructions on the disc.

c. DOS-M reads the Loader into core from the disc.

d. The Loader reads the relocatable binary programs from the disc and stores the converted binary instructions on the disc.

e. DOS-M reads the program in from the disc and runs it.

Directives

The DOS-M Supervisor operates in response to directives input by the programmer or operator. Directives are strings of up to 72 characters that specify tasks to DOS-M. They are entered in one of the two modes of DOS-M operation: keyboard or batch. In keyboard mode, the directives are entered manually from the teleprinter keyboard. In batch mode, directives can be input as punched cards integrated with the source program into a *job deck*.

A job is a related set of user tasks and data. In keyboard mode, the directives (tasks) are entered separately from the job data. In batch mode, they are included in a job deck that can execute without manual intervention. Jobs may be stacked directly upon one another in a queue.

The DOS-M directives are used for the following functions:

- Create, edit, list, dump, and purge user files (relocatable, loader-generated, source and ASCII or binary data).
- Turn on systems programs such as FORTRAN, Assembler, etc.
- Modify the logical organization of the I/O.
- Start and stop a job; type comments; suspend operations.
- Translate, load and execute a user program.
- Dump core or disc memory.
- Resume execution of suspended programs.
- Set the date; abort programs; transfer to batch mode (from keyboard mode); return to keyboard mode (from batch mode).
- Check status of user disc tracks.
- Change the subchannel of the user disc.
- Search the various disc subchannels for specified file names.
- Initialize (label) disc.
- Dump a disc to another disc.

DOS-M directives are described in detail in Section II.

EXEC Calls

After being translated and loaded, an executing user program communicates with DOS-M by means of EXEC calls. An EXEC call is a JSB instruction which transfers control to the DOS-M Supervisor.

The EXEC calls perform the following functions:

- I/O read and write operations.
- User file and work area read and write operations.
- I/O control operations (backspace, EOF, etc.),
- Request I/O status.
- Change the subchannel of the user disc.
- Request limits and status of WORK area (temporary disc storage).
- Program completion.
- Program suspension.
- Loading of program segments.
- Request the time.

Section III describes EXEC calls in detail.

Input/Output

All I/O operations and interrupts are channeled through the DISCM section of the DOS-M Supervisor. DISCM is always core-resident and maintains ultimate control of the computer resources. (See *"Software I/O Structure,"* Section V.)

I/O programming is device-independent. Programs written in FORTRAN, ALGOL, and Assemvly Language specify a logical unit number (with a predefined function, such as data input) in I/O statements instead of a particular device. Logical unit numbers are assigned to appropriate devices by the operator, depending upon what is available. Thus, the programmer need not worry about the type of input or output device performing the actual operation. (See *"Logical Unit Numbers,"* Section V.)

Core Layout

When DOS-M is active, the core memory is divided into a user program area and a system area (as shown in Figure 1-1). The Disc Monitor program handles all EXEC calls and, if they are legal, transfers them to the proper module for processing. The I/O drivers handle all actual I/O transfers of information. If some I/O drivers are disc-resident, they are read into core by the supervisor when needed. The user program area provides space for execution of user programs. In addition, large DOS-M software modules, such as the FORTRAN Compiler, Assembler, Relocating Loader, and Job Processor, reside on the disc and execute in the user program area.

If the memory protect option is present, a memory protect boundary is set between the executive area and the user program area. This boundary interrupts whenever a user program attempts to execute an I/O instruction (including a HALT) or to modify the executive area. (Instructions can reference the switch register and overflow register.) Programs to be run in the user area must use EXEC calls for input/output, termination, suspension, and other external processes.

DISC USAGE

The controller for the moving-head discs supports up to four disc drives (one is required). Each drive contains two discs: a fixed disc and a removable cartridge. Each disc is referenced through a subchannel of the controller. Therefore, the controller has eight subchannels (numbered 0 to 7). The channels are assigned as follows:

Disc Drive Numbers	0	1	2	3
Permanent Subchannels	0	2	4	6
Removable Subchannels	1	3	5	7

Each disc contains 203 tracks of 24 sectors each. At least three of these tracks must be spares. (A sector contains 128 16-bit words and is the smallest addressable unit on the disc.) DOS-M normally allows two subchannels to be available to the user: one subchannel contains the system disc and the other contains the user disc (may be the same subchannel as the system disc). The user subchannel can be changed during job or program execution. In addition, an optional system search mode is available to allow searching for user files on any specified subchannels.

The disc storage has four parts:

1. The System Area:
   Executable code created by the system generator and hardware protected; includes DOS-M Supervisor and other system programs.
2. The User Area (optional):
   User file directory and user files (data, object programs, source statements, etc.).
3. The Work Area:
   Temporary storage for the current job.
4. Job Binary Area:
   Temporary storage for relocatable object code generated by the assembler and compilers; this is an area of variable size and starts from the end of the disc.

All four of these areas can reside on the system disc. Or the user area can be on a separate subchannel. Only one user area is available to the system at a time. The standard user subchannel is assigned at system generation time; this can be the system disc or another subchannel (removable or permanent disc). The :UD directive and an analogous EXEC call allow the user to temporarily change the user area to another subchannel.

Automatic track switching is provided within each subchannel.

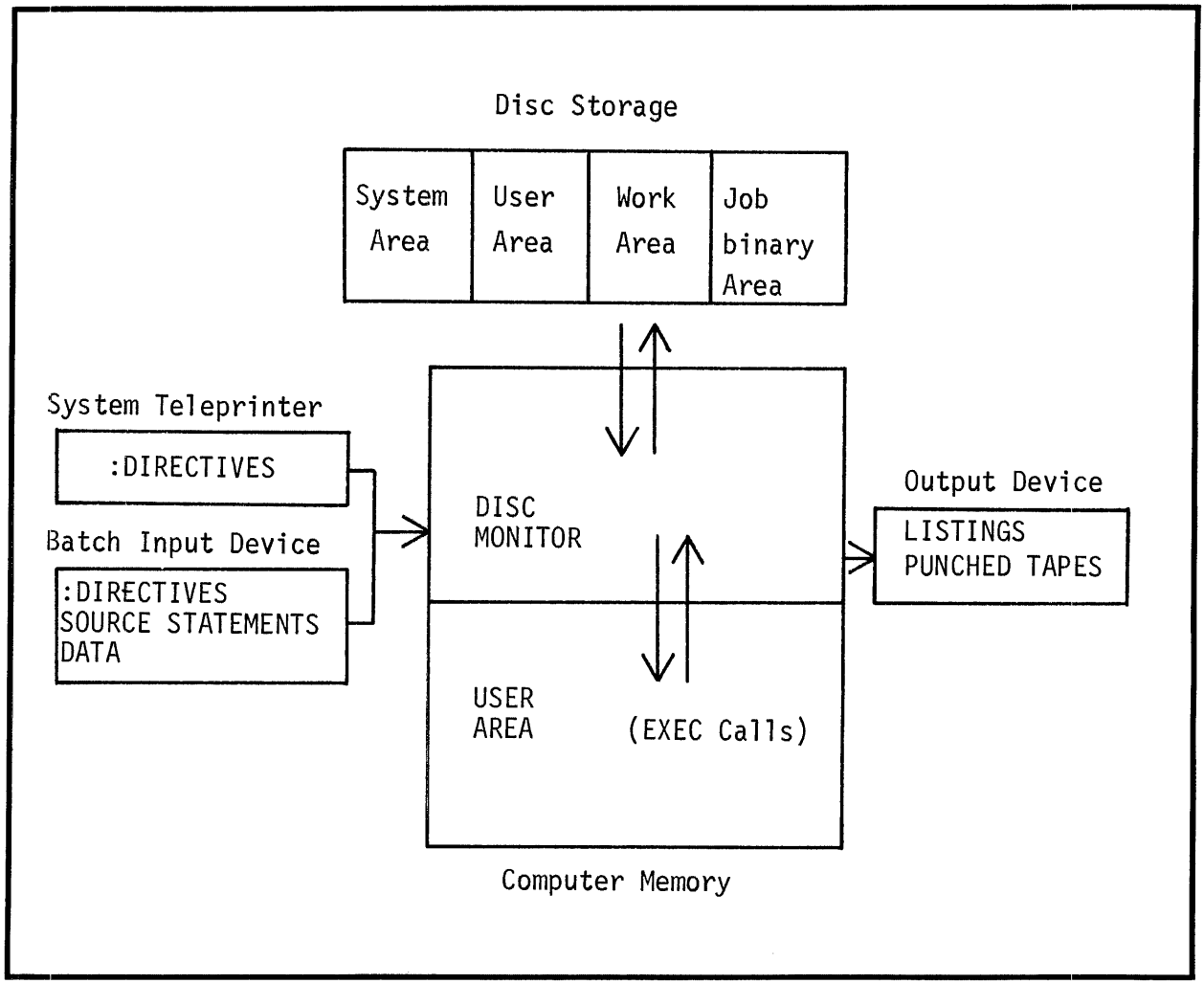


Figure 1-1. Functional Diagram of DOS-M

DOS-M Files

The disc provides quick access and mass storage for user files consisting of source statements, relocatable and loader-generated object programs, and ASCII or binary data. Each file has a name that is used to reference it.

Programs use the work area of the disc for temporary storage. The system area contains files of systems programs, EXEC modules, a system directory, and library subroutines (see *LIST*, Section II).

DOS-M Installation

DOS-M is a series of relocatable binary software modules. Since each module is an independent, general purpose program, the hardware and software configuration of each DOS-M is quite flexible. A separate absolute program, DSGEN, accepts the software modules and generates a configured DOS-M following dialogue-type instructions from the user. (See *DOS-M Generator*, Section VI.)

Certain DOS-M modules may be either core- or disc-resident. In a minimum 8K core system, all possible modules are disc-resident; but a 16K memory allows more modules to be core-resident for greater efficiency.

An absolute copy of the configured DOS-M is stored on the disc and is protected from alteration by a hardware override switch. A bootstrap program is used to initiate DOS-M from the disc.

SECTION II
DIRECTIVES

Directives are the direct line of communication between the keyboard or batch input device and the Moving-Head Disc Operating System. The operator enters these directives manually through the keyboard or the programmer enters them on punched cards within his job deck. Directives are able to:

- Initiate, suspend, terminate, and abort jobs,
- Switch between keyboard and batch mode,
- Execute, suspend, and resume programs (including compilers, loaders, etc.),
- Print the status of the disc tracks and the I/O tables,
- Create and purge files of source statements, relocatable and loader-generated binary programs, and ASCII or binary data,
- Edit source statement files,
- Set up source files for compilers and assemblers,
- List and dump files, dump disc and core,
- Declare I/O devices up and down,
- Set the date and print comments,
- Change user disc subchannel,
- Dump a copy of a disc onto another subchannel,
- Search specified subchannels for file names,
- Initialize a disc, and
- Turn off an executing program.

Directives may enter DOS-M in two modes: keyboard and batch. In either mode, all directives are listed on the teleprinter. Certain directives are legal in one mode only; other directives are operable in both. In keyboard mode, the operator manually inputs the directives through the teleprinter keyboard. In batch mode, the programmer prepares the directives on punched cards or paper tapes and inputs them along with programs, data, etc, in a complete job.

Directives have the same format, regardless of the mode in which they occur: ":" followed by a directive word (first two characters are significant) and, if necessary, a list of parameters separated by commas (maximum is 15). For example,

```
:PROG,FTN,99
```

When optional parameters are missing, they must be represented by commas if the following parameters are to be recognized. The first blank character not preceded by a comma is the end of the directive. Comments may appear after this blank; they are ignored by DOS-M. A "rubout" anywhere in a directive deletes the entire directive, while a "control-A" (striking the "A" key and the "control" key simultaneously) deletes the previous character.

DOS-M has two conventions for notifying the operator that directives may be entered. An asterisk (*) means that DOS-M is waiting for an operator attention directive (see below). A "@" with the bell signals that DOS-M is waiting for further directions. (During some operations, such as editing, there may be perceptible waits while DOS-M processes the directive. Further directives *must* not be input until the "@" is output.)

The operator attains control of DOS-M at any time by striking any system teleprinter key. If the teleprinter is available, DOS-M prints an asterisk (*) on it; if it is busy, DOS-M prints an asterisk as soon as it is free. At this time, the operator may enter any of the following directives (described in detail in this section):

```
:ABORT
:DN
:EQ
:LU (reports only)
:TYPE
:UP
:OFF
```

If the operator types any other directives, DOS-M prints the following message and returns to the executing program.

```
IGNORED
```

JOB

Purpose

To initiate a user job and assign it a name for accounting purposes.

Format

:JOB[,*name*]

where *name* is a string of up to five characters (starting with an alphabetic character) which identifies the job.

Comments

When DOS-M processes the JOB directive, it prints an accounting message on the system teleprinter and the list device recording the job's *name* (as specified in the JOB directive), the date (as specified in the DATE directive) and the current time (if a time base generator is present):

JOB *name date* TIME = *xxxx* MIN. *xx.x* SECS.

or

JOB *name date* (if no time-base generator)

For example,

```
:JOB,START
JOB START MON 6.16.9     TIME = ØØ13 MIN 41.6 SEC.
```

or

```
JOB START MON 6.16.9
```

If an EJOB directive has not been encountered, JOB also acts as the EJOB for the previous job. In this case, all actions of the EJOB are carried out, except for returning to keyboard mode from batch mode, before starting the new job.

Only the first two characters of JOB are significant. DOS-M skips everything up to the comma.

EJOB

Purpose

To terminate the current job normally and return to keyboard mode.

Format

```
:EJOB
```

Comments

EJOB condenses all user discs by eliminating spaces left by non-permanent programs. (:EJOB follows the :SS condition.) EJOB outputs a message recording the total job and execution time, then returns to keyboard mode. (See STORE directive and *Relocating Loader*, Section IV.) All directives except TRACKS, OFF, or BATCH are ignored until the next JOB directive.

EJOB resets logical units 1 through 9 and resets the :SS condition. EJOB resets the user disc assignment to the standard subchannel unless the standard is not ready or a new cartridge has been inserted (with a different label and without a :UD directive).

When the EJOB directive occurs, a message is printed, similar to that of JOB, giving the total run time of the job and total execution time (if a time-base generator is present). For example,

```
END JOB START RUN = ØØØ7 MIN. 52.6 SEC. EXEC = ØØØ1 MIN. 21.Ø SEC.
```

or

```
END JOB START (No TBG)
```

This message is printed on the system teleprinter and on the standard list device.

ABORT

Purpose

To terminate the current job before the next JOB or EJOB directive.

Format

:ABORT

Comments

ABORT carries out all the operations of an EJOB. All I/O devices are cleared. When it returns to the batch device, DOS-M ignores all directives, except TRACKS, OFF, BATCH, or TYPE, until it finds a new JOB directive. An ABORT may be entered through the keyboard, even if DOS-M is in batch mode.

:OFF must never be given during a purge or after a /E in an EDIT list.

PAUSE

Purpose

To interrupt the current job and return to the keyboard for operator action.

Format

```
:PAUSE
```

Comments

PAUSE may be entered through the keyboard even when DOS-M is in batch mode. PAUSE suspends the current job until the operator inputs a GO directive. During this time the operator may mount magnetic tapes or prepare I/O devices. (A series of COMMENT directives or a remark in the PAUSE directive itself can be used to tell the operator what to do during the PAUSE.)

The GO directive returns DOS-M to the job in the previous mode.

COMMENT

Purpose

To print a message on the system teleprinter.

Format

:COMMENT *Character String*

where *Character String* is a message to be printed on the teleprinter.

Comments

The programmer may use the COMMENT directive with the PAUSE directive to relay instructions to the operator about setting up magnetic tapes, etc. A space (but not a comma) is required between the directive word and the comment string.

Examples

```
:COMMENT PLACE MAGTAPE LABELED "INPUT" ON THE M.T. UNIT
:COMMENT PUT "INPUT" PAPERTAPE IN PHOTOREADER
```

TYPE

Purpose

To return from batch mode to keyboard mode.

Format

```
:TYPE
```

Comments

Control is returned to the teleprinter keyboard. TYPE may be entered through the batch device or keyboard device; but when it is entered from the keyboard, DOS-M waits until the current executing program is completed or is aborted before returning to keyboard mode. If TYPE is entered while already in keyboard mode, the directive is ignored.

PROG

Purpose

To turn on (i.e., load from the disc and begin executing) a program from the system area or programs from the user file which were generated through the DOS-M Relocating Loader. (Follows the :SS condition in searching for the program.)

Format

:PROG,*name*[,P_1,P_2....P_5]

where *name* denotes a system program, such as FTN for the DOS-M FORTRAN Compiler, ASMB for the DOS-M Assembler, LOADR for the DOS-M Relocating Loader, or ALGOL for the RTE/DOS ALGOL Compiler. A user program is specified via the file name assigned in the DOS-M Relocating Loader.

P_1 through P_5 are optional parameters which DOS-M transfers to the program named. P_1 through P_5 must be positive integers less than 32767. The program must retrieve the parameters immediately. This procedure is described under :GO.

Comment

Consult Section IV for the parameters required by FTN, ASMB, ALGOL, and LOADR. Additional programs may be added at system generation time if desired. (See DOS-M *Generator*, Section VI.)

NOTE: User programs can be run using :PROG. This may be useful when the program needs parameters. DOS-M first searches the user files for the program, then the system files.

Examples

```
:PROG,FTN,2,99
:PROG,ASMB,2,6,4
```

RUN

Purpose

To run a user program. (Follows the :SS condition.)

Format

:RUN,*name*[,*time*][,N]

where *name* is a user file containing the desired program,

time is an integer specifying the maximum number of minutes the program may run (set to five minutes if not specified). DOS-M ignores *time* if a time-base generator is not present.

N, if present, tells DOS-M to allow the program to continue running even if it makes EXEC calls with illegal request codes.

Comments

Programs which have been relocated during the current job but not stored (see STORE directive) permanently in a user file, may be run using this directive. If the program executes longer than the time limit, the current job is aborted and DOS-M scans to the next JOB directive.

If N is not present in the RUN directive, the current job will be aborted by any illegal request codes. The N option is provided so that programs can be written and tested on DOS-M ultimately to execute with other HP software which does not have the same request codes. (See Appendix D, *RELATION TO OTHER SOFTWARE.)*

Example

```
:RUN,ROUT,15
```

executes program ROUT up to fifteen minutes not allowing illegal request codes.

CHANGE USER DISC

Purpose

To change the subchannel assignment for the user disc.

Format

:UD[,[*label*][,*n*]]

where *label* is a six-character disc label (* for an unlabeled disc).

n is the subchannel.

Comments

Discs are labeled by the :IN directive.

Each form of the :UD directive has a different purpose:

Example	Action
:UD (without label or subchannel)	Interrogates the current user disc subchannel and prints its label on the system teleprinter: SUBCHAN = *n* LBL = *label* (or UNLBL)
:UD,,*n* (no label)	If *n* is labeled, DOS-M prints: LBL = *label* (or UNLBL) No assignment is made.
:UD,*label*,*n*	If *n* is labeled with the specified *label*, DOS-M assigns *n* as the user disc. If *n* is unlabeled or has a different *label*, DOS-M prints: LBL = *label* (or UNLBL) Operator can then reissue :UD,*label*,*n* with the correct label.

Example	Action
:UD,*label* (no subchannel)	DOS-M searches for the *label*, starting with the highest number subchannel (determined at system generation). If *label* is found, DOS-M makes it the user disc and prints: SUBCHAN = *n* If *label* is not found, DOS-M prints: DISC NOT ON SYS
:UD,*,*n*	If *n* is unlabeled, DOS-M assigns *n* as the user disc. If *n* is labeled, DOS-M makes no assignment and prints: LBL = *label*
:UD,*	Assigns the highest number unlabeled disc as the user disc and prints: :SUBCHAN = *n* If there are no unlabeled discs, DOS-M prints: DISC NOT ON SYS

If the :UD directive specifies a subchannel with an incorrect system proprietary code (see Appendix J), DOS-M still makes the assignment, and prints:

TSB DISC or ??? DISC

If the :UD directive specifies a subchannel whose system generation code (see Section VI) does not match that of the current system disc, DOS-M still makes the assignment but prints:

DISC GEN CODE *nnnn* NOT SYS GEN CODE *mmmm* ERR POSS

The changes made by :UD are only temporary; the user disc is reset at the end of each job.

DISC-TO-DISC DUMP

Purpose

i. To dump an entire disc onto another subchannel (:DD)

ii. To dump the system area (including system buffer) onto another subchannel (:DD,X)

iii. To dump all or specified files of the user area (optionally assigning some new file names) onto another subchannel (:DD,U...)

Formats

i. :DD

ii. :DD,X

iii. :DD,U[,*file 1*[,(*file A*)],*file 2*[,(*file B*)[,...]

where X specifies the system area,

U specifies the user area,

file 1, file 2, ... specify the files to be dumped (the entire user area if no files are specified),

file A, file B, ... specify the optional new names for *file 1, file 2,* etc. (renamed files can be intermixed with unchanged files).

The destination disc must be specified by a :UD directive immediately following the :DD directive. (For :DD and :DD,X, the directive must be :UD,*,*n* where *n* is not the system disc.)

Comments

When the destination for a :DD,U is a system disc, other than the current system, the user files are dumped in the user area following the system files. This allows the user to dump a system and selected user files to a single disc. (See also :IN.)

The :SS directive does not apply to :DD.

If the files of the source disc cannot completely fit on the destination disc, DOS-M transfers as many whole files as possible and prints

TRAC # TOO BIG

If DOS-M cannot find some of the files specified to be dump, the messages

file
UNDEFINED

is printed. This does not effect dumping of the files which are defined.

If a file specified to be dumped has the same name as an existing file on the destination disc, the message

file
DUPLICATE FILE-NAME

is printed and the file is not dumped. This does not effect dumping of other files.

SYSTEM SEARCH
(Optional Directive)

Purpose

To specify a list of disc subchannels to be searched for file names; the :SS condition applies to all EXEC calls and directives that require a file search. (No check is made for existing duplicate file names during searches; the first file found is used.)

Format

:SS	All active subchannels are searched, starting with the current user subchannel, then continuing from the highest to the lowest number.
:SS,n_1,n_2,n_3....	Where n_1,n_2...are subchannel numbers. The current user subchannel is searched first, then the subchannels specified, starting with the lowest number.
:SS,99	Only the current user subchannel is searched. This is the default condition. Every job starts out in this condition.

Comments

The :SS directive can only be used if it was specifically allowed during system generation. If the operator answers YES to the question

ALLOW :SS?

then :SS directives will be allowed. Otherwise, they are not, and any :SS directive will cause the following message:

BAD CONTROL STATE.

If a file search results in the file being found, the current user subchannel is changed to the subchannel containing the file. If the file was not found, the current user subchannel is restored to its previous assignment. The LIST, U directive is an exception: this directive does not stop after it finds the file; it continues to look for duplicate entries. When the LIST search is complete, the user subchannel is always restored.

However, if a search is interrupted before completion, the current user disc may be on any subchannel. (This should be checked with a :UD directive.)

More than one :SS can occur during a job. The job starts in :SS,99 condition until a different :SS directive is issued. Each :SS directive remains in effect until another is issued. :SS directives do not apply to file searches initiated by the Relocating Loader or to disc dumps initiated by the :DD directive.

Whenever the user subchannel assignment is changed (except by a running program), the system prints a message:

SUBCHAN = *n*

TRACKS

Purpose

To print the next available track on the current user disc.

Format

```
:TRACKS
```

Comments

The number of the first track beyond the end of the current user area, followed by the number of faulty tracks that have been replaced by spares.

Tracks are replaced by spares when parity errors occur on read or write.

Examples

The following is an example in which no faulty tracks are reported.

```
(INPUT)    :TRACKS
(OUTPUT)   NEXT AVAIL TRACK = ØØ1Ø
           @                          (End of directive processing)
```

Examples

The following is an example in which no faulty tracks are reported.

```
(INPUT)     :TRACKS
(OUTPUT)    NEXT AVAIL TRACK = ØØ1Ø
            @                          (End of directive processing)
```

In this example, the system reports that 2 tracks have been replaced by spares.

```
(INPUT)     :TRACKS
(OUTPUT)    NEXT AVAIL TRACK = ØØ12
            BAD = 2
            @                          (End of directive processing)
```

In this example, the system reports that there are no more work tracks available.

```
(INPUT)     :TRACKS
(OUTPUT)    NEXT AVAIL TRACK = NONE
            @                          (End of directive processing)
```

STORE

Purpose

To create a user file on the disc and assign it a name. The STORE directive can create relocatable object program files (type-R), loader-generated object program files (type-P), source statement files (type-S), ASCII data files (type-A), and binary data files (type-B). (Follows :SS in checking for duplicate file names.)

Format

The format varies according to what type file is being created. See Comments below for details:

TYPE-R	:STORE,R,*file*[,*logical unit*]
TYPE-P	:STORE,P[,$name_1$,$name_2$,...]
TYPE-S	:STORE,S,*file*,*logical unit*
TYPE-A	:STORE,A,*file*,*sectors*
TYPE-B	:STORE,B,*file*,*sectors*

NOTE: The second character of file cannot be "Control @."

Comments

TYPE - R FILES

The directive format is:

:STORE,R,*file*[,*logical unit*]

where *file* is a name consisting of five characters or less.

A user file is created under this name, and relocatable binary programs are read into it from the logical unit specified or from the *job binary area* of the work tracks if none is specified. The *job binary area* remains as it was before the STORE directive. (See Section IV, *DOS-M FORTRAN* and *DOS-M ASSEMBLY LANGUAGE*.)

If DOS-M comes to an end-of-tape, it asks:

```
DONE?
```

If there are more tapes, the operator places the next tape in the reader and replies NO; otherwise, he answers YES.

The user should not assign any file names that will be used as program names as this will make loading impossible. The file may be input to the DOS-M Relocating Loader for relocation into an executable program. (See Section IV, *DOS-M RELOCATING LOADER*.)

Examples

```
:STORE,R,RINE
```

(Stores all of the relocatable programs from the job binary area into the file RINE created for that purpose.)

```
:STORE,R,JUGG,5
```

(Stores relocatable programs from logical unit 5, the standard input device, into the file JUGG.)

TYPE - P FILES

The directive format is:

:STORE,P[,*name*$_1$,*name*$_2$,....]

where *name*$_1$,*name*$_2$ ... are programs that the DOS-M Relocating Loader had relocated into executable format during the current job. Up to 14 programs per directive are allowed. If none are specified, all programs loaded during the current job are stored. DOS-M finds these temporary programs in the user file and converts them to permanent user files; the program name automatically becomes the file name.

Programs loaded during the current job but not stored as files (as shown above) may be executed normally (RUN or PROG directive) and appear in the user directory (LIST directive). At the end of a job, however, they are purged from the directory unless they have been converted to user files by a STORE,P directive.

Examples

```
:STORE,P
```

(Changes all programs loaded during the current job using the Relocating Loader into permanent user files.)

```
:STORE,P,ARITH,MATH,TRIG,ALGEB
```

(Searches for the programs listed and makes them permanent user files.)

TYPE - S FILES

The directive format is:

:STORE,S,*file,logical unit*

where *file* is the name of the user file to be filled with source statements from the *logical unit* specified. *File* must not duplicate a name already present in the user or system files. The source statement input must be terminated by a double colon (::). If the :: is omitted, DOS-M stores the succeeding data on the disc as if it were source statements.

If DOS comes to an end-of-tape before finding the ::, it asks

DONE?

If there are more tapes, the operator replies NO: otherwise, he answers YES.

When DOS-M completes the STORE, it prints

nnnn LINES

where *nnnn* is the number of statements stored.

Example

:STORE,S,SOURC,5

(Reads source statements from the standard input device and stores them in a new file SOURC.)

TYPE - A and TYPE - B FILES

The directive format is:

:STORE,*type,file,sectors*

where *type* is either A (for ASCII character data) or B (for binary data), and

file is the name assigned to a file containing the number of *sectors* requested. These requests are made prior to executing a program to reserve a file area; no data is involved. The program may store and retrieve data from the file through a call to EXEC.

It is the programmer's responsibility to store the right kind of data in the file. The EXEC call must specify the file name and the relative sector within the file. DOS-M checks that the file name exists and contains the sector specified.

Example

```
:STORE,A,ASCII,2Ø
```

(Creates a file name ASCII,20 sectors in length. A sector equals 128 words.)

SPECIFY SOURCE FILE

Purpose

To specify the user source file to be used as input by the assembler and compilers. (Follows the :SS condition.)

Format

:JFILE,*file*

where *file* is the name of a TYPE-S file on any active subchannel.

Comments

If logical unit 2 is specified as the input device when the compiler or assembler is turned on (using :PROG) and a :JFILE has been defined, then the compiler or assembler reads the source statements from the :JFILE.

Only one program can be translated from a file; any statements beyond the end of the source program will be ignored. The JFILE assignment is only changed at the end of the current job or by another JFILE directive.

It is highly recommended that the :JFILE directive immediately precede the corresponding :PROG directive.

EDIT

Purpose

To perform listed edit operations on a user source file.

Format

:EDIT,*file*,*logical unit*[,*new file*]

where *file* is the name of a source file (follows the :SS condition) to be edited according to an edit list (edit operations plus associated source statements) input on the specified *logical unit*. If *new file* appears, the edited source file is stored in a new file (with the name *new file*) on the same subchannel and the old file is not purged. Otherwise, the edited source file is the updated old file. (Follows :SS in searching for duplicate file names.)

Position one of a source statement must not be a slash (/) or a colon (:). The legal edit operations in an edit list are described under Comments.

Comments

An edit list consists of several edit operations and, optionally, a series of associated source statements (i.e., following REPLACE, INSERT). Edit operations are executed when they are entered. When using the keyboard, the operator must not enter the next operation until the previous one is completed (completion is signaled by "@" output on the keyboard).

All edit operations begin with a slash (/), and only the first character following the slash is required. The rest are ignored up to a comma. If a colon (:) is encountered in column one before the end of the edit list, the job is aborted. In the edit operation formats, the letters *m* and *n* are the

sequence numbers of the source statements to be edited, starting with one. Letter *m* signifies the starting statement, and *n* is the ending statements of the operation, inclusive. In all cases, *n* must be greater than or equal to *m*; neither can be less than one, nor greater than the last source statement of the file. The *m* must be greater than the *n* of the previous operation.

All edit operations are listed on the system teleprinter as they are executed.

EDIT OPERATIONS

The following operation causes source statements *m* through *n*, inclusive, to be deleted from the file.

/DELETE,*m*[,*n*]

If only *m* is specified, only that one statement will be deleted.

By means of an edit operation, the source statements *m* through *n* can be replaced by one or more source statements following /REPLACE in the edit list.

/REPLACE,m[,*n*]

Again, if *n* is absent, only *m* is replaced.

The format for the INSERT operation is:

/INSERT,*m*

The source statements which follow /INSERT in the edit list are inserted in the file after statement *m*.

In the END operation,

/END

the edit directive is terminated and DOS-M returns to its previous mode for further directives.

Examples

If a file named SOURC contains:

```
Statement 1         ASMB,R,B,L
Statement 2               NAM START
Statement 3         A     EQU 3Ø
Statement 4         B     EQU 2Ø
Statement 5         START NOP
Statement 6               LDA A
Statement 7               END
```

and the EDIT directive is:

```
:EDIT,SOURC,5
```

and the edit list, which follows :EDIT on the batch device, is:

```
/R,3
A     EQU 1ØØ
B     NOP
/D,4
/I,6
      STA B
/E
```

then the new file equals:

```
Statement 1         ASMB,R,B,L
Statement 2               NAM START
Statement 3         A     EQU 1ØØ
Statement 4         B     NOP
Statement 5         START NOP
Statement 6               LDA A
Statement 7               STA B
Statement 8               END
```

PURGE

Purpose

To remove a user file from the user file area.

Format

:PURGE[,*file*$_1$,*file*$_2$,...]

where *file*$_1$,*file*$_2$,... (up to 15 file names or 72 characters per directive) designate files in the user area. These are purged from the user area. If a file cannot be found, a message is printed on the keyboard:

FILE UNDEFINED

If no file names are given, all temporary files are purged.

Comments

Purge follows the :SS condition. After the files are purged from the disc, the remaining user area files are repacked for efficiency. If the end of the user area moves below a track boundary during the purge, the work area becomes a track larger. As each file is purged, DOS-M prints its name on the teleprinter.

IMPORTANT NOTE: A :PURGE must never be interrupted by :AB or :OF because this will cause unpredictable destruction of all or part of the user area.

Example

```
ORIGINAL CONTENTS OF USER FILE:   F1,F2,F3,F4, FLONG, and F5 (at least)
                  DIRECTIVE:      :PURGE,FLONG,F1,F2,D3,D7,F3,F4,F5
                     OUTPUT:      FLONG
                                  F1
                                  F2
                                  D3 UNDEFINED
                                  D7 UNDEFINED
                                  F3
                                  F4
                                  F5
```

The fastest way to purge all files of a single disc is to use :IN,*.

LIST

Purpose

To list file information recorded in the user or system directories. To list and number the contents of a source file sequentially statement-by-statement.

Format

(System) :LIST,X,*logical unit*[,*file*$_1$,...] (Unaffected by :SS)

(User) :LIST,U,*logical unit*[,*file*$_1$,...]

(Lists the specified directory entries from all the subchannels defined by :SS.)

where X specifies the system area directory, and

U specifies a user area directory,

logical unit specifies the list device, and

file$_1$,... names the entries to be listed (if none is specified, the entire directory is listed).

(Source) :LIST,S,*logical unit, file*[,*m*[,*n*]] (follows :SS)

where *file* names the source file to be listed on the *logical unit* specified.

m and *n*, if present, specify the first and last statements to be listed. If *n* is absent, then all statements from *m* on are listed. If neither appear, then the entire field is listed. The restrictions for *m* and *n* are the same as those for the EDIT directive.

Comments

DIRECTORY LISTING OUTPUT

The first line is a heading, identifying the information that follows:

NAME TYPE SCTRS DISC ORG PROG LIMITS B.P.LIMITS ENTRY LIBR. P-BIT

SUBCHAN = *n* (This is printed when :LIST switches to the next subchannel under :SS.)

The following lines are then printed:

name type sctrs trk sec $lower_p$ $upper_p$ $lower_b$ $upper_b$ *entry libr p-bit*

where *name* identifies the file,

type tells what kind of file *name* is,

AD = ASCII data BD = binary data RB = relocatable binary program	User File Only
SS = source statements	
DR = disc resident I/O driver LB = library SR = system core-resident program	System File Only
XS = supervisor module	
UM = user main program US = user program segment	Either File

sctrs is the number of sectors in the file,
trk is the track origin of the file,
sec is the starting sector of the file within the track specified.

The information below does not appear for types AD, BD, LB, RB and SS.

$lower_p$ is the lower limit (octal) of the program,
$upper_p$ is the upper limit (octal) of the program,
$lower_b$ is the lower limit (octal) of the program base page links,
$upper_b$ is the upper limit (octal) of the program base page links,
entry is the absolute octal address where execution begins,
libr is the beginning absolute octal address of the first library routine included in the program, and
p-bit is equal to T if the file is temporary and will be purged by :EJOB unless stored by :ST.

If the requested file does not exist, a message appears,

file UNDEFINED

SOURCE LISTING FORMAT

Each source statement is preceded by a four-digit decimal sequence number. If the requested file is not a source file, a three-line message appears,

```
file
ILLEGAL
RE-ENTER STATEMENT ON TTY
```

The list is terminated by the message

```
**** LIST END ****
```

Examples

(On the keyboard:)

```
:LI,U,6
@
```

(On the list device:)

```
NAME TYPE SCTRS DISC ORG PROG LIMITS   B.P. LIMITS   ENTRY  LIBR. P-BIT
SUBCHAN=4
EX9   SS  ØØ8Ø  TØØ1 ØØØ
EXM   RB  ØØ63  TØØ4 ØØ8
BBB   SS  ØØØ1  TØØ6 Ø23
SRCH  RB  ØØØ3  TØØ7 ØØØ
SSERH UM  ØØØ2  TØØ7 ØØ3 1ØØØØ 1Ø271   ØØ713 ØØ713   1ØØØØ 1Ø271    T
ASCII AD  Ø2ØØ  TØØ7 ØØ5
BINRY BD  Ø3ØØ  TØ15 Ø13
```

NOTE: T on the "P-BIT" column means that the entry is temporary.

(On the keyboard:)

```
:ST,P     (To make all temporary files permanent.)
@
:LI,U,6
@
```

(On the list device:)

```
NAME TYPE SCTRS DISC ORG PROG LIMITS   B.P.LIMITS    ENTRY LIBR. P-BIT
SUBCHAN=4
EX9   SS  ØØ8Ø  TØØ1 ØØØ
EXM   RB  ØØ63  TØØ4 ØØ8
BBB   SS  ØØØ1  TØØ6 Ø23
SRCH  RB  ØØØ3  TØØ7 ØØØ
SSERH UM  ØØØ2  TØØ7 ØØ3 1ØØØØ 1Ø271   ØØ713 ØØ713   1ØØØØ 1Ø271
ASCII AD  Ø2ØØ  TØØ7 ØØ5
BINRY BD  Ø3ØØ  TØ15 Ø13
```

NOTE: "P-BIT" no longer equals "T."

(On the keyboard:)

```
:LI,S,6,EX19,926,936
@
```

(On the list device:)

```
Ø926  ASMB,L,R,X,C,N,B
Ø927        HED DUMMY $LIBR AND $LIBX FOR RTS SIMULATION ON DOS
Ø928        NAM DUMRX,6
Ø929        ENT $LIBR,$LIBX
Ø93Ø        SPC 2
Ø931  *     CALLING SEQUENCES: ENTRY          TERMINATION
Ø932  *
Ø933  *
Ø934  *     PRIVILEGED .       JSB $LIBR      JSB $LIBX
Ø935  *                        NOP            DEF (PROGRAM ENTRY POINT)
Ø936  *
      **** LIST END ****
```

FILE DUMP

Purpose

To dump a user file to a specified peripheral I/O device in a format appropriate to the file content.

Format

:DUMP,*logical unit*,*file*[,*S1*[,*S2*]]

where *logical unit* is the output device to be used for the dump,
file is the user file to be dumped,
S1 and *S2* are the first and last relative sectors to be dumped.

If *S1* and *S2* are not given, the entire file is dumped. If only *S1* is given, then the file, starting with *S1*, is dumped.

Comments

Files may be dumped on list devices or punch devices. The dump format varies with the type of file and the type of device. See Table 2-1.

Table 2-1
FILE DUMP Formats

File Type	Punch Device	List Device
ASCII data	64 characters/record	64 characters/record
Binary data	64 words/record	8 octal words/line
Rel. binary programs	Relocatable binary records (loadable)	8 octal words/line
Source statements	1 statement/record	1 statement/line

Source statements are packed and do not necessarily start on sector boundaries. Thus, if the *S1* and *S2* parameters are used, dumping begins with the start of the first statement beginning in sector *S1*, and ends with the last statement beginning in sector *S2* (this will probably end in the following sector).

Files in the system area cannot be dumped. Errors occur when *S1* > *S2*, or when either *S1* or *S2* is greater than the length of the file.

Examples

Where L is a source file:

```
:DUMP,1,L
A
BB
CCC
DDDD
EEEEE
FFFFFF
GGGGGGG
@
```

Where SSERH is a binary file:

(On the keyboard:)

```
:DU,6,SSERH,1,1
@
```

(On the list device:)

```
ØØ1 ØØØØØØ  Ø62125  Ø72121  114535  Ø1ØØ1Ø  Ø1ØØ75  Ø1Ø156  Ø1Ø1ØØ
    ØØ24ØØ  Ø521ØØ  Ø26Ø14  Ø26Ø36  Ø62ØØ6  Ø42154  Ø72Ø23  114535
    Ø1ØØ25  Ø1ØØ76  Ø1ØØ77  Ø1ØØØ6  Ø1Ø153  114535  Ø1ØØ33  Ø1ØØ76
    Ø1ØØ77  Ø1Ø1Ø1  Ø1Ø117  1Ø25Ø1  ØØ2ØØ2  Ø26Ø56  Ø62ØØ6  Ø72Ø46
    114535  Ø1ØØ5Ø  Ø1Ø123  Ø1ØØ76  Ø1Ø127  Ø1Ø124  Ø1ØØØ6  Ø1Ø122
    114535  Ø1ØØ56  Ø1ØØ76  Ø1ØØ77  Ø1Ø126  Ø1Ø153  Ø36ØØ6  Ø36ØØ6
    Ø36ØØ6  Ø36121  Ø26ØØ3  114535  Ø1ØØ71  Ø1ØØ76  Ø1ØØ77  Ø1Ø1Ø6
    Ø1Ø12Ø  114535  Ø1ØØ74  Ø1ØØ74  ØØØØØ6  ØØØØ22  ØØØØØ2  ØØØØØ1
    ØØØØØØ  Ø2Ø116  Ø4752Ø  Ø2Ø1Ø6  Ø4752Ø  Ø471Ø4  Ø2Ø12Ø  Ø51117
    Ø4352Ø  Ø4Ø515  Ø2Ø1Ø3  Ø4751Ø  Ø5Ø114  Ø4252Ø  Ø425Ø4  ØØØØØ5
    ØØØØ11  ØØØØØØ  ØØØØØØ  ØØØØ16  ØØØØØ2  177746  Ø2ØØ4Ø  Ø2ØØ4Ø
    Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø
    Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø
    Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØØ3  177777  Ø2ØØ4Ø  Ø2Ø5Ø1  Ø4Ø44Ø
    Ø2ØØ4Ø  Ø411Ø2  Ø41Ø4Ø  Ø2ØØ4Ø  Ø415Ø3  Ø4144Ø  Ø2ØØ4Ø  Ø421Ø4
    Ø42Ø4Ø  Ø2ØØ4Ø  Ø425Ø5  Ø4244Ø  Ø2ØØ4Ø  Ø431Ø6  Ø43Ø4Ø  Ø2ØØ4Ø
```

SECTOR DUMP

Purpose

To dump any specified sector or sectors of the current user disc on the standard list device in either ASCII or octal format.

Format

:SA,*track*,*sector*[,*number*] (ASCII)
:SO,*track*,*sector*[,*number*] (Octal)

where *track* and *sector* give the starting disc address for the dump, and

number gives the number of sectors to be dumped. If *number* is absent, only one sector is dumped. All three parameters are decimal numbers.

Comments

The ASCII dump format (:SA) is 64 characters per record. The octal dump format (:SO) is eight octal numbers per line. Two ASCII characters equal one computer word (also represented by one octal number). Although :SA dumps 64 characters per record, these do not necessarily appear on one line since the binary numbers are converted to ASCII characters, some of which might be linefeeds or returns.

Example

(On the keyboard:)

```
:SO,Ø,1
@
```

(On the list device:)

```
ØØ1 ØØØØØØ  Ø67767  Ø1757Ø  Ø67744  Ø77743  Ø17613  Ø17613  Ø17613
    Ø17613  Ø6412Ø  ØØ7ØØ4  Ø7731Ø  Ø64117  Ø44Ø55  16ØØØ1  Ø44Ø51
    Ø1ØØ72  Ø73773  Ø53774  Ø77761  Ø53775  Ø77762  Ø773Ø4  Ø44Ø56
    16ØØØ1  ØØ1727  Ø13733  Ø733Ø5  Ø5ØØ6Ø  Ø2746Ø  Ø53763  Ø27445
    Ø673Ø4  Ø44Ø66  Ø3731Ø  Ø27415  Ø275Ø5  Ø44Ø52  16ØØØ1  Ø23773
    Ø33774  17ØØØ1  Ø63773  Ø733Ø2  ØØ2ØØ4  Ø733Ø3  Ø63774  Ø73773
    Ø673Ø4  16ØØØ1  Ø73766  164ØØØ  Ø1757Ø  Ø633Ø5  Ø5ØØ6Ø  Ø2744Ø
    ØØ6ØØ4  16ØØØ1  Ø33773  17ØØØ1  ØØ6ØØ4  Ø6373Ø  17ØØØ1  ØØ6ØØ4
    ØØ3ØØ4  17ØØØ1  Ø673Ø4  Ø77311  Ø2744Ø  Ø6Ø154  ØØ1722  Ø13765
    Ø33774  ØØ1727  ØØ1723  Ø7Ø154  Ø63761  Ø673Ø2  Ø176Ø6  Ø6376Ø
    Ø673Ø3  Ø176Ø6  ØØ24ØØ  Ø67774  Ø176Ø6  Ø6331Ø  Ø67775  Ø176Ø6
    Ø6776Ø  ØØ6ØØ3  Ø2754Ø  Ø44Ø55  16ØØØ1  Ø23774  Ø333Ø2  17ØØØ1
    Ø67762  ØØ6ØØ3  Ø27546  Ø23775  Ø333Ø3  17ØØØ1  Ø63776  ØØ12ØØ
    Ø6777Ø  ØØ6ØØ3  ØØ2ØØ4  Ø6415Ø  Ø7Ø155  Ø5417Ø  Ø7Ø175  ØØ64ØØ
    Ø5Ø175  Ø6411Ø  Ø742ØØ  Ø4774Ø  Ø74157  Ø6417Ø  Ø7416Ø  124ØØ3
    ØØØØØØ  Ø57766  12757Ø  Ø37766  163766  ØØ2Ø21  Ø27571  Ø13764
```

PROGRAM DUMP

Purpose

To request that a user program be dumped when it completes execution. Two directives are provided: PDUMP for dumping on a normal completion, and ADUMP for dumping when the program aborts.

Format

:PDUMP[,*FWA*[,*LWA*]][,B][,L]
:ADUMP[,*FWA*[,*LWA*]][,B][,L]

where *FWA* is the first word address, relative to the program origin,

B means dump the base page linkage area of the program, and,

L means dump the library subroutines used by the program.

FWA and *LWA* are octal numbers that specify the limits of the program being dumped.

If *LWA* is missing, the entire program, starting with *FWA*, is dumped.

B alone dumps all the main program, pluss base page linkages, but not the library routines.

L alone dumps only the library routines.

If no parameters are given, everything is dumped

Comments

Any parameter following L is ignored. If FWA is greater than LWA, a message is printed. When the directive :PDUMP precedes a :RUN or :PROG request, the program contained in the request will be dumped, if it runs to normal completion. To dump a program that is aborted while running, the directive :ADUMP must precede the :RUN request. To make sure that a program will be dumped whether it runs normally or is aborted, both dump directives must be declared preceding the :RUN request. Only one of the requests will be honored, depending upon whether the program runs normally or is aborted.

Since DOS-M sets a flag when it encounters either dump directive then clears the flag after the dump routine is executed, the flag representing the executed dump routine will remain set. This flag can cause an unwanted dump of some program run later under the same :JOB directive. Either dump flag can be cleared by requesting the dump with both FWA and LWA equal to 0. All flags can be cleared by calling a new :JOB directive, or giving an :OF when no programs are running.

The main program and library subroutines are dumped eight octal words per line, along with the octal starting address for that line. For example,

adr$_8$	*wd-1*	*wd-2*	*wd-3*	*wd-4*	*wd-5*	*wd-6*	*wd-7*	*wd-8*
ad$_8$*+10*$_8$	*wd-1*	*wd-2*	*wd-3*	*wd-4*	*wd-5*	*wd-6*	*wd-7*	*wd-8*

If present, the base page dump follows the main program and library. Base page linkages exist for page boundary crossings and subroutines. For each line, the starting address appears first, followed by four pairs of octal numbers. The first number of each pair records the content of the base page word (an address elsewhere in core). The second number of each pair records the contents of the address specified by the first item. If the first item is the address of a subroutine, then the second item contains the last address from which the subroutine was called. For example,

	pair-1		*pair-2*		*pair-3*		*pair-4*	
adr	*item-1*	*item-2*	*item-1*	*item-2*	*item-1*	*item-2*	*item-1*	*item-2*
adr+4$_8$	*item-1*	*item-2*	*item-1*	*item-2*	*item-1*	*item-2*	*item-1*	*item-2*

Example

```
:ADUMP,Ø15,B        (Set up dump flag)
:PR,LOADR           (Run program)
LU    Ø12140
ABRT  Ø12140        (Program aborted)
(Page Eject)
```

(Main program dump)

```
12ØØØ  16ØØØ1  ØØ2ØØ2  13Ø573  17Ø574  ØØ6ØØ4  16ØØØ1  ØØ2ØØ3  Ø26Ø12
12Ø1Ø  13Ø575  17Ø576  ØØ6ØØ4  16ØØØ1  17Ø577  ØØ6ØØ4  16ØØØ1  17Ø6ØØ
```

(Page Eject)

(Base page dump)

```
ØØ57Ø  Ø1Ø137  ØØ2Ø45  Ø1Ø711  ØØ3237  Ø1Ø763  ØØ2Ø45  Ø17Ø14  ØØØ3ØØ
ØØ574  Ø17641  ØØØØØØ  Ø17Ø15  ØØØ4ØØ  Ø17641  ØØØ4Ø6  Ø176Ø1  ØØØØØØ
ØØ6ØØ  Ø1765Ø  ØØØØØØ  Ø17615  ØØØØØØ  Ø17664  ØØØØØØ  Ø17662  ØØØ573
ØØ6Ø4  Ø17637  ØØØ573  Ø17571  177Ø5   Ø17563  ØØ1Ø4   Ø17714  Ø17715
ØØ61Ø  Ø17562  Ø21121  Ø17534  Ø21122  Ø17536  Ø21122  Ø17633  16Ø656
ØØ614  Ø17544  Ø37626  Ø17546  Ø37626  Ø17673  ØØØØØØ  Ø176Ø5  ØØØØ4Ø
```

EQUIPMENT

Purpose

To list one or all entries in the equipment table.

Format

:EQ[,*n*]

where *n*, if present, indicates the one entry to be listed. If *n* is absent, the entire equipment table is listed.

Comments

Each entry is output in the following format:

EQT *nn* CH *vv* DVR*mm* *d* *r* U*u* S*s*

where *nn* is the decimal number of the entry,
vv is the octal channel number of the device,
DVR*mm* is the I/O driver number for the device,
d specifies DMA if equal to D, no DMA if Ø,
r specifies core-resident if equal to R, disc-resident if Ø,
u is one decimal digit used for subchannel addressing,
s is the availability status of the device:
Ø for not busy, and available,
1 for disabled (down),
2 for busy,
3 for awaiting an available DMA channel.

Example

```
:EQ
EQT Ø1 CH 1Ø DVR31 D R UØ SØ
EQT Ø2 CH 12 DVR22 D Ø UØ SØ
EQT Ø3 CH 14 DVRØ5 Ø R UØ SØ
EQT Ø4 CH 15 DVRØ1 Ø Ø UØ SØ
EQT Ø5 CH 16 DVRØ2 Ø Ø UØ SØ
EQT Ø6 CH 17 DVR12 Ø Ø UØ SØ
EQT Ø7 CH 21 DVR15 D Ø UØ SØ
@
```

LOGICAL UNIT

Purpose

To assign logical unit numbers (4 through 63) for a job or to list the device reference table (logical unit assignments).

Format

:LU[,n_1[,n_2]]

where n_1 and n_2, if both present, assign the device recorded in equipment table entry n_2 to logical unit number n_1 (both are decimal numbers). If only n_1 is present, then the equipment table entry number (see *EQUIPMENT* directive) assigned to logical unit number n_1 is output. If no parameters appear, the entire device reference table is printed.

Comments

Assignments made by :LU for logical units 4 through 9 are only valid during the current job. Assignments for 10 and above remain after EJOB. At the beginning of each new job, the device reference table for the first nine logical units is reset to the assignments given when the system was configured. (See Section VI, *DOS-M Generator*.) This insures a standard I/O organization for all users.

Example

```
:LU
LUØ1 EQTØ3
LUØ2 EQTØ1
LUØ3 EQTØ1
LUØ4 EQTØ5
LUØ5 EQTØ4
LUØ6 EQTØ6
LUØ7 EQTØ7
LUØ8 EQTØ2
@
```

UP

Purpose

To declare an I/O device ready for use.

Format

:UP,*n*

where *n* is the equipment table entry number corresponding to the device.

Comments

The :UP directive (followed by a :GO) is usually used in response to the following messages from DOS-M:

I/O ERR ET EQT #*n*
I/O ERR NR EQT #*n*
I/O ERR PE EQT #*n*

where ET indicates end of tape,
NR indicates device not ready,
PE indicates parity error, and
n is the equipment entry number.

DOWN

Purpose

To declare an I/O device unavailable for use.

Format

:DN,*n*

where *n* is the equipment table entry number for the device to be set down.

Comments

The system teleprinter and the disc (logical units 1,2, and 3) cannot be set down. Once set down, a device is unavailable until set UP by the operator.

BATCH

Purpose

To switch from keyboard mode to batch mode or to reassign the batch device.

Format

:BATCH,*logical unit*

where *logical unit* is the device to be used as the batch input device.

Comments

See "TYPE" in this section for the opposite procedure of returning from batch mode to keyboard mode.

NOTE:

The directives in the rest of this section pertain to operation in the keyboard mode only.

DATE

Purpose

To set the date and time for accounting purposes whenever DOS-M is started up.

Format

:DATE,*day*[,*hour*,*min*]

where *day* is any string of ten or less characters (commas not permitted) chosen by the operator (such as 7/1Ø/69,1Ø.JULY.69, etc.);

hour and *min* are the current time in hours and minutes on a 24-hour clock. If not given or time-base generator is not present, they are set to zero.

Comments

The DATE directive is legal only following a start-up procedure. (See Section VI, *DOS-M INITIATION FROM THE DISC*.) The directive is not accepted any other time.

Examples

```
:DATE,7/1Ø/69,12,23
:DATE,WEDNESDAY,7,45
:DATE,1ØJULY1969
```

GO

Purpose

To resume a program that has been suspended, and optionally, to transfer up to five parameters to that program.

Format

$$:GO[,P_1,P_2,...P_5]$$

where P_1 through P_5 are optional parameters and must be decimal values between Ø and 32767.

Comments

When a program suspends itself (see Section III, *PROGRAM SUSPEND EXEC CALL*), it is restarted by a GO directive. Upon return to a suspended program, the initial address of the five parameters is located in the B-register. A FORTRAN program calls the library subroutine RMPAR to transfer the parameters to a specified 5-word array. The first statement after the suspend call, in a FORTRAN program, must be the call to RMPAR. For example,

```
DIMENSION I(5)
CALL RMPAR (I)
```

An assembly language program should use the B-register upon return from the suspend to obtain and save the parameters prior to making any EXEC request or I/O request.

INITIALIZE

Purpose

To label or unlabel the current user disc.

Format

:IN,*label*

where *label* is a six-character name to be written on the disc or "*" which means unlabel the disc. (The *label* cannot start with a "Control @.")

Comments

If the user disc is already labeled, DOS-M prints:

TSB (or ???) LABEL *nnnnnn* (*nnnnnn* is existing label)
OK TO PURGE

The operator must respond with

YES

to actually execute the directive, or

NO

to leave the disc unchanged.

If the label equals "*", the user files also are purged.

If the current user disc is labeled SYSTEM and is not hardware protected (which means it was created by a :DD,X), the system area is destroyed and any files are moved down to low disc.

Labeling a disc eliminates any old label on the disc but does not eliminate the directory or files on the disc.

Unlabeling a disc also eliminates the contents.

:IN always changes the system generation code and system proprietary code to that of the current system. :IN can prepare discs for use by DOS-M that were formatted by a diagnostic or other software.

OFF

Purpose

To abort the currently executing user program of system operation without terminating the job.

Format

:OFF

Comments

:OFF returns the system to keyboard mode.

OFF can be used to terminate undesired lists, edits, disc-to-disc dumps, program loops, loader operations, assemblies, and compilations.

:OFF must never be given during a :PURGE or after /E in an EDIT.

:OFF cancels any pending :DD, :AD, or :PD directives, unless a program is running, in which case, a pending :ADUMP is executed.

SECTION III
EXEC CALLS

Using EXEC calls, which are the line of communication between an executing program and DOS-M, a program is able to:

- Perform input and output operations,
- Request status of I/O devices
- Determine availability of work area tracks,
- Terminate or suspend itself,
- Load its segments,
- Search for file names,
- Obtain the time of day, or
- Change the user disc subchannel.

An EXEC call is a block of words, consisting of an executable instruction and a list of parameters defining the request. The execution of the instruction transfers control to DOS-M. DOS-M then determines the type of request (from the parameter list) and, if it is legally specified, initiates processing of the request. The executable instruction is a jump subroutine (JSB) to EXEC.

In FORTRAN, EXEC calls are coded as CALL statements. In ALGOL, procedure calls are used. In Assembly Language, EXEC calls are coded as a JSB, followed by a series of parameter definitions. For any particular call, the object code generated for the FORTRAN CALL Statement and the ALGOL procedure call is equivalent to the corresponding Assembly Language object code.

This section describes the basic formats of FORTRAN, ALGOL and Assembly Language EXEC calls, then each EXEC call is presented in detail.

FORMAT OF THE ASSEMBLY LANGUAGE CALLING SEQUENCE

The following is a general model of an EXEC call in Assembly Language:

```
        EXT EXEC              (Used to link program to DOS-M)
         :
        JSB EXEC              (Transfer control to DOS-M)
        DEF *+n+1             (Defines point of return from DOS-M, n
                               is number of parameters; may not be an
                               indirect address; must be the location
                               immediately following the last parameter
                               address)
        DEF P1    )
         :        }           (Define addresses of parameters which
        DEF Pn    )           may occur anywhere in program; may be
                               multi-level indirect)
        return point          (Continue execution of program)
         .
         :
         .
         :
   P1 ---
    .          )
    .          }              (Actual parameter values)
    .          )
   Pn ---
```

EXEC CALLS IN ALGOL

In ALGOL, certain conventions must be followed in making EXEC calls. First, since EXEC is external to the program it must be declared a CODE procedure. Second, parameters that are going to be changed must be declared "name" and those that are not to change must be VALUE parameters. Third, since ALGOL requires that the format of each procedure call be defined, a program must declare a dummy external procedure for each type of EXEC call it makes. (These dummy procedures must be compiled as separate procedures to provide proper linkage in the Loader.)

For example,

(In the main program):

```
        :
        PROCEDURE EXECA(A); INTEGER A;CODE;
        PROCEDURE EXECB(A,B,C,D);INTEGER A,B,C,D;CODE;
        :
        EXECA(I);
        :
        EXECB(J,K,L,M);
        :
        END$
```

(External)

```
        HPAL,P,"EXECA",B,L
        PROCEDURE EXECA(A); INTEGER A; BEGIN PROCEDURE EXEC(A); INTEGER A;
        CODE; EXEC(A);
        END$
```

FORMAT OF THE FORTRAN CALLING SEQUENCE

In FORTRAN, the EXEC call consists of a CALL Statement and a series of assignment statements defining the variable parameters of the call:

CALL EXEC (P_1, P_2 .... , P_n)

where P_1 through P_n are either values or variables defined elsewhere in the program. Variables must begin with a letter I through N, since they are integer variables.

Example

```
CALL EXEC (7)
    or
IRCDE = 7
CALL EXEC (IRCDE)
```

Equivalent calling sequence

Some EXEC call functions are handled automatically by the FORTRAN compiler or special subroutines. (Refer to "FORTRAN," Section IV, DOS-M PROGRAMMING, and the specific EXEC calls in this section.)

READ/WRITE

Purpose

To transfer information to or from an external I/O device or the work area of the disc. (DOS-M handles track switching automatically.)

Assembly Language

```
        EXT     EXEC
         :
        JSB     EXEC            (Transfer control to DOS-M)
        DEF     *+5 (or 7)      (Point of return from DOS-M; 7 is
                                for disc request)
        DEF     RCODE           (Request code)
        DEF     CONWD           (Control information)
        DEF     BUFFER          (Buffer location)
        DEF     BUFFL           (Buffer length)
        DEF     DTRAK           (Track number-disc transfer only)
        DEF     DSECT           (Sector number-disc transfer only)
        (return point)          (Continue execution)
         :
RCODE   DEC     1 (or 2)        (1=READ, 2=WRITE)
CONWD   OCT     conwd           (conwd is described in Comments)
BUFFR   BSS     n               (Buffer of n words)
BUFFL   DEC     n (or -2n)      (Same n; words (+) or characters (-))
DTRAK   DEC     f               (Work area track number, decimal)
DSEC    DEC     g               (Work area sector number, decimal)
```

FORTRAN

I/O transfers to regular devices are programmed by standard FORTRAN READ and WRITE Statements. I/O on the work area of the disc is done with a subroutine BINRY, described in the Comments, or the FORTRAN equivalent of the EXEC call:

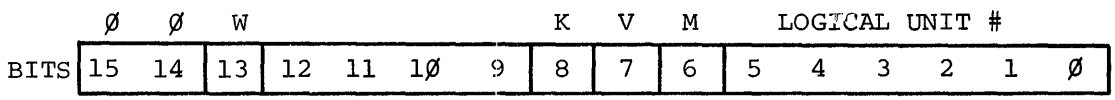
CALL EXEC (ICODE, ICON, IBUF, IBUFL, ITRAK, ISECT)

Comments

READ/WRITE EXEC calls carry out I/O transfers including those on the work area of the disc. (See FILE READ/WRITE EXEC CALL.)

CONWD

The conwd, required in the calling sequence, contains the following fields:



Field	Function
W	If 1, tells DOS-M to return to the calling program after starting the I/O transfer. If W = Ø, DOS-M waits until the transfer is complete before returning.
K	Used with keyboard input, specifies printing the input as received if K = 1. If K = Ø, "no printing" is specified.
V	Used when reading variable length records from punched tape devices in binary format (M = 1, below). If V = Ø, the record length is determined by buffer length. If V = 1, the record length is determined by the word count in the first non-zero character which is read in.
M	Determines the mode of data transfer. If M = Ø, transfer is in ASCII character format, and if M = 1, binary format. (Disc is always binary.)

BINRY

User FORTRAN programs call the FORTRAN disc read/write library routine, BINRY, to accomplish I/O in the work area. The user must specify: an array to be used as a buffer, the length of the buffer in words (equal to the number of elements in an integer array, double that for a real array), the disc logical unit, track number, sector number, and offset in words within the sector. (If the offset equals Ø, the transfer begins on the sector boundary. If the offset equals N, then transfer skips N words of the sector before starting). BINRY has two entry points, BREAD and BWRIT, for read and write operations respectively. An example below gives the calling procedure.

```
      DIMENSION  IBUF(1Ø), BUF(2Ø)
      LUN = 2
      ITRK = 12
      ISECT = 63
      IOFF = Ø
      CALL BREAD (BUF, 4Ø, LUN, ITRK, ISECT, IOFF),
or
      CALL BWRIT (IBUF, 1Ø, LUN, ITRK, ISECT, IOFF)
```

Waiting and No Waiting

If the program requests the *no waiting* option in the *conwd*, it can check for the end of the I/O operation with the I/O STATUS EXEC call. In the Assembly Language calling sequence, the buffer length can be given in words (+) or characters (-). When the transfer is complete, the amount actually transferred can be learned by the same status call. A positive number of words or characters, depending upon which were originally requested, is returned. If the WAIT option is used, DOS-M returns the number of transmitted words or characters to the B register.

FILE READ/WRITE

Purpose

To transfer information to or from a file on the user disc; the file must be referenced by name. (The :SS condition is followed.)

Assembly Language

```
        EXT EXEC
         :
        JSB EXEC            (Transfer control to DOS-M)
        DEF *+7             (Point of return from DOS-M)
        DEF RCODE           (Request code)
        DEF CONWD           (Control information)
        DEF BUFFER          (Buffer location)
        DEF BUFFL           (Buffer length)
        DEF FNAME           (File name)
        DEF RSECT           (Relative sector within file)
        return point        (Continue execution)
         :
RCODE   DEC 14 or 15        (14 = READ, 15 = WRITE)
CONWD   OCT conwd           (See Comments, READ/WRITE EXEC CALL.)
BUFFR   BSS n               (Buffer of n words)
BUFFL   DEC n or -2n        (Same n; words (+) or characters (-))
FNAME   ASC 3,xxxxx         (User file name = xxxxx)
RSECT   DEC m               (Relative sector number )
```

FORTRAN

```
DIMENSION IFILE (3)
IFILE(1) = xxxxxB          (First two characters of file name)
IFILE(2) = xxxxxB          (Second two characters)
IFILE(3) = xxxxxB          (Last character and blank)
IRCD = 14 (or 15)          (Request code)
ICNWD = xxxxxB             (conwd)
DIMENSION IBUF(10)
CALL EXEC (IRCDE, ICNWD, IBUF, 10, IFILE, 0)
```

Comments

See the Comments under READ/WRITE EXEC CALL for a description of the *conwd* fields needed in the above calling sequences.

To read or write on the first sector of a file, m=Ø, for the last sector, m=number of sectors in the file -1. To determine the size of a file, use the SEARCH FILE NAMES EXEC call.

Any type of file may be read, but only ASCII or binary data files may be written.

If the DOS-M installation is likely to have more than one user disc, the program should use the CHANGE USER DISC EXEC call without a subchannel specified to check whether the correct user disc is currently assigned. Alternatively, the user can use an :SS directive to set up a system search condition for referencing files on many subchannels.

I/O CONTROL

Purpose

To carry out various I/O control operations, such as backspace, write end-of-file, rewind, etc.

Assembly Language

```
        EXT EXEC
         :
        JSB EXEC          (Transfer control to DOS-M)
        DEF *+4(or 3)     (Point of return from DOS-M)
        DEF RCODE         (Request code)
        DEF CONWD         (Control information)
        DEF PARAM         (Optional parameter)
        return point      (Continue execution)
         :
RCODE   DEC 3             (Request code = 3)
CONWD   OCT conwd         (See Comments)
PARAM   DEC n             (Required for some control functions;
                           see Comments)
```

FORTRAN

Use the FORTRAN auxiliary I/O statements or an EXEC calling sequence.

```
IRCDE = 3                 (Request code)
ICNWD = conwd             (See Comments)
IPRAM = x                 (Optional; see Comments)
CALL EXEC (IRCDE, ICNWD, IPRAM)
CALL EXEC (IRCDE, ICNWD)
```

Comments

CONWD

The control word value (*conwd*) has two fields:

	Ø	Ø	W	FUNCTION CODE (see below)						LOGICAL UNIT NUMBER					
BITS	15	14	13	12	11	1Ø	9	8	7	6	5	4	3	2	1	Ø

If W = 1, DOS-M returns to the calling program after starting the control request.

If W = Ø, DOS-M waits until the control request is complete before returning.

Function Code (Octal)	Action
ØØØ	Unused
ØØ1	Write end-of-file (magnetic tape)
ØØ2	Backspace one record (magnetic tape)
ØØ3	Forward space one record (magnetic tape)
ØØ4	Rewind (magnetic tape)
ØØ5	Rewind standby (magnetic tape)
ØØ6	Dynamic status (magnetic tape)
ØØ7	Set end-of-paper tape
Ø1Ø	Generate paper tape leader
Ø11	List output line spacing (PARAM or IPRMA required)
Ø12 : 177	Unused

Function code 11_8, list output line spacing, requires the optional parameter mentioned in the calling sequences. PARAM (or IPRAM) designates the number of lines to be spaced on the specified logical unit. A negative parameter specifies a page eject on a line printer or number of lines to be spaced on the teleprinter. For details of line printer formatting, consult Appendix E.

I/O STATUS

Purpose

To request the status of a particular I/O device, and the amount transmitted in the last operation.

Assembly Language

```
      EXT EXEC
       :
      JSB EXEC          (Transfer control to DOS-M)
      DEF *+5           (Point of return from DOS-M)
      DEF RCODE         (Request code)
      DEF CONWD         (Logical unit)
      DEF STATS         (Status returned)
      DEF TLOG          (Transmission log returned)
      return point      (Continue execution)
       :
RCODE DEC 13            (Request code - 13)
CONWD DEC n             (Logical unit number)
STATS NOP               (Status returned here)
TLOG NOP                (Transmission log returned here)
```

FORTRAN

```
      IRCDE = 13        (Request code)
      ICNWD = n         (n is decimal logical unit)
      CALL EXEC (IRCDE, ICNWD, ISTAT, ITLOG)
```

Comments

The status returned in the A-register and in STATS is the hardware status of the device specified by the logical unit. The transmission log in the B-register and in TLOG contains the amount of information which was transferred (a positive number of words or characters depending on which was requested by the call initiating the transfer).

WORK AREA LIMITS

Purpose

To ascertain the first and last tracks of the work area on the system disc and the number of sectors per track.

Assembly Language

```
      EXT EXEC
       :
      JSB EXEC          (Transfer control to DOS-M)
      DEF *+5           (Point of return from DOS-M)
      DEF RCODE         (Request code)
      DEF FTRAK         (First track)
      DEF LTRAK         (Last track)
      DEF SIZE          (Number of sectors/track)
      return point      (Continue execution)
       :
RCODE DEC 17            (Request code = 17
FTRAK NOP               (Returns first work track number here)
LTRAK NOP               (Returns last work track number here)
SIZE  NOP               (Returns number of sectors per track here)
```

FORTRAN

```
      IRCDE = 17        (Request code)
      CALL EXEC (IRCDE, IFTRK, ILTRK, ISIZE)
```

Comments

This call returns the limits of the work area, that area of the system disc which programs use for temporary storage with the READ/WRITE EXEC call.

WORK AREA STATUS

Purpose

To ascertain whether a specified number of consecutive operable tracks exist in the work area of the disc.

Assembly Language

```
        EXT EXEC
         :
        JSB EXEC          (Transfer control to DOS-M)
        DEF *+5           (Point of return from DOS-M)
        DEF RCODE         (Request code)
        DEF NTRAK         (Number of tracks desired)
        DEF RTACK         (Starting track desired)
        DEF STRAK         (Actual starting track)
        return point      (Continue execution)
         :
RCODE   DEC 16            (Request code = 16)
NTRAK   DEC n             (Consecutive tracks desired)
TRACK   NOP               (Desired track; from LIMITS call)
STRAK   NOP               (Actual starting track available,
                          Ø if n tracks not available)
```

FORTRAN

```
        IRCDE = 16        (Request code)
        ICNWD = n         (Consecutive tracks desired)
        ITRAK = m         (Desired starting track)
        CALL EXEC (IRCDE, ICNWD, ITRAK, ISTRK)
```

Comments

This call is used with the WORK AREA LIMITS EXEC call to establish the nature of the work area. The READ/WRITE EXEC call then transmits information to and from this area, using the track numbers determined by this call. DOS-M handles track switching automatically. When the disc would overflow, DOS-M prints this message:

```
TRAC # TOO BIG
```

DOS-M checks whether there are *n* consecutive tracks starting at the track specified. Upon location of tracks, DOS-M returns the starting track number to the program. If DOS-M does not locate *n* consecutive tracks, it returns Ø in TRAK or ITRAK.

PROGRAM COMPLETION

Purpose

To notify DOS-M that the calling program is finished and wishes to terminate.

Assembly Language

```
        EXT EXEC
         :
        JSB EXEC          (Transfer control to DOS-M)
        DEF *+2           (Return point from DOS-M)
        DEF RCODE         (Request code)
        return point
         :
RCODE   DEC 6             (Request code = 6)
```

FORTRAN

The FORTRAN and ALGOL compilers generate a PROGRAM COMPLETION EXEC CALL automatically when they compile an END or STOP statement.

PROGRAM SUSPEND

Purpose

To suspend the calling program from execution until restarted by the GO directive.

Assembly Language

```
        EXT EXEC
         :
        JSB EXEC          (Transfer control to DOS-M)
        DEF *+2           (Point of return from DOS-M)
        DEF RCODE         (Request code)
        return point      (Continue execution)
         :
RCODE   DEC 7             (Request Code = 7)
```

FORTRAN

The library subroutine PAUSE, which is automatically called by a PAUSE statement, generates the SUSPEND EXEC call.

Comments

DOS-M prints a message on the system teleprinter when it processes the PROGRAM SUSPEND EXEC call:

name SUSP

When the operator restarts the program with a GO, the B-Register contains the address of a five-word parameter array set by the GO request. (The parameters equal zero if no values have been given.) In a FORTRAN program, the library subroutine RMPAR can load these parameters; however, the call to RMPAR must occur immediately following the SUSPEND EXEC call, as in the following example:

```
DIMENSION I (5)
CALL EXEC (7)          (Suspend)
CALL RMPAR (I)         (Return point; get parameters)
```

PROGRAM SEGMENT LOAD

Purpose

To load a segment of the calling program from the disc into the segment overlay area and transfer execution control to the segment's entry point. (See Section IV, DOS-M PROGRAMMING, for information on segmented programs.) Follows the :SS condition.

Assembly Language

```
        EXT EXEC
         :
        JSB EXEC        (Transfer control to DOS-M)
        DEF *+3         (Point of return from DOS-M)
        DEF RCODE       (Request code)
        DEF SNAME       (Segment name)
        return point    (Continue execution)
         :
RCODE   DEC 8           (Request code = 8)
SNAME   ASC 3,xxxxx     (xxxxx is the segment name)
```

FORTRAN

```
      IRCDE = 8
      DIMENSION INAME 93)
      INAME (1) = xxxxxB        (First two characters)
      INAME (2) = xxxxxB        (Second two)
      INAME (3) = xxxxxB        (Last character)
      CALL EXEC (IRCDE, INAME)
```

Comments

In the FORTRAN or ALGOL calling sequence, the name of the segment must be converted from ASCII to octal and stored in the INAME array, two characters per word.

See OVERLAY SEGMENTS and SEGMENTED PROGRAMS, Section IV, for a description of segmented programs.

SEARCH FILE NAME

Purpose

To check whether a specific file name exists in the directory of user or system files. (Follows the :SS condition.)

Assembly Language

```
        EXT EXEC
         :
        JSB EXEC          (Transfer control to DOS-M)
        DEF *+4           (Return address)
        DEF RCODE         (Request code)
        DEF FNAME         (File name)
        DEF NSECT         (Number of sectors)
        return point
         :
RCODE   DEC 18            (Request code = 18)
FNAME   ASC 3,xxxxx       (xxxxx is the file name)
NSECT   NOP               (Number of sectors returned here;
                          Ø if not found)
```

FORTRAN

```
        IRCDE = 18                  (request code)
        DIMENSION INAME (3)         (File name)
        INAME (1) = xxxxxB          (First two characters)
        INAME (2) = xxxxxB          (Next two characters)
        INAME (3) = xxxxxB          (Last character and blank)
        CALL EXEC (IRCDE, INAME, ISECT)
```

TIME REQUEST

Purpose

To request the current time.

Assembly Language

```
        EXT EXEC
         :
        JSB EXEC          (Transfer control to DOS-M)
        DEF *+3           (Point of return from DOS-M)
        DEF RCODE         (Request code)
        DEF ARRAY         (Time value array)
        return point      (Continue execution)
         :
RCODE   DEC 11            (Request code = 11)
ARRAY   BSS 5             (Time value array)
```

FORTRAN

```
        IRCDE = 11
        DIMENSION ITIME (5)
        CALL EXEC (IRCDE, ITIME)
```

Comments

When DOS-M returns, the time value array contains the time on a 24-hour clock:

ARRAY	or ITIME (1)	=	Tens of milliseconds
ARRAY + 1	or ITIME (2)	=	Seconds
ARRAY + 2	or ITIME (3)	=	Minutes
ARRAY + 3	or ITIME (4)	=	Hours
ARRAY + 4	or ITIME (5)	=	Not used, but must be present (always = Ø)

If DOS-M does not contain a time base generator, all values in the time array are set to zero (0).

CHANGE USER DISC

Purpose

To change the subchannel assignment for the user disc.

Assembly Language

```
        EXT EXEC
         :
        JSB EXEC                (Transfer control to DOS-M)
        DEF *+3 (or 4)          (Point of return from DOS-M)
        DEF RCODE               (Request code)
        DEF LABEL               (Disc Label)
        DEF SUBCH               (Disc Subchannel; optional)
        return point
         :
RCODE   DEC 23                  (Request code = 23)
LABEL   ASC 3, xxxxxx           (Label = xxxxxx)
SUBCH   DEC (Ø to 7)
```

FORTRAN

```
        IRCDE = 23
        DIMENSION LABEL (3)
        LABEL (1) = xx
        LABEL (2) = xx
        LABEL (3) = xx
        ICHNL = M          (Ø through 7)
        CALL EXEC (IRCDE, LABEL, ICHNL)
```

Comments

1. If both the label and subchannel are specified, DOS-M checks whether the subchannel has that label. If it does, the assignment is made and DOS-M returns. If not, DOS-M prints:

```
LBL = name      (name is label on the subchannel)
or UNLBL
UD nnnnn        (nnnnn = address of EXEC call)
xxxxx SUSP      (xxxxx = name of program)
```

   The operator can load a correctly labeled disc on the subchannel and type in

```
:GO
```

   This returns to the *beginning* of the EXEC call (not the normal return point) so that the program can reissue the EXEC call.

   If the operator does not have a properly labeled disc (or the subchannel is a permanent disc), he should use "OFF or :ABORT.

2. If only a label is specified, DOS-M searches for the label, starting with the highest subchannel. If DOS-M finds the label, it makes the assignment.

   If DOS-M cannot find the label, it suspends the program and prints:

```
DISC NOT ON SYS
UD nnnnn
xxxxx SUSP
```

   The operator can then abort the program or load a properly labeled disc and type in:

```
:GO
```

   This returns to the *beginning* of the EXEC call.

3. If the label equals "*" and a subchannel is specified, DOS-M checks whether the subchannel is unlabeled. If it is, DOS-M makes the assignment. If the subchannel is labeled, DOS-M suspends the program and prints:

   LBL = *name*
   UD *nnnnn*
   xxxxx SUSP (*xxxxx* is the program)

   The operator can then abort the program or load an unlabeled disc on the proper channel and type in:

   :GO

   This returns to the *beginning* of the EXEC call.

4. If the label equals "*" and a subchannel is *not* given, DOS-M searches for an unlabeled disc, starting with the highest subchannel. DOS-M assigns the first unlabeled disc as the user disc, or if no unlabeled discs are found, it suspends the program and prints.

   DISC NOT ON SYS
   UD *nnnnn*
   xxxxx SUSP

   The operator can then abort the program or load an unlabeled disc and type in:

   :GO

   This returns to the beginning of the EXEC call.

   If the EXEC call specifies a subchannel with an incorrect system proprietary code (see Appendix J), DOS-M still makes the assignment but prints:

   TSB DISC *or* ??? DISC

If the EXEC call specifies a subchannel whose system generation code (see Section VII) does not match that of the system disc, DOS-M still makes the assignment, but prints:

DISC GEN CODE *nnnn* NOT SYS GEN CODE *nnn* ERR POS

The changes made by this EXEC call are only temporary, and will be reset at the end of each job.

If the specified subchannel is not active (physically present), DOS-M aborts the program and prints

UD *nnnnn* (*nnnnn* = address of EXEC call)

SECTION IV
PROGRAMMING

Section IV describes the operating procedures and formatting conventions of the five user programming aids of DOS-M:

- ALGOL Compiler
- FORTRAN Compiler
- Assembler
- Relocating Loader
- Relocatable Libraries

Using the EDIT directives, the operator creates and edits files of source programs written in FORTRAN, ALGOL, or Assembly Language. In load-and-go operations the DOS-M FORTRAN Compiler, ALGOL Compiler and DOS-M Assembler generate relocatable binary code onto temporary disc storage. The DOS-M Relocating Loader can then relocate and merge the code with referenced subroutines of the Relocatable Library. Once loaded, a program is executed by the PROG or RUN directive.

LOAD-AND-GO FACILITY

The Moving Head Disc Operating System provides the facility for "load-and-go" which is defined as compilation or assembly, loading, and execution of a user program without using intervening object paper tapes. To accomplish this, the compiler or assembler generates relocatable object code from source statements and stores it on the disc in the job binary area of the WORK tracks. Then separate directives initiate loading (PROG, LOADR) and execution (RUN, *program*).

DOS-M stores the object code of several programs and associated subroutines on the disc. The Relocating Loader locates them on the disc, and relocates them into executable absolute program units.

DOS-M FORTRAN COMPILER

The DOS-M FORTRAN Compiler, a segmented program, operates under control of the DOS-M Supervisor. The compiler consists of a main program (FTN) and four overlay segments (FTNØ1, FTNØ2, FTNØ3, FTNØ4). It resides on the disc and is read into core only when needed.

DOS-M FORTRAN, a problem-oriented programming language, is very similar to regular HP FORTRAN. Source programs, accepted from either an input device or a user file, are translated into relocatable object programs, punched on paper tape, and optionally, stored in the job binary area of the disc. The object program can be loaded using the DOS-M Relocating Loader and executed using the RUN or PROG directive.

Compiler Operation

The DOS-M FORTRAN compiler is started by a PROG directive. Before entering the PROG directive, place the source program in the input device, or, if input is from a source file, specify the file with a JFILE directive.

PROG, FTN

:PROG,FTN [,p_1,p_2,p_3,p_4,99]

Where

p_1 = logical unit of input device (standard is 5; set to 2 for source file input).
p_2 = logical unit of list device (standard is 6).
p_3 = logical unit of punch device (standard is 4).
p_4 = lines/page on listing (standard is 56).
99 = the job binary parameter. If present, the object program is stored in the job binary area for later loading. Any requested punch output still occurs. (The 99 may occur anywhere in the parameter list, but terminates the list.)

p_1 through p_4 are optional. If not present, the standard operation is assumed. If 99 is not present, then binary is not placed in the job binary area.

MESSAGES TO OPERATOR DURING COMPILATION

This message is printed on the operator console when an end-of-tape occurs on device #*n*:

I/O ERR ET EQT #*n*

EQT #*n* is unavailable until the operator declares it up:

:UP,*n*
:GO

Compilation continues after the GO. More than one source tape can be compiled into one program by loading the next tape before giving the GO.

At the end of compilation, the following message is printed.

```
$END, FTN
```

If the job binary area (where binary code is stored because of a 99 parameter) overflows, the following message is printed, and compilation continues:

```
JBIN OVF
```

There is no further loading into the job binary area.

The compiler terminates if...

- No JFILE is declared, although logical unit 2 has been given for input. Error E-ØØ19 is printed on the list device. ($END,FTN is not printed.)
- There are not enough work tracks for the compiler. The following message is printed:

  ```
  #TRACKS UNAVAILABLE
  ```

- Colons occur in the first column of a source program entered through the batch device. (Blank cards in the source program are ignored.) The following message is printed:

  IE *nnnnn*

  where *nnnnn* is the memory location of the input request.

FORTRAN CONTROL STATEMENT

Besides the standard options described in the FORTRAN manual, two new compiler options, T and *n*, are available. A "T" lists the symbol table for each program in the compilation. If a "u" follows the address of a variable, that variable is undefined (the program does not assign a value to it). The A option includes this T option. If *n* appears, *n* is a decimal digit (1 through 9) which specifies an error routine. The user must supply an error routine, ERR*n*. If this option does not appear, the standard library error routine, ERRØ, is used. The error routine is called when an error occurs in ALOG, SQRT, .RTOR, SIN, COS, .TROI, EXP, .ITOI or TAN.

PROGRAM STATEMENT

The program statement includes an optional type parameter.

PROGRAM *name* [,*type*]

where *name* is the name of the program and its main entry point. When the program is executed using a RUN directive, this *name* is used. (It should not equal any file name.)

type is a decimal digit specifying the program type. Only types 3 (main), 5 (segment), and 6 or 7 (library) are significant in DOS-M. The type is set to 3 if not given.

Seven more parameters may be included but they are used only with the Real-Time Executive System. Programs can be compiled on DOS-M to be run under Real-Time. (Consult the *Real-Time Software* Manual.)

I/O LOGICAL UNIT NUMBERS

DOS-M FORTRAN function assignments for logical unit numbers are different from regular FORTRAN. (See Section V.)

When preparing input data for the batch device, the user never puts a colon (:) in column one of a record because the colon in first position signifies a directive. DOS-M aborts the job if a directive occurs during data input.

DATA STATEMENT

A new statement, the DATA statement, has been added to DOS-M FORTRAN. DATA sets initial values for variables and array elements. The format of the DATA statement is:

$$\text{DATA } k_1/d_1/,k_2/,\ldots,k_n/d_n/$$

where k is a list of variables and array elements separated by commas,

d is a list of constants or signed constants, separated by commas and optionally preceded by j* (j is an integer constant).

The elements of d_i are serially assigned to the elements of k_i. The form j* means that the constant is assigned j times. The k_i and d_i must correspond one-to-one.

Elements of k_i may not be from COMMON.

Arrays must be defined (i.e., DIMENSION) before the DATA statements in which they appear. DATA statements may occur anywhere in a program following the specification statements.

Example,

```
DIMENSION A(3), I(2)
DATA A(1),A(2),A(3)/1.Ø,2.Ø,3.Ø/I(1),I(2)/2*1/
```

EXTERNAL STATEMENT

With the new statement, EXTERNAL, subroutines and functions can be passed as parameters in a subroutine or function call. For example, the routine XYZ can be passed to a subroutine if XYZ is previously declared EXTERNAL. Each program may declare up to five EXTERNAL routines.

The format of the EXTERNAL statement is

EXTERNAL $v_1, v_2, \ldots, v_5$

Where v_1 is the entry point of a function, subroutine or library program.

EXAMPLE

```
FUNCTION RMX(X,Y,A,B)
RMX=(A)*Y(B)
END
EXTERNAL XYZ,FL1
Z=Q-RMX(XYZ,FL1,3.56,4.75)
```

ERROR E-ØØ18 means too many EXTERNALS.

Note: If a library routine, such as SIN, is used as an EXTERNAL, the compiler changes the first letter of the entry point to "%". Special versions of the library routines exist with the first character changed to "%".

PAUSE & STOP

PAUSE causes the following message to be printed.

PAUSE *xxxx*

Where *xxxx* is an octal number.

To restart the program, the operator uses a GO directive.

STOP causes the program to terminate after the following message.

STOP *program name xxxx*

Where *xxxx* is an octal number.

OVERLAY SEGMENTS

Segmented user programs may be written in FORTRAN, but certain conventions are required. A segment must be defined as type 5 in the PROGRAM statement. The segment must be initiated using the PROGRAM SEGMENT LOAD EXEC call from main or segment. A dummy call to main must appear in each segment. In this way, the proper linkage is established between the main and its segments.

Chaining of segments is unidirectional. Once a segment is loaded, execution transfers to it. The segment, in turn, may call another segment using an EXEC call, but a segment written in FORTRAN cannot return to the main program. All communication between the main program and segments must be through COMMON. Segments must not contain DATA Statements.

ERRØ LIBRARY ROUTINE

ERRØ, the error print routine referred to under the FORTRAN control statement prints the following message whenever an error occurs in a library routine:

nn xx

Where *nn* is the routine identifier, and
xx is the error type.

The compiler generates calls to ERRØ automatically. If the FORTRAN control statement includes an *n* option, the call will be to ERR*n*, a routine which the user must supply.

Check the *FORTRAN* manual (Chapter 9.9) for the meaning of error codes.

REFERENCE ON FORTRAN

For a complete description of the FORTRAN language, read the *FORTRAN* programmer's reference manual (02116-9015). Sections 9.5,9.6, and 9.8 are not pertinent to DOS-M FORTRAN.

RTE/DOS ALGOL COMPILER

The RTE/DOS ALGOL Compiler consists of a main program and a data segment. It requires a 16K memory computer and can operate under the control of DOS-M, DOS, or the RTE System. The compiler resides on the disc and is read into core when called for in a :PROG directive. RTE/DOS ALGOL is very similar to the HP ALGOL language described in manual HP 02116-9072. The HP ALGOL compiler implements a language much like ALGOL 60, but it is non-recursive and has I/O capabilities.

Source programs written in DOS-M ALGOL are accepted either from an input device or from a user file and are translated by the ALGOL Compiler into relocatable object programs, punched on paper tape, and optionally, stored in the job binary area of the disc. The object program can be loaded using the DOS-M RELOCATING LOADER and executed using the RUN or PROG directive.

Compiler Operation

The ALGOL Compiler is started by a PROG directive. Before entering the PROG directive place the source program in the input device, or, if the input is from a source file, specify the file with a JFILE directive. The PROG directive for the ALGOL Compiler should take the following form:

PROG,ALGOL

:PROG, ALGOL, p_1,p_2,p_3,p_4,p_5

Where

p_1 = Input unit (=5 if not specified). Input unit = 2 means source input from disc. The source file has to be specified prior to this statement (by "JFILE" control statement).

p_2 = List unit (=6 if not specified).

p_3 = Punch unit (=4 if not specified).

p_4 = Number of lines on a page (=56 if not specified).

p_5 = Load-and-go parameter. To specify load-and-go, set p_5=99. The value of 99 is reserved for the load-and-go parameter. Its appearance in any position (p_1 through p_5) will be interpreted as p_5=99, and it also signals the end of the parameter list.

MESSAGES TO OPERATOR DURING COMPILATION

When the end of a source tape is encountered, the following will be outputed on the system teleprinter:

I/O ERR ET EQT #*n*

The compiler will wait until the following messages are entered on the system teleprinter:

:UP,*n*
:GO

At the end of the compilation, the following message is output to the system teleprinter:

```
$END, ALGOL
```

If the job binary area (where binary code is stored because of a "99" parameter in the PROG directive) overflows, the following message is output by the system teleprinter and compilation continues:

```
JBIN OVF
```

The compilation will be completed, but there will be no further loading of binary code into the job binary area.

The compiler terminates if...

- No JFILE is declared, although logical unit 2 had been specified as p_1 of the PROG directive. The following message is output:

  ```
  NO SOURCE
  ```

- The first statement of the source file specified by the PROG directive p_1 parameter does not begin with the word HPAL. Or the control statement contains an error. The following message is output:

  ```
  HPAL??
  ```

- A colon occurs in the first position of a source statement line. The following message is output:

  IE *nnnnn*

  where *nnnnn* is the memory location of the input request.

ALGOL Control Statement

The word HPAL is mandatory. Any combination of the following symbols may appear next, separated by commas:

L: produce source program listing
A: produce object code listing
B: produce object tape
P: a procedure only is to be compiled

If no symbols are specified, the program will run but will not produce any output other than diagnostic messages and job binary (if requested). A program name in quotes (the NAM-record name which must be a legitimate identifier without blanks) must follow the symbols. (It should not equal any file name.) (It should not equal any file name.)

Sense switch control is not used with DOS-M. Two parameters may be specified following the NAM-record name.

> p_1 is a decimal digit between Ø and 9 specifying the name of the error routine to be called if an error occurs in ALOG, SQRT, .RTOR, SIN, COS, .RTOI, EXP, .ITOI, TAN. The name of the error routine is ERR*n*, where $n = p_1$ or $n = Ø$ if p_1 is not specified. ERRØ is supplied in the Relocatable Library, all other error routines must be supplied by the user.
>
> p_2 is a decimal digit specifying the type of the program: 3 for a main program, 5 for a segment, and 6 or 7 for a utility subroutine or procedure. If p_2 is not specified, the type is set to 3 for main programs and to 7 for procedures (P option in the control statement).

EXAMPLE

```
HPAL,L,B,"TEST",1,3
```

ALGOL Segmentation

ALGOL programs can be segmented if certain conventions are followed. A segment must be defined as type 5 in the HPAL statement. The segment must be initiated by using the PROGRAM SEGMENT LOAD EXEC call from the main or another segment.

In order to establish the proper linkage between a main program and its segments, each segment must declare the main a code procedure. For example, if MAIN is the main program, the following must be declared in each segment.

```
PROCEDURE MAIN; CODE:
```

Chaining of segments is undirectional. Once a segment is loaded, execution transfers to it. The segment, in turn, may call another segment using an EXEC call, but a segment written in ALGOL cannot return to the main program.

ALGOL I/O

The HP ALGOL I/O statements should specify the proper logical unit numbers for the DOS-M configuration. (See Section V.)

ALGOL Error Messages

See the manual HP ALGOL (HP 02116-9072) for the meanings of HP ALGOL compilation time and run time error messages.

DOS-M ASSEMBLER

The DOS-M Assembler, a segmented program that executes in the user program area of core, operates under control of DOS-M. The Assembler consists of a main program (ASMB) and six segments (ASMBD, ASMB1, ASMB2, ASMB3, ASMB4, ASMB5), and resides on the disc.

DOS-M Assembly Language, a machine-oriented programming language, is very similar to the HP Extended Assembly Language. Source programs, accepted from either an input device or a user source file on the disc, are translated into absolute or relocatable object programs; absolute code is punched in binary records, suitable for execution only outside of DOS-M. ASMB can store relocatable code in the load-and-go area of the disc for on-line execution, as well as punch it on paper tape. The DOS-M Relocating Loader accepts assembly language relocatable object programs from paper tape, the load-and-go area, and user files.

A source program passes through the input device only once, unless there is insufficient disc storage space. In the latter case, two passes are required.

Assembler Operation

The DOS-M Assembler is started by a PROG directive. However, before entering the PROG directive, the operator must place the source program in the input device. If the source program is on the disc, the operator must first specify the file with a JFILE directive, and set parameter p_1 = logical unit 2 in the PROG directive.

PROG,ASMB

:PROG,ASMB,p_1,p_2,p_3,p_4,99

Where

p_1 = logical unit of input device (5 is standard; 2 is used for source file input indicated by a JFILE directive)

p_2 = logical unit of list device (6 is standard)

p_3 = logical unit of punch device (4 is standard)

p_4 = lines/page on listing (56 is standard)

99 = job binary parameter. If present, the object program is stored in the job binary area for later loading. Any requested punching still occurs. The 99, which may follow any parameter in the list, terminates the list.

MESSAGES DURING ASSEMBLY

The messages described in this section are printed at the teleprinter console or in the program listing.

When an end-of-tape occurs on device #*n*, this message appears on the system teleprinter:

I/O ERR ET EQT #*n*

EQT #*n* is unavailable until the operator declares it up and restarts the assembler by means of a GO directive:

:UP,*n*
:GO

Thus, more than one source tape can be assembled into one program. The next tape is loaded each time the input device goes down. The program should be placed in the input device before entering the GO.

The following message on the system teleprinter signifies the end of assembly:

```
$END ASMB
```

If another pass of the source program is required, the message is printed on the system teleprinter at the end of pass one.

```
$END ASMB PASS
```

The operator must replace the program in the input device and type:

```
:GO
```

If an error is found in the Assembler control statement, the following message is printed on the system teleprinter:

```
$END ASMB CS
```

The current assembly stops.

If an end-of-file condition on source input occurs before an END statement is found, the teleprinter signals:

```
$END ASMB XEND
```

The current assembly stops.

If source input for logical unit 2 (disc) is requested, but no file has been declared (see JFILE, Section II), the system teleprinter signals:

```
$END ASMB NPRG
```

If the job binary area, where binary code is stored by a 99 parameter, overflows, assembly continues but the following message is printed on the system teleprinter:

```
JBIN OVF
```

However, no binary code is stored in the job binary area.

The next message is associated with each error diagnostic printed in the program listing during pass 1.

nnn

nnn is the "tape" number on which the error (reported on the next line of the listing) occurred. A program may consist of more than one tape. The tape counter starts with one and increments by one whenever an end-of-tape condition occurs (paper tape) or a blank card is encountered. When the counter increments, the numbering of source statements starts over at one.

Each error diagnostic printed in the program listing during pass 2 of the assembly is associated with a different message:

PG *ppp*

ppp is the page number (in the listing) of the *previous* error diagnostic. PG ØØØ is associated with the first error found in the program.

These messages (#*nnn* and PG *ppp*) occur on a separate line, just above each error diagnostic in the listing.

DOS-M Assembly Language

The DOS-M Assembly Language is equivalent to extended assembly language, as defined in the *ASSEMBLER* programmer's reference manual (02116-9014). A few language changes are required to run under DOS-M; programs must request certain functions, such as I/O, from the executive. These requests are made using the EXEC calls described in Section III.

ASSEMBLER CONTROL STATEMENT

The control statement has the same form as that of regular assembly language; and although only relocatable code can be run under DOS-M, the DOS-M Assembler is able to assemble absolute code if it is specified. Absolute code is never

stored in the job binary area. To get absolute code, the control statement must include an "A". The "R", however, is not required for relocatable code. An "X" causes the assembler to generate non-extended arithmetic unit code.

Examples

ASMB,L,B	List and Punch Relocatable Binary
ASMB,R,L,B,X	List and Punch Relocatable, non-EAU Binary.
ASMB,T,L	List and Print Symbol Table.
ASMB,A,B,L	List and Punch Absolute Binary.

ORB STATEMENT

DOS-M Assembly Language does not contain the ORB statement, since information cannot be loaded into the protected base page area by user programs. However, programs can read information from base page using absolute address operands up to 1777_8.

INPUT/OUTPUT

DOS-M has different function assignments for the logical unit numbers. (See Section V.)

When preparing input for the batch device, the programmer must remember to never put a colon (:) in column one of a source statement. DOS-M aborts the current program if a directive (signified by : in column one) occurs during data input.

If present, the memory protect option protects the resident supervisor from alteration and interrupts the execution of a user program under these conditions:

- Any operation that would modify the protected area or jump into it.
- Any I/O instruction, except those referencing the switch register or overflow.
- Any halt instruction.

Memory protect gives control to DOS-M when an interrupt occurs, and DOS-M checks whether it was an EXEC call. If not, the user program is aborted.

NAM STATEMENT

The NAM psuedo-instruction allows up to eight optional parameters. (The last seven parameters are used only by programs to be executed under the Real-Time Executive System.) Only the first parameter is significant in DOS-M. If the first parameter equals 3, the program is a main program; if 5, a program segment; if 6, a library routine; if 7, a subroutine. If the parameter equals another number, the assembler and DSGEN will accept it, but the Relocating Loader will not. (See Section VI for DSGEN program type codes.)

NAM *name* [,*type*]

where *name* is the name of the program (it should not equal any file name), and

type is the type code.

In addition to the *name* defined by NAM, each program has one or more *entry points* defined by an ENT statement with the exception of the main program. The transfer address on the END statement is sufficient for the main program (type 3). *Name* is used in programmer-to-DOS-M communication, while the *entry points* are program-to-program communication.

Segmented Programs

User programs may be structured into a main program and several segments, as shown in Figure 4-1. The main program begins at the start of the user program area. The area for the segments starts immediately following the last location of the main program. The segments reside on the disc, and are read into core by an EXEC call, when needed. Only one segment may be in core at a time. When a segment is read into core, it overlays the segment previously in core.

The main program must be type 3, and the segments must be type 5. When using DSGEN to configure the system or loading programs with LOADR, the main program must be entered prior to its segments. One external reference from each segment to the main routine is required for DSGEN to link the segments and main programs. Also, each segmented program should use unique external reference symbols. Otherwise, DSGEN or LOADR may link segments and main programs incorrectly.

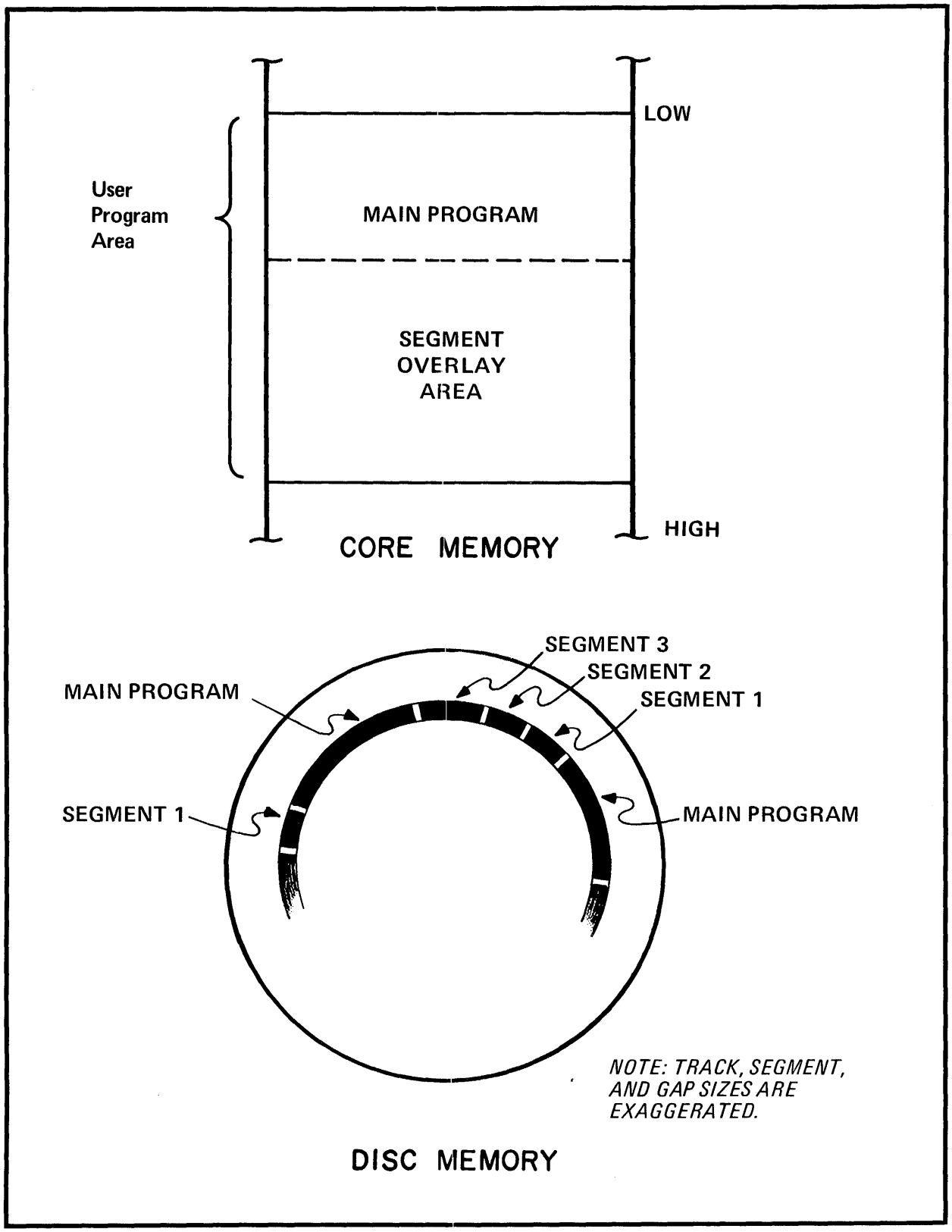


Figure 4-1. Segmented Programs

Figure 4-2 shows how an executing program may call in any of its segments from the disc using the PROGRAM SEGMENT LOAD EXEC request (1-2). DOS-M locates the segment on the disc (3-4), loads it into core (5) and begins executing it. The segment may call in another of the main program's segments using the same EXEC request (6).

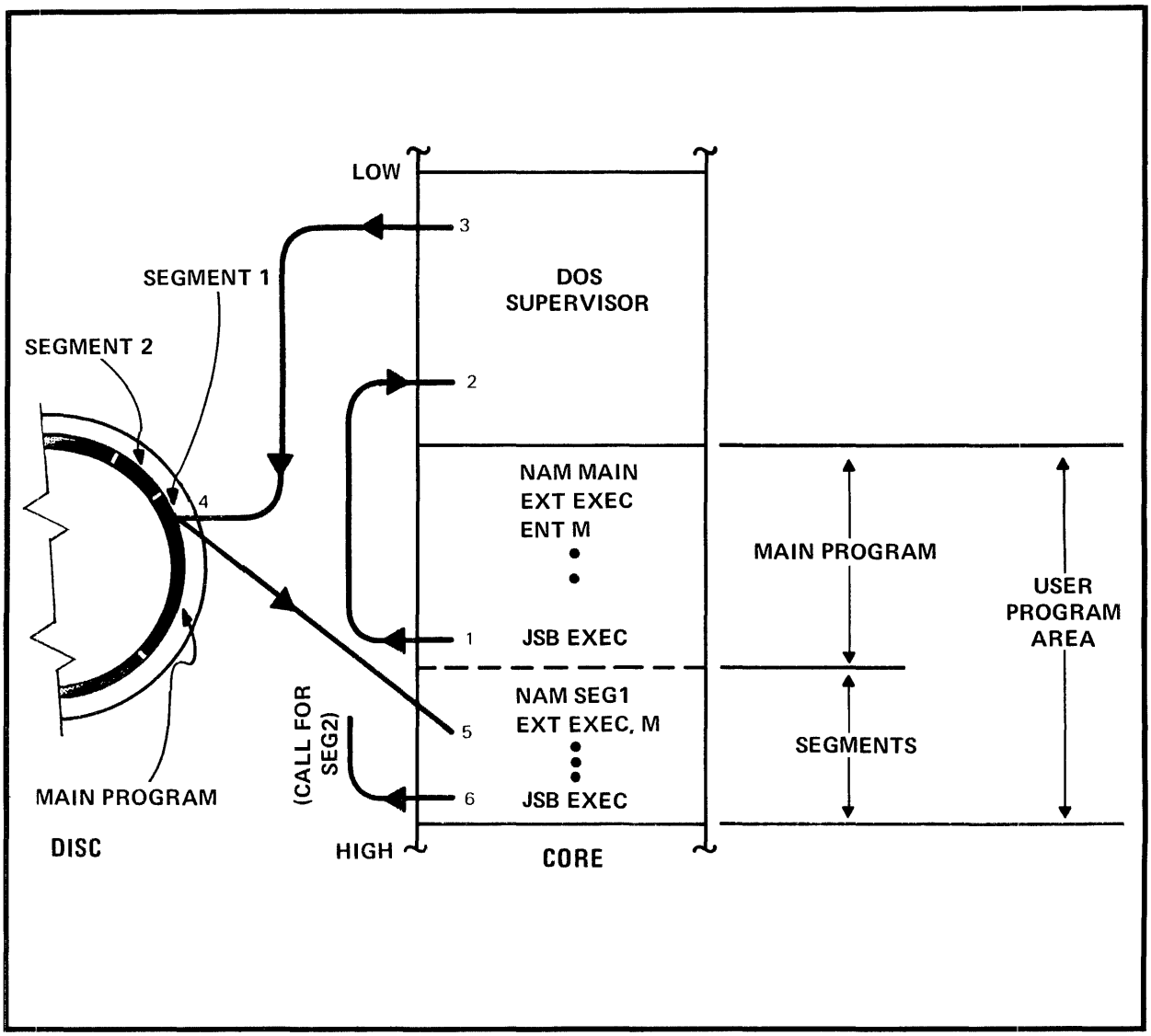


Figure 4-2. Main Calling Segment

Figure 4-3 shows how DOS-M processes the request from the segment (7) by locating the segment on the disc (8-9), loading it into core (10), and beginning execution of it.

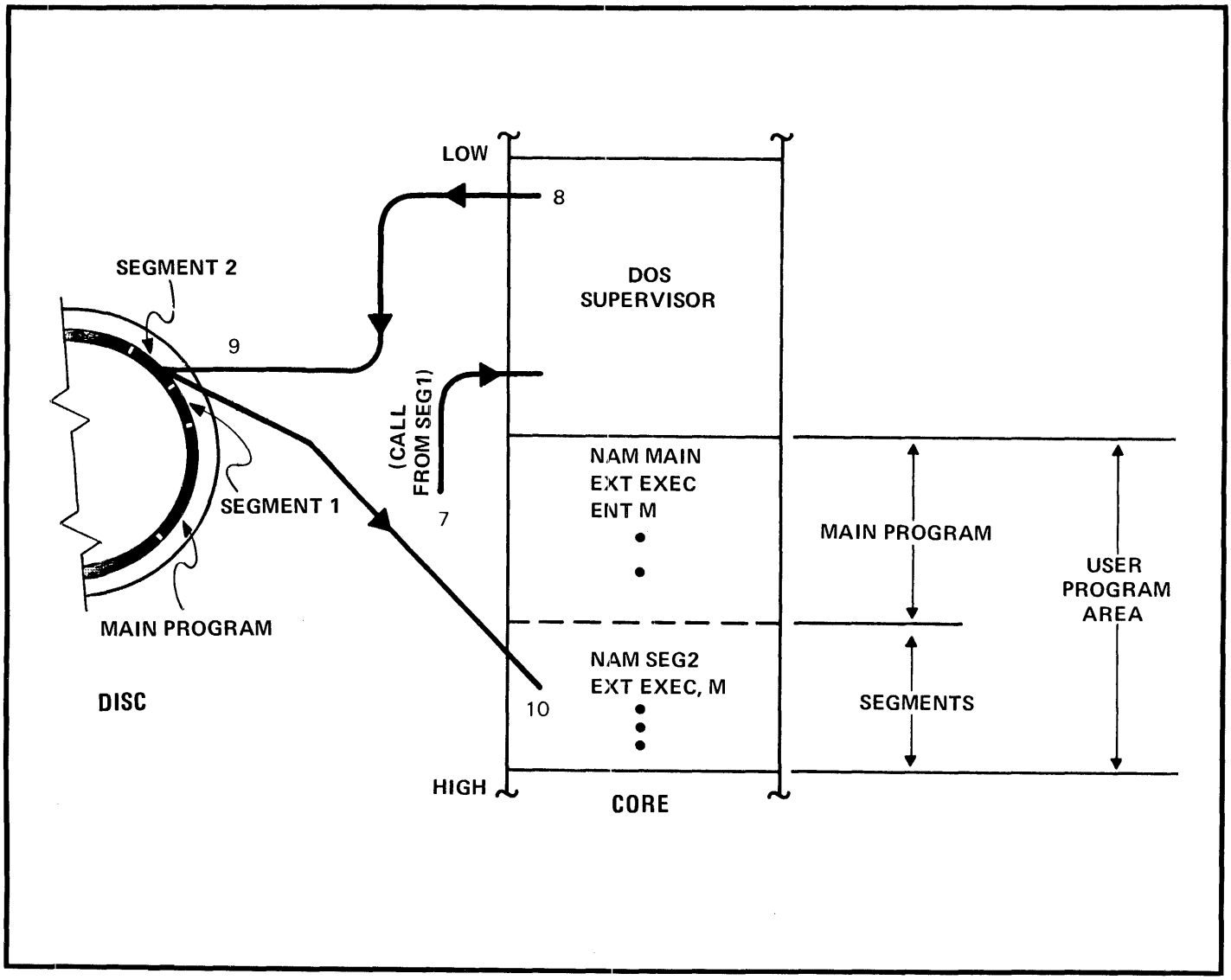


Figure 4-3. Segment Calling Segment

When a main program and segment are currently residing in core, they operate as one single program. Jumps from a segment to a main program (or vice versa) can be programmed by declaring an external symbol and referencing it via a JMP instruction. (See Figure 4-4.) A matching entry symbol must be defined as the destination in the other program. DSGEN associates

the main programs and segments, replacing the symbolic linkage with actual absolute addresses (i.e., a jump into a segment is executed as a jump to a specific address). The programmer should be sure that the correct segment is in core before any JMP instructions are executed.

Reference on Assembly Language

Consult the *ASSEMBLER* programmer's reference manual (02116-9014) for a full description of assembly language. Sections 5.5 and 5.6 of that text do not apply to DOS-M.

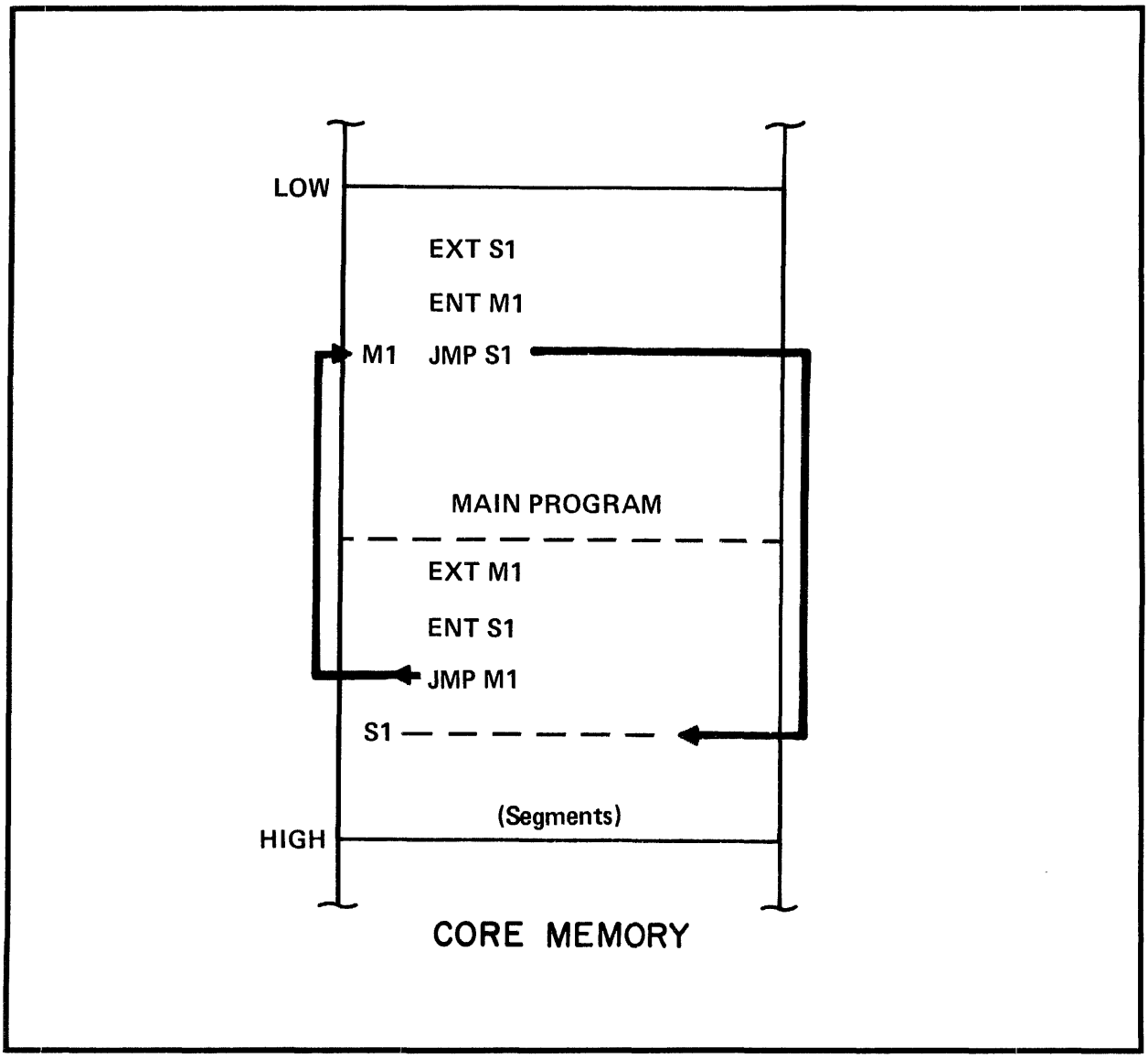


Figure 4-4. Main-to-Segment Jumps

DOS-M RELOCATING LOADER

The DOS-M Relocating Loader accepts relocatable object programs which have been translated by the DOS-M Assembler, RTE/DOS ALGOL Compiler or DOS-M FORTRAN Compiler. It generates an executable core image of each such program on the disc. The relocatable programs may enter the loader as

- Job binary area programs translated during the current job,
- User files,
- Punched tapes, or
- Subroutines from the disc-resident Relocatable Library.

Each main program is relocated to the start of the user area and linked to its external references, such as library routines. Segments will overlay the area following the main program and its subroutines. Programs may run under control of the DEBUG library routine. The main program, plus its subroutines and its longest segment, can be as large as the user area. With a RUN or PROG directive, the program is called by name from the disc and executed, or the program is stored as a permanent user file to be run during a later job. The loader may be executed only once during each job, so all load-and-go assemblies or compilations must be done prior to calling the loader.

Starting the Loader

The DOS-M Relocating Loader is initiated by a PROG directive from the batch or keyboard device.

PROG,LOADR

Format

:PROG,LOADR[,P_1,P_2,P_3,P_4,P_5]

P_1 determines the relocatable object program input combination:

- P_1 = Ø for loading from jbin and relocatable library.
- = 2 for loading from jbin, user files, and relocatable library.
- = *n* for loading from jbin, user files, relocatable library and paper tape (logical unit *n*).

P_2 = list device logical unit.

P_3 = Ø for no DEBUG, ≠ Ø for DEBUG.

P_4 = Ø for list of program load map, ≠ Ø for none.

P_5 = Ø for list of entry point addresses, ≠ Ø for none.

If values $P_1,...P_5$ are not set, P_1=Ø, P_2=6, P_3=P_4=P_5=Ø.

Comments

Selecting the DEBUG option causes DEBUG to be appended to each main program and segment. The loader sets the primary entry point of each to DEBUG, rather than the user routine. When the program is run, DEBUG takes control of the program's execution and seeks instructions from the keyboard.

RELOCATABLE FILES

A list of relocatable file names follows the PROG directive (unless P_1 equals Ø). In batch mode, the list starts on the next record and stops at "/E". In keyboard mode, the loader prints

ENTER FILE NAME(S) OR /E

then waits for input. After each list of files is entered, the message repeats until a /E is entered. In batch mode the list of files follows the PROG directive on the batch input device.

file-name 1, file-name 2,..,/E

The file list is a series of records containing file names separated by commas, ending with a "/E." All programs in each file are loaded unless a particular subset of the file is specified:

file-name (prog 1, prog 2...)

Only the programs specified within the parenthesis are loaded from the *file-name*. The file list is simply a "/E" if no files are to be loaded. (The search for these files is made only on the current user disc; the loader is unaffected by :SS.)

Operating the Loader

SCANNING THE PROGRAMS

The loader scans the relocatable binary programs and maintains two tables--one of program names, and another of entry points and externals. Since mains are matched with segments during the scan, each main program must occur before the associated segments. Programs from tape are stored on the work tracks as they are read in.

If the job binary area contains any programs, it is scanned first. User files given in the file list (if any) are scanned for entries and externals.

If paper tape input is requested, the following messages are printed,

```
LOAD TAPE
LOADR SUSP
@
```

The loader suspends. The operator places a tape in the input device and types

```
:GO
```

When an end-of-tape condition occurs, three messages are printed on the system teleprinter:

```
I/O ERR ET EQT# nn
LOAD TAPE
LOADR SUSP
@
```

The operator places the next tape in the input device, enters :UP,*n*, and :GO to read the next tape. Enter :GO,1 to indicate that all tapes have been read in.

Matching Entries with Externals

After matching all possible entry points and external references in the user programs, the loader scans the Relocatable Library (disc-resident) looking for entry points to match the undefined external references. If undefined external references still exist,

```
UNDEFINED EXTS
```

is printed and the external references are listed, one per line.

To load additional programs from paper tape, the operator types:

:GO,Ø[,*n*]

where *n* is the logical unit number of the input device, if different from P_1 of the PROG,LOADR directive.

To continue without fulfilling external references, the operator types:

:GO,1

To specify a file name from the keyboard, the following directive is typed:

:GO,2

RELOCATION

The main and segment names become user file names once the programs are loaded. To ensure unique file names, the loader compares all program and segment names against the names of previous system and user files (current user disc only). If duplicate names occur, an error message is printed and loading stops.

The loader converts each main program into an absolute core image, stores it on the disc, places the name in the user directory where it remains during the current job, and lists the program address map and entry points, if requested. After each main program, any associated segments are loaded in the same way. When the loader is completely finished, the following message is printed:

LOADR COMPLETED

During the current job, the absolute core images appear in the user file area (see LIST directive, Section II) and can be executed by name (see RUN and PROG directives). At the end of the job, however, they disappear from the file area, unless they are made permanent files by means of the STORE directive.

If no programs are entered, the loader prints the following messages and terminates:

```
NO PROGRAMS LOADED.
LOADR COMPLETED
```

DEBUG Library Subroutine

RTE/DOS DEBUG, a subroutine of the Relocatable Library, allows programmers to check for logical errors during execution. If the P_3 parameter of the PROG, LOADR directive equals 1, the loader combines DEBUG with the user program being loaded. The primary entry point (the location where execution begins) is set to DEBUG. Therefore, when the program is executed with a RUN directive, DEBUG takes control and prints the message:

```
BEGIN 'DEBUG' OPERATION
```

The programmer now enters any legal debug operation. DEBUG ignores illegal requests and prints a message:

```
ENTRY ERROR
```

DEBUG OPERATIONS

B,*A*	Instruction breakpoint at address *A*. (NOTE: if *A* = JSB EXEC, a memory protect violation occurs.)
D,A,N_1[,N_2]	ASCII dump of core address N_1 or from N_1 to N_2.
D,B,N_1[,N_2]	Binary dump of core address N_1 or from N_1 to N_2.
M,*A*	Sets absolute base of relocatable program unit.
R,*A*	Execute user program starting at *A*. Execute starting at next location in user program (used after a breakpoint or to initiate the program at the transfer point in the user program).
S,A_1,D_1	Set D_1 in location A_1.
S,A_1,D_1,D_n	Set D_1 to D_n in successive memory locations beginning at location A_1.
W,A,D_1	Set A-Register to D_1.
W,B,D_2	Set B-Register to D_2.
W,E,D_3	Set E-Register (Ø=off, non-zero=on).
W,O,D_4	Set Overflow (Ø=off, non-zero=on).
X,*A*	Clear breakpoint at address *A*.
A	Abort Debug operation.

Loader Example

In the following example, DOS-M is in keyboard mode.

```
:PROG,LOADR,5,6,Ø,Ø,Ø        Paper tape input is specified.
ENTER FILE NAME(S)OR/E       No files are specified.
/E
LOAD TAPE                    Place paper tape in input device.
LOADR SUSP
@:GO                         Return to loader.
I/O ERR ET EQT # Ø3          End of tape.
LOAD TAPE                    Put in next tape.
LOADR SUSP
@:UP,3                       Declare input device ready.
@:GO
I/O ERR ET EQT # Ø3
LOAD TAPE
LOADR SUSP
@:UP,3
@:GO
I/O ERR ET EQT # Ø3
LOAD TAPE
LOADR SUSP
@:UP,3
@:GO                         Repeat tape loading process 4 times.
I/O ERR ET EQT # Ø3
LOAD TAPE
LOADR SUSP
@:UP,3
@:GO
I/O ERR ET EQT # Ø3
LOAD TAPE
LOADR SUSP
@:UP,3
@:GO,1                       No more paper tapes.
```

RELOCATING LOADER

NAME/ENTRY	ADDR	
QA1	12ØØØ	Main program, starting address.
*QA1	12Ø76	Main program, entry point.
QA1A	122ØØ	
*QA1A	122Ø1	
QA1B	12262	
*QA1B	12263	
QA1C	12336	
*QA1C	12337	
QA1D	12364	
*QA1D	12365	
FRMTR	12431	
*.DIO.	14612	
*.BIO.	14665	
*.IOI.	145Ø7	
*.IOR.	14462	Subroutine starting addresses and entry
*.IAR.	14546	points. Asterisk signifies entry point.
*.RAR.	14522	
*.DTA.	1471Ø	
.ENTR	15162	
*.ENTR	15162	
.FLUN	1523Ø	
*.FLUN	1523Ø	
.PACK	15243	
*.PACK	15243	
FLOAT	1535Ø	
*FLOAT	1535Ø	
IFIX	15355	
*IFIX	15355	
LOADR COMPLETE		End of Loading.

Loader Error Messages

During its operation the loader may print one of the following error messages on the keyboard:

Message	Error Messages
LØ1	Checksum error on tape
LØ2	Illegal record
LØ3	Memory overflow
LØ4	Base page overflow
LØ5	Symbol table overflow
LØ6	Duplicate main or segment name (may be caused by attempting to run the loader twice in one job)
LØ7	Duplicate entry point
LØ8	No main or segment transfer address
LØ9	Record out of sequence
L1Ø	Insufficient directory or work area space
L11	Program name table overflow
L12	User file specified cannot be found
L13	Program name duplication
L14	Non-zero base page length
L15	Segment occurred before main
L16	Program overlay (illegal ORG)

The loader aborts (programmer must start over) on each of these conditions, and prints a message.

```
LOADR TERMINATED
```

THE RELOCATABLE LIBRARIES

There are two libraries, or collections of relocatable subroutines that can be used by DOS-M: the RTE/DOS Relocatable Library (EAU or Non-EAU versions) and the RTE/DOS FORTRAN IV Library. These libraries contain mathematical routines such as SIN and COS, and utility routines such as BINRY, etc. A program signifies its need for a subroutine by means of an "external reference." External references are generated by EXT statements in assembly language, by CALL statements and the compiler in FORTRAN, and by CODE procedures and the compiler in ALGOL.

When the system is generated, several combinations of libraries are possible. Every system must contain an RTE/DOS Relocatable Library: either an EAU version or a non-EAU version, depending on the computer hardware. This library does not contain a formatter, but the FORTRAN IV Library contains a formatter that handles extended precision numbers. If extended precision arithmetic is not needed, a separate RTE/DOS Basic FORTRAN Formatter is available to take the place of the FORTRAN IV Library.

All of these libraries and the subroutines they contain are documented in the *Relocatable Subroutines* manual (02116-9032).

SECTION V
INPUT/OUTPUT

In the Moving-Head Disc Operating System, centralized control and logical referencing of I/O operations effect simple, device-independent programming. Each I/O device is interfaced to the computer through one or more I/O channels which are linked by hardware to corresponding core locations for interrupt processing. By means of several user-defined I/O tables, multiple-device drivers, and program EXEC calls, DOS-M relieves the programmer of most I/O problems.

SOFTWARE I/O STRUCTURE

An Equipment Table records each device's I/O channels, driver entry points, DMA requirements, and location on disc if disc-resident. A Device Reference Table (logical unit table) assigns an equipment table number to each of its entries, thus allowing the programmer to reference changeable logical units instead of fixed physical units.

An Interrupt Table relates each channel to an entry in the Equipment Table.

A driver is responsible for initiating and continuing operations on all devices of an equivalent type.

The programmer requests I/O by means of an EXEC call in which he specifies only the logical unit, control information, buffer location, buffer length, and type of operation.

The Equipment Table

The Equipment Table (EQT) has an entry for each device recognized by DOS-M (these entries are established by the user when DOS-M is generated). The EQT entries reside in the permanent core-resident part of the system and have this format:

WORD	CONTENTS																
1	Driver "Initiation" Section Address																
2	Driver "Continuation" Section Address																
3	D	R						Unit #					Channel #				
4	Av		Equipment Type Code						Status								
5	(saved for system use)																
6	(saved for system use)																
7	Request Return Address																
8	Request Code																
9	Current I/O Request Control Word																
10	Request Buffer Address																
11	Request Buffer Length																
12	Temporary or Disc Track #																
13	Temporary or Starting Sector #																
14	Temporary Storage for Driver																
15	Upper Memory Address of Main Driver Area																Ø's if core-resident
16	Upper Memory Address of Driver Linkage Area																
17	Starting Track #								Starting Sector #								
BITS	15	14	13	12	11	1Ø	9	8	7	6	5	4	3	2	1	Ø	

D = 1 if DMA channel required.

R = 1 if driver type is core-resident.

Unit # May be used for sub-channel addressing.

Channel # I/O select code for device (lower number if multiboard interface.)

Av = Ø - Unit not busy and available
1 - Unit disabled (down)
2 - Unit busy
3 - Unit waiting for an available DMA channel

Status - Actual or simulated unit status at end of operation.

Equipment Type Code - Identifies type of device and associated software driver. Assigned equipment type codes in octal are:

ØØ-Ø7	Paper Tape Devices
ØØ	Teleprinter
Ø1	Punched Tape Reader
Ø2	High Speed Punch
Ø5	Teletype (System)
1Ø-17	Unit Record Devices
1Ø	Reserved for Plotter
12	Line Printer
15	Mark Sense Card Reader
2Ø-37	Magnetic Tape/Mass Storage and other devices capable of both input and output
22	3030 Magnetic Tape
31	Moving-Head Disc

For equipment type codes Ø1 through 17, odd numbers indicate input devices and even numbers indicate output devices (except Ø5, which is both input and output).

When DOS-M initiates or continues an I/O operation, it places the address of the EQT entry for the device into the base page communication area (see Appendix A) before calling the driver routine.

Logical Unit Numbers

Logical unit numbers from 1_{10} to 63_{10} provide logical addressing of the physical devices defined in the EQT. These numbers are maintained in the Device Reference Table (DRT or logical unit table), which is created by the System Generator (DSGEN) and can be modified by the LU directive.

Each one-word entry in the DRT contains the EQT entry number of the device assigned to the logical unit. DOS-M has the following function assignments for logical unit numbers.

	Logical Unit Number	Function
Restored after each :JOB.	1	System Teleprinter
	2	User Mass Storage
	3	System Mass Storage
	4	Standard Punch Device
	5	Standard Input Device
	6	Standard List Device
	7	Can be assigned to any device by user
	8	
	9	
	10	
	:	
	63_{10}	

The user determines the number of logical units when the system is generated. At the beginning of each JOB, logical units 1 through 9 are restored to the values set by DSGEN (System Generator), whereas 10 through 63 are restored only on a start-up from the disc.

Executing programs use logical unit numbers to specify the type of device for I/O transfers. In an I/O EXEC call, the program simply specifies a logical unit number and does not need to know which actual device or which I/O channel handles the transfer.

The Interrupt Table

The interrupt table contains an entry, established at system generation time, for each I/O channel in the computer which can cause an interrupt. The entry contains the address of the EQT entry for the device on the channel.

The interrupt locations in core contain a jump subroutine to $CIC which is the central interrupt control routine which examines the interrupt table to decide what action to take. On a power failure interrupt, DOS-M halts. However, the user can write his own routine to handle power failure interrupts.

Input/Output Drivers

The I/O driver routines, either core-or disc-resident, handle the actual transfer of information between the computer and external devices. When a transfer is initiated, DOS-M places the EQT entry addresses into the base page communication area and jumps to the driver entry point. The driver configures itself for the particular channel (in this way the same driver can handle several devices of the same type on many channels), initiates the transfer and returns to DOS-M. When an interrupt occurs on the channel, indicating continuation or completion of the transfer, DOS-M again transfers control to the driver. DOS-M contains only two drivers: the Moving-Head Disc Driver (DVR31) and the System Teleprinter Driver (DVRØ5). However, these drivers of the Disc Operating System (DOS, fixed-head disc/drum) are fully compatible with DOS-M:

DVRØØ - Teleprinter
DVRØ1 - Photo-reader
DVRØ2 - High speed punch
DVR1Ø - Plotter
DVR12 - Line Printer
DVR15 - Mark Sense Card Reader
DVR22 - 3030 Magnetic Tape

The driver name consists of the letters "DVR" added to the equipment type code. In addition, the programmer can write drivers for special devices, following the guidelines in this section. The driver is only responsible for updating the status field in the EQT entry; DOS-M handles the availability field.

System I/O

DOS-M itself initiates many I/O transfers. It reads in directives from the batch or keyboard device and transfers modules in from the disc. These functions are accomplished by $SYIO, a routine within the DOS-M Supervisor, which calls the appropriate driver routine.

User Program I/O

The user program initiates an I/O transfer by means of an EXEC call--a "JSB EXEC" as described in Section III. The supervisor recognizes the EXEC call as an I/O request and sends it along to the I/O supervisor EXEC MODULE ($EX18) which determines if the driver for the requested device is core-resident. If not, the driver is read into core from the disc.

$EX18 places the address of the EQT entry in the base page communication area (see Appendix A, *TABLES*) and transfers control to the driver. The driver configures itself to I/O operation on the appropriate channel, initiates the transfer and returns to $EX18. DOS-M either returns to the executing user program or waits until the I/O transfer is complete as requested by the program.

Interrupt Processing

When an interrupt occurs on the computer, control is transferred to the instruction in the interrupt location corresponding to the device. Each interrupt location (memory locations 6_8 through 37_8) contains a "JSB $CIC" instruction. $CIC, the central interrupt control routine of DOS-M, then performs the following:

a. Disables interrupt system
b. Saves registers, point of program suspension
c. Clears interrupt flag
d. Determines the type of interrupt
    1) If power fail, halts
    2) If memory protect, goes to EXEC (goes directly to EXEC if no memory protect)
    3) If time base, goes to CLOCK routine (if installed)
    4) If not a legal I/O channel, returns to suspension point
    5) If legal I/O channel, puts EQT entry addresses in base page communication address and transfers to driver continuation address
e. Upon return from the I/O driver, turns on interrupt system, restores registers, and returns to the point of suspension.

PLANNING I/O DRIVERS

Before attempting to program an I/O driver, the programmer should be thoroughly familiar with Hewlett-Packard computer hardware I/O organization, interface kits, computer I/O instructions and Direct Memory Access (DMA).

An I/O driver, operating under control of the Input/Output Control ($EX18) and Central Interrupt Control ($CIC) modules of DOS-M, is responsible for all data transfer between an I/O device and the computer. The device equipment table (EQT) entry contains the parameters of the transfer, and the base page communication area contains the number of the allocated DMA channel, if required.

An I/O driver includes two relocatable, closed subroutines, -- the Initiation Section and the Completion Section. If *nn* is the octal equipment type code of the device, I.*nn* and C.*nn* are the entry point names of the two sections and DVR*nn* is the driver name.

Initiation Section

The I/O control module ($EX18) calls the initiation section directly when an I/O transfer is initiated. Locations EQT1 through EQT17 of the base page communication area contain the addresses of the appropriate EQT entry. CHAN in base page contains the number of the DMA channel assigned to the device, if needed. This section is entered by a jump subroutine to the entry point, I.*nn*. On entry, the A-register contains the select code (channel number) of the device (bits Ø through 5 of EQT entry word 3). The driver returns to $EX18 by an indirect jump through I.*nn*.

Before transferring to I.*nn*, DOS-M places the request parameters from the user program's EXEC call into words 7 through 13 of the EQT entry. Word 9, CONWD, is modified to contain the request code in bits Ø through 5 in place of the logical unit. See the EQT entry diagram and Section III, *READ/WRITE EXEC CALL*, for details of the parameters.

Once initiated, the driver can use words 10 through 14 of the EQT entry in any way, but words, 1, 2, 3, 5, 6, 7, 8, 9, 15, 16 and 17 must not be altered. The driver updates the status field in word 4, if appropriate, but the rest of word 4 must not be altered.

FUNCTIONS OF THE INITIATION SECTION

The initiation section of the driver operates with the interrupt system disabled. The initiation section is responsible for those functions (as flow-charted in Figure 5-1):

1. Rejects the request and proceeds to step 5 if:
   - the device is inoperable, or
   - the request code, or other of the parameters, is illegal.
2. Configures all I/O instructions in the driver to include the select code of the device (or DAM channel). (Does not apply to DVRØ5 and DVR31.)
3. Initializes DMA, if appropriate.
4. Initializes software flags and activates the device. All variable information pertinent to the transmission must be saved in the EQT entry because the driver may be called for another device before the first operation is complete.
5. Returns to $EX18 with the A-register set to indicate initiation or rejection and the cause of the reject:

   If A = Ø, then the operation was initiated.
   If A ≠ Ø, then the operation was rejected with A set as:
   - 1 - read or write illegal for device,
   - 2 - control request illegal or undefined,
   - 3 - equipment malfunction or not ready,
   - 4 - immediate completion (for control requests).

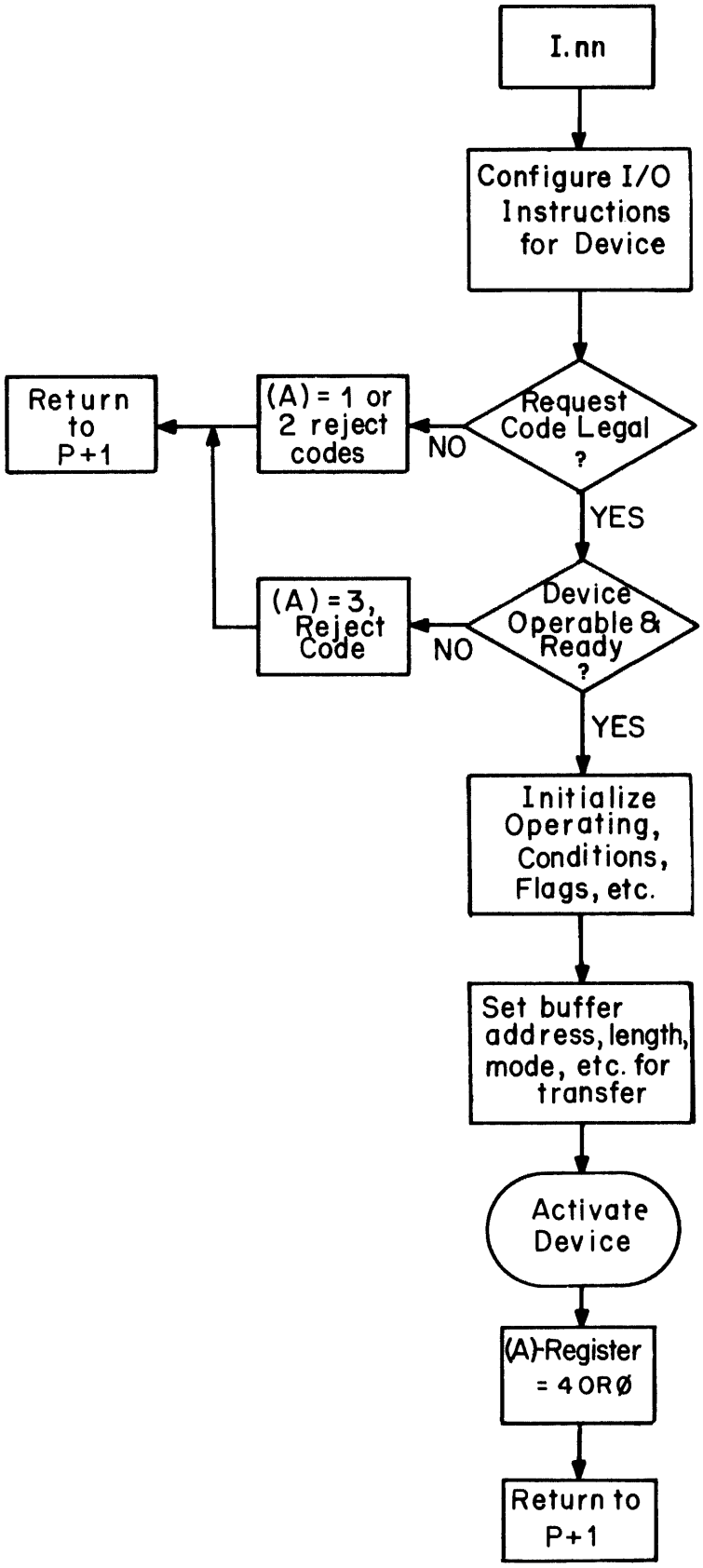


Figure 5-1. I/O Driver Initiation Section

Completion Section

DOS-M calls the completion section of the driver whenever an interrupt is recognized on a device associated with the driver. Before calling the driver, $CIC sets the EQT entry addresses in base page, sets the interrupt source code (select code) in the A-register, and clears the I/O interface or DMA flag. The interrupt system is disabled. The calling sequence for the completion section is:

Location	Action
	Set A-register equal to interrupt source code
(P)	JSB C.*nn*
(P+1)	Completion return from C.*nn*
(P+2)	Continuation return from C.*nn*

The point of return from C.*nn* to $CIC indicates whether the transfer is continuing or has been completed (in which case, end-of-operation status is returned also).

The completion section of the driver is responsible for the functions below (as flow-charted in Figure 5-2):

1. The driver configures all I/O instructions in the Completion Section to reference the interrupting device, and then proceeds to step 2.
2. If both DMA and device completion interrupts are expected and the device interrupt is significant, the DMA interrupt is ignored by returning to $CIC in a continuation return.
3. Performs the input or output of the next data item if the device is driven under program control. If the transfer is not completed, the driver proceeds to step 6.

4. If the driver detects a transmission error, it can re-initiate the transfer and attempt a retransmission. A counter for the number of retry attempts can be kept in the Equipment Table. The return to $CIC must be (P+2) as in step 6.

5. At the end of a successful transfer or after completing the retry procedure, the following information must be set before returning to $CIC at (P+1):

   a. Set the actual or simulated device status into bits Ø through 7 of EQT word 4.

   b. Set the number of transmitted words or characters (depending on which the user requested) to the B-register.

   c. Set the A-register to indicate successful or unsuccessful completion.

      Ø = successful completion,
      1 = device malfunction or not ready,
      2 = end-of-tape (information),
      3 = transmission parity error.

6. Clears the device and DMA control on end-of-operation, or sets the device and DMA for the next transfer or retry. Returns to $CIC at:

   (P+1) - completion, with the A- and B-registers set as in step 5.

   (P+2) - continuation; the registers are not significant.

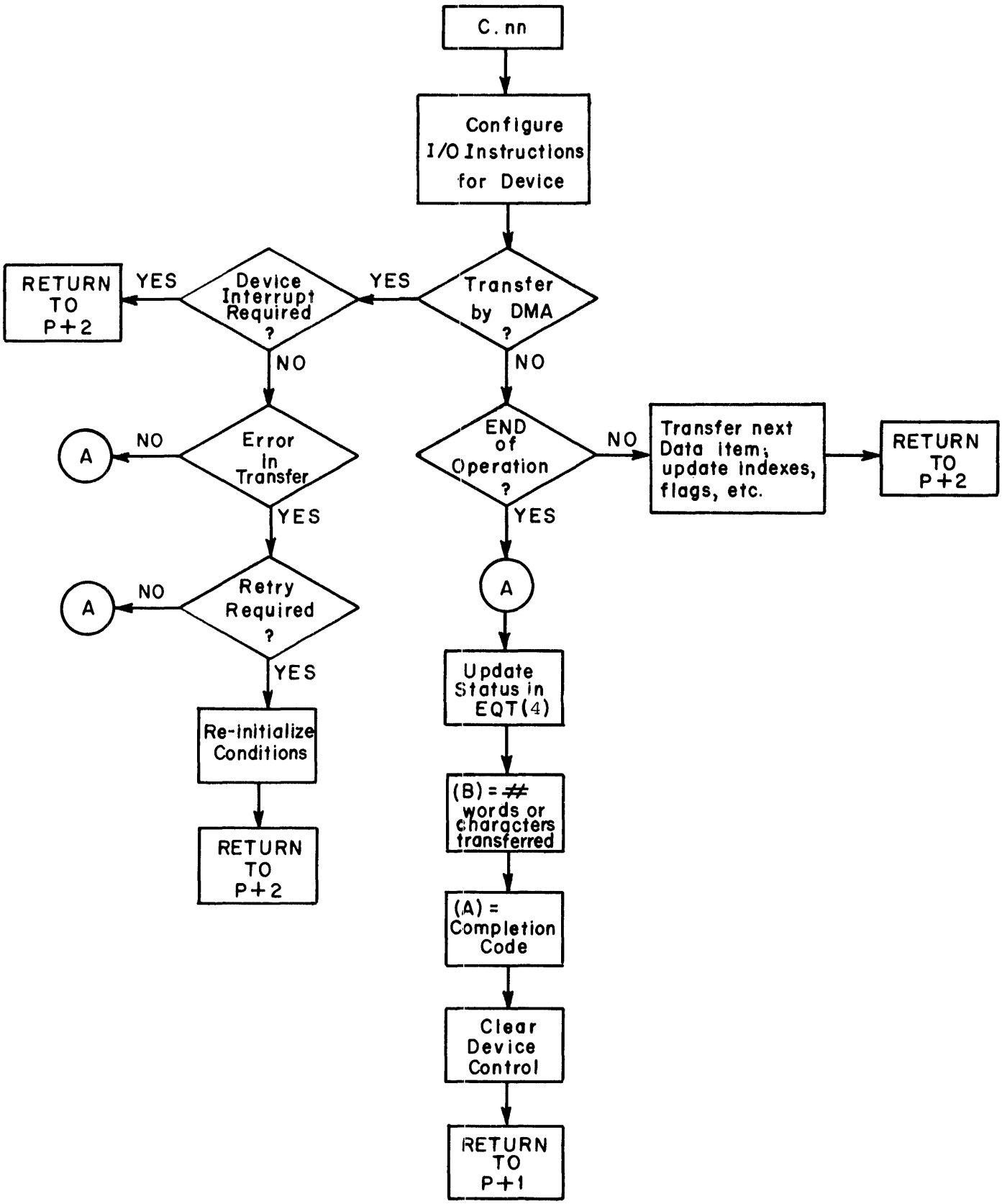


Figure 5-2. I/O Driver Completion Section

SECTION VI
INSTALLATION

Two programs are used in the installation of DOS-M: DSGEN (the DOS-M Generator) and the DOS-M Bootstrap. DSGEN is used to format new user discs and cartridges and to generate a protected system disc. The Bootstrap is used to load DOS-M into core and initiate system operation.

To install DOS-M, follow these steps:

1. Generate a system disc out of the relocatable modules of DOS-M on any active subchannel. (See "Generating DOS-M.")

2. Configure and dump several copies of the Bootstrap. (See "Configuring DOS-M Bootstrap.")

3. Format all user discs and cartridges. (See "Formatting User Discs.")

4. Initiate operation of DOS-M using Bootstrap. (See "How to Initiate DOS-M.")

CONVENTIONS USED IN THIS SECTION

Information printed on the teleprinter by the computer appears in the text as

OUTPUT EXAMPLE

Information typed on the teleprinter by the user appears in the text as

INPUT EXAMPLE

Items within input or output that appear as

VARIABLE

are variable items and stand for a class of possible entries.

The contents of the registers on the 2114 and 2116 computers (i.e., switch registers, memory data registers, etc.) appear as a series of 16 binary digits (bits) organized into octal digits:

```
0/000/000/000/000/000
↑                   ↑
Bit 15              Bit 0
```

0 means the bit is off or down (equal to binary 0).
● means the bit is on or up (equal to binary 1).
/000 represents a octal digit (e.g., /0●0 = 2_8).

For example, ●/000/0●0/000/●●●/●●● = 102077_8.

GENERATING DOS-M

DSGEN generates a DOS-M to fit a particular user's core memory size, I/O equipment, and programming needs.

To accomplish this, DSGEN requests certain information from the user. DOS-M then accepts the relocatable program modules to be included in the system, determines where they belong in core or on the disc, relocates them into absolute format, and stores them on the disc. DSGEN also creates I/O tables by identifying each I/O device and its associated driver routine, and establishing procedures for interrupt processing on each channel.

DSGEN is an absolute program, loaded into core by the Basic Binary Loader (BBL). Since DSGEN is independent of the DOS-M which it generates, the I/O operations of DSGEN require special programs called SIO Drivers.

Using other standard Hewlett-Packard software, the user can create a magnetic tape file of the relocatable program modules to speed up the program loading phase of system generation. (See *PREPARE TAPE SYSTEM,* 02116-91751.)

DSGEN operates on the same minimum configuration as that required for a DOS-M.

Operating Procedures

The operation of DSGEN involves four phases:

- INITIALIZATION PHASE. DSGEN requests specifications for the DOS-M, including description of available disc space, memory, time base generator channel, system generation code, computer identification, system and user disc subchannel, and program input devices.
- PROGRAM INPUT PHASE. DSGEN reads in the relocatable programs provided with the system and created by the user.

- PARAMETER INPUT PHASE. Parameters to change EXEC modules or drivers from disc to core-resident may be entered. The programs' NAM records are already set for a minimum core system except that DVRØØ should be changed to disc-resident. DISCM, DVR31 (moving-head disc driver) and DVRØ5 (teleprinter driver) must be core-resident.

- DISC LOADING PHASE. DSGEN requests a specification of the base page linkage, and begins loading programs onto the disc. Systems programs (i.e., the modules of DOS-M) are loaded first, after which DSGEN requests information for the equipment table, device reference table (logical unit table), and interrupt table and proceeds to load the rest of the programs onto the disc.

execute DSGEN and configure DOS-M, follow these steps:

- Turn on all equipment, set the system teleprinter to LINE, and turn on the disc protect override switch (located in the lower right-hand corner of the controller behind the front panel). For the disc drive, press POWER; place a cartridge in the slot; press DISC UNLOCK; wait for the READY light to come on.

- Load a configured DSGEN (see "configuring DSGEN") through the tape reader using the Basic Binary Reader (see "Basic Binary Loader").

- Set the switch register to $1ØØ_8$, ($0/000/000/00●/000/000_8$) press LOAD ADDRESS, then press PRESET and RUN. DSGEN begins the initialization phase.

INITIALIZATION PHASE

During the initialization phase, DSGEN requests information necessary to begin generating the DOS-M. After each question is printed, the operator responds by giving the required information terminated by a *return linefeed*. The following responses are typical. (The operator responses are only examples; actual responses should be appropriate to the particular system being generated.)

1. DSGEN requests a decimal system generation code. This code is written in the label field of the system disc for identification.....SYS GEN CODE?

   Operator responds.....*79*

2. DSGEN requests the octal channel number (select code) of the disc controller.....SYS DISC CHNL?

   Operator responds.....*10*

3. DSGEN requests the number of sectors per physical track on the disc. This is half the number of sectors per logical or software track.....# SECTORS/TRACK?

   Operator responds (usually 12).....*12*

4. DSGEN requests the number of tracks (decimal) on the system disc.....SYS DISC SIZE?

   Operator responds with a decimal number less than or equal to 200. (A response of 200 leaves three tracks as spares. A response less than 200 leaves extra tracks as spares.).....*200*

5. DSGEN requests the number of drives on the system. Each drive contains two discs.....# DRIVES?

   Operator responds with a number between 1 and 4 inclusive.....*3*

6. DSGEN requests the decimal number of the first track on the system disc which is available to DOS-M.....FIRST SYSTEM TRACK?

   Operator responds.....Ø

7. DSGEN requests the decimal number of the first sector available to DOS-M.....FIRST SYSTEM SECTOR?

   Operator responds. (The system area cannot begin before track 0, sector 3).....3

8. DSGEN requests the subchannel number of the system disc.....SYS DISC SUBCHNL?

   Operator responds with a number between 0 and 7.....Ø

9. DSGEN requests the subchannel number of the user disc. (This may be the same as the system disc.).....USER DISC SUBCHNL?

   Operator responds with a number between 0 and 7. (System efficiency increases if the user disc is on a different drive from the system disc.).....2

10. DSGEN requests the I/O channel (select code) of the Time Base Generator.....TIME BASE GEN CHNL?

    Operator responds with an octal number or Ø if the time base is not present.....Ø

11. DSGEN asks whether the computer is a 2114 or not...IS 2114?

    Operator responds with YES or NO.....YES

12. DSGEN requests the last word of available core memory in octal.....LWA MEM?

    Operator responds.....*17677*

13. DSGEN asks whether :SS directives are to be allowed in the system.....ALLOW :SS?

    Operator responds either YES or NO.....*YES*

14. DSGEN requests the type of input unit for relocatable program modules ..... PRGM INPT?

    Operator responds with PT (for paper tape), TY (for teleprinter), or MT (for magnetic tape; see PTS manual, 02116-91751).....*PT*

15. DSGEN requests the type of input unit for relocatable library programs.....LIBR INPT?

    Operator responds with PT, TY, or MT.....*MT*

16. DSGEN requests the type of input unit for parameters, describing the relocatable programs.....PRAM INPT?

    Operator responds with PT or TY.....*TY*

When DSGEN finishes the initialization phase, the computer halts.

PROGRAM INPUT PHASE

During the program input phase, DSGEN accepts relocatable programs from the Program Input Unit and Library Input Unit specified during the initialization phase. The operator selects the input device by setting switch register bits Ø-1 ($ØØ_2$ for the Program Input Unit, or $1Ø_2$ for the Library Input Unit), and places the programs in the input device. Main programs must enter prior to their segments. DISCM must be the first module loaded.

The suggested order of tape input is:

DOS-M CORE-RESIDENT SYSTEM (DISCM)
DOS-M DISC-RESIDENT EXEC MODULES ($EXØ1 THRU $EX2Ø)
DOS-M I/O DRIVERS (DVRØ5, DVRØ1,...ETC)
DOS-M JOB PROCESSOR/FILE MANAGER (JOBPR)
DOS-M RELOCATING LOADER (LOADR)
DOS-M ASSEMBLER (MAIN CONTROL, SEGMENTD, SEGMENT1,....)
DOS-M FORTRAN (MAIN CONTROL, PASS 1,...)
RTE/DOS ALGOL
RTE/DOS FORTRAN IV LIBRARY OR RTE/DOS BASIC FORMATTER
RTE/DOS RELOCATABLE PROGRAM LIBRARY (EAU OR NON-EAU)
Any relocatable user programs to be made a permanent part of DOS-M.

NOTE: If the FORTRAN IV Library is to be included in an 8K system, certain rules must be followed:

1. *The system must be generated without any compilers or an assembler.*
2. *A magnetic tape SIO driver cannot be used with DSGEN.*
3. *The compilers and assembler must be loaded into the system during operation (using the Loader).*

The operator presses RUN. When entering paper tape, the message "*EOT" is printed whenever an end-of-tape occurs. The computer halts.

At this point, the operator has several options: additional programs can be input from the same device by repeating the steps above; input can be switched to the other input device (by setting the switch register bits to $\emptyset\emptyset_2$ or $1\emptyset_2$).

To terminate the program input phase, the user must set switch register bits to $\emptyset 1_2$, and press RUN. If there are no undefined externals, this message is printed on the system teleprinter:

NO UNDEF EXTS

If there are undefined externals, the following message is printed:

UNDEF EXTS

The externals are listed one per line and the computer halts. External references are satisfied by loading more programs. The user must set switch register bits to $\emptyset\emptyset_2$ (for Program Input Unit) or $1\emptyset_2$ for the Library Input Unit) and press RUN. If the externals are to be left unsatisfied, set the switch register bits to $\emptyset 1_2$ and press RUN.

PARAMETER INPUT PHASE

During the parameter input phase, the operator can change some modules from disc to core-resident. (If an I/O driver is changed from disc-resident--type 4-- to core-resident--type Ø--the associated EQT entry must include the R parameter.) Since DVRØØ is a DOS driver, it is distributed as a core-resident driver; it should be changed to disc-resident if DVRØ5 is included in DOS-M. Any unnecessary I/O drivers must be eliminated at this time.

Each parameter record is of this general form:

name,type

where *name* is the name of the program
type is the program type code;

- Ø - System core-resident
- 1 - System disc-resident EXEC modules
- 3 - User disc-resident main
- 4 - Disc-resident I/O driver
- 5 - User segment
- 6,7 - Library
- >7 - Program deleted from the system

EXEC modules and drivers that are often used may be changed from disc to core-resident. The functions of the EXEC modules are:

Module Name		Function
$EXØ1	-	Disc Work Tracks Status
$EXØ2	-	Disc Work Track Limits
$EXØ3	-	Program Completion
$EXØ4	-	Program Suspension
$EXØ5	-	Program Segment Load
$EXØ6	-	User File Name Search
$EXØ7	-	Current Time Processor
$EXØ8	-	Real-Time Disc Allocation. (See Appendix D.)
$EXØ9	-	Execution Time :EQ Processor
$EX1Ø	-	Load and Execute Program

Module Name		Function
$EX11	-	System File Name Search
$EX12	-	System Startup
$EX13	-	Error Message Processor
$EX14	-	Execution Time, :UP, :DN, :LU Processor
$EX15	-	Abort and Post Mortem Dump
$EX16	-	:GO Parameter Processor
$EX17	-	:UD Processor
$EX18	-	I/O Initiation Processor
$EX19	-	:IN Processor
$EX2Ø	-	Disc Parity Error Processor

When EXEC modules are made core-resident, certain associated subroutines must also be changed to be core-resident. Several EXEC modules use $ADDR:

$EXØ1
$EXØ2
$EXØ6
$EXØ7
$EXØ8

The following EXEC modules use $LBL:

$EX17
$EX19

The following EXEC modules use $SRCH:

$EXØ5
$EXØ6
$EX11

These EXEC modules use ASCII:

$EXØ4
$EXØ9
$EX13
$EX14
$EX15
$EX2Ø

To end the parameter input phase and continue on to the disc loading phase, the operator enters "/E" instead of a parameter record.

DISC LOADING PHASE

DSGEN asks two questions before entering DISC LOADING PHASE.

1. DSGEN requests the estimated number of system linkages required in base page.....# SYSTEM LINKS?

   Operator responds with a decimal number. (The more modules that are core-resident the more links are needed, 1ØØ should be the minimum response.).....*1ØØ*

2. DSGEN requests the estimated number of user linkages required in base page.....# USER LINKS?

   Operator responds with a decimal number. (Since the loader requires approximately 32Ø linkages, 32Ø should be the minimum number entered.).....*32Ø*

NOTE: If the system requires more linkages than you have assigned, it takes them away from the user link area. If the total of the two responses overflows base page (>677$_{10}$), the questions are repeated.

Figure 6-1 shows the relative location of the various core areas. Loading of the absolute, resident supervisor begins after the establishment of the user and system linkage areas. As each program is loaded, DSGEN prints a memory map giving the starting locations and, if switch register bit 15 is up, the entry points for all main programs and subroutines. (Subroutines are indented two spaces, and entry point addresses are preceded by an asterisk.)

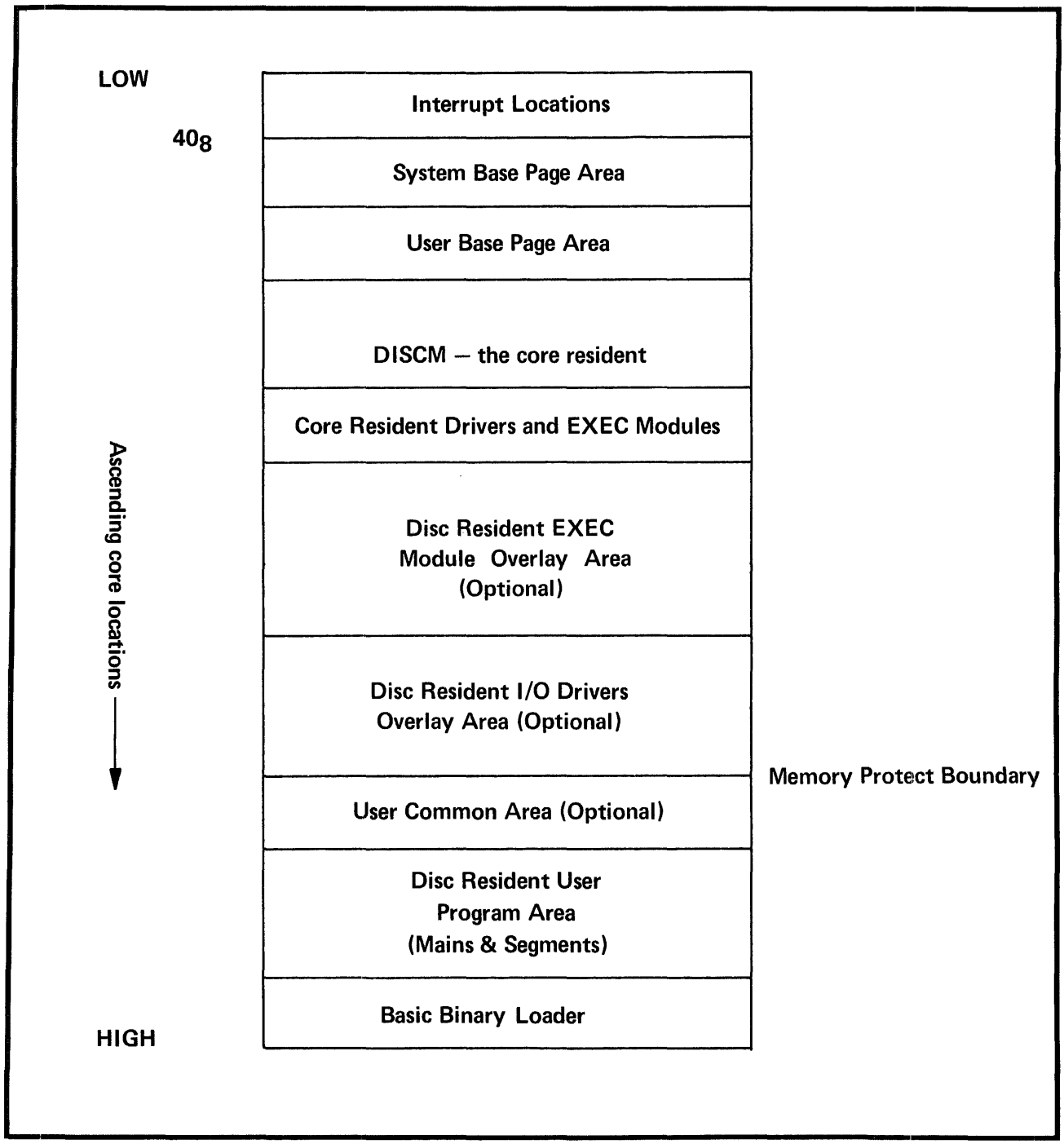


Figure 6-1. Core Allocations in DOS-M

Input/Output Tables

Next, DSGEN generates the three I/O tables: equipment table, device reference table (logical unit table), and the interrupt table.

3. DSGEN requests the equipment table entries.....EQUIPMENT TABLE E

   Operator responds with a series of one line EQT entries, which are assigned EQT numbers sequentially from one as they are entered. The EQT entry relates the EQT number to an I/O channel and driver, in this format.....*n1,DVRnn [,D*

where *n1* is the I/O channel (lower number if multi-board),
DVRnn is the driver name (*nn* is the equipment type code).
D, if present, means DMA channel required,
R, if present, means driver is core-resident (must be ty
U, is the physical subchannel number.

Operator terminates the equipment table entries by typing.....*/E*

Here is a sample Equipment Table:

```
* EQUIPMENT TABLE ENTRY
10,DVR31,D,R      (EQT entry #1 = disc)
12,DVR22,D        (EQT entry #2 = magnetic tape)
14,DVR05,R        (EQT entry #3 = special teleprinter)
15,DVR01          (EQT entry #4 = photoreader)
16,DVR02          (EQT entry #5 = tape punch)
17,DVR12          (EQT entry #6 = line printer)
/E                (End of table)
```

4. DSGEN requests the logical unit assignments for
the device reference table.....DEVICE REFERENCE TABLE?

For each logical unit number, DSGEN prints.....*n=EQT#?*

Operator responds with an EQT entry number (*m*)
appropriate to the standard definition of *n*.
Numbers above 6 may be assigned any EQT entry
desired.....*m*

Operator terminates entry by typing...../E

Here is a sample Device Reference Table:

```
* DEVICE REFERENCE TABLE
1     = EQT #?    (System teleprinter on channel 14, EQT #3
3
2     = EQT #?    (User disc on channel 10, EQT #1)
1
3     = EQT #?    (System disc on channel 10, EQT #1)
1
4     = EQT #?    (Standard punch unit on channel 16, EQT #5)
5
5     = EQT #?    (Standard input unit on channel 15, EQT #4)
4
6     = EQT #?    (Standard list unit on channel 17, EQT #6)
6
7     = EQT #?    (Standard unit definable by user)
2
8     = EQT #?    (End of table)
/E
```

NOTE: The number of responses given here determines the number of logical units allowed in the system. To allow unassigned logical units for the user, respond with a Ø to as many questions as units are desired.

5. DSGEN requests the interrupt table entries.....INTERRUPT TABLE

   Operator responds with an entry for each I/O location which may interrupt, in ascending order, and in this format.....*n1, n2*

where *n1* is the octal channel number (high number for 2 board interfaces) between 10_8 and 37_8 inclusive (must be entered in ascending order), and

n2 is a decimal EQT entry number.

Operator terminates entry by typing...../E

Here is a sample Interrupt Table:

```
* INTERRUPT TABLE
11,1          (Channel 11 linked to EQT #1)
13,2          (Channel 13 linked to EQT #2)
14,3          (Channel 14 linked to EQT #3)
15,4          (Channel 15 linked to EQT #4)
16,5          (Channel 16 linked to EQT #5)
17,6          (Channel 17 linked to EQT #6)
/E            (End of table)
```

NOTE: The EQT numbers need not appear in numerical order. This order is determined by referring back to the Equipment Table. The octal channel numbers, however, must be in ascending sequence.

Following the completion of the I/O tables, DSGEN loads the disc-resident executive modules (if any), and the disc-resident I/O drivers (if any).

6. DSGEN reports the last address of the supervisor.....LWA SYS *xxxxx*

7. DSGEN requests the first word address of the user program area.....FWA USER?

   Operator responds with an octal address greater than *XXXXX*. In an 8K computer, this response must be less than or equal to $72ØØ_8$. (This option is provided so that user programs can start on a page boundary, if desired).....*nnnnn*

   DSGEN proceeds to load all user main programs and segments onto the disc with memory map listings as described for system programs.

8. When system generation is complete, DSGEN prints... *SYSTEM STORED ON DISC

   DSGEN then reports the last track used in bits 15 through 8 of the A-register, and the last sector used in bits 7 through Ø of the A-register.

   The disc protect override switch should be turned off.

Restart

During any of the phases, DSGEN can restart that phase if any error occurs. The operator sets the switch register equal to $1ØØ_8$, and presses LOAD ADDRESS and RUN.

HOW TO INITIATE DOS-M

1. Load a configured Bootstrap with BBL. (See "BOOTSTRAPPING DOS-M.")
2. Set the SWITCH REGISTER equal to $10\emptyset_8$ (0/000/000/00●/000/000 = $10\emptyset_8$).
3. Press LOAD ADDRESS.
4. Set switches 2 through 0 equal to the octal subchannel of the system disc. (If this subchannel is not the same as the subchannel that was specified when the system was generated, the new subchannel overrides the old.)
5. Press PRESET.
6. Press RUN.

DOS-M then prints the following message until the operator types a valid DATE directive (see Section II):

```
INPUT :DATE,XXXXXXXXXX          (No time base generator)
          or
INPUT :DATE,XXXXXXXXXX,H,M      (Time base present)
```

Following the DATE directive, the only valid directives are TRACKS, BATCH, TYPE and JOB. All others are ignored until a JOB directive is given.

FORMATTING USER DISCS

The DOS-M DSGEN is used to format a new user disc or cartridge. This must be done anytime a new disc is added or an old system disc is reused as a user disc. (This special process need not be done for the system disc because DSGEN will format it during system generation.)

The formatting process involves assigning a system generation code, reading every sector, clearing any existing user directory, clears any protected or disabled sectors, etc. The result is an unlabeled user disc ready for use in DOS-M.

Operating Procedures: FORMATTING

1. Turn on all equipment; for the disc drive, press POWER; place a cartridge in the slot; press LOAD; wait for the READY light to come on.
2. Turn on the Disc Protect Override Switch. (This switch is located in the lower right hand corner of the controller.)
3. Load a configured DSGEN (see "DSGEN Configuration") through the tape reader using the Basic Binary Loader (see "Basic Binary Loader").
4. Set the SWITCH REGISTER equal to $1\emptyset\emptyset_8$ (0/000/000/00●/000/000 = $1\emptyset\emptyset_8$) and press LOAD ADDRESS.
5. Set SWITCH REGISTER bit 15 on or up (●/000/000/000/000/000).
6. Press PRESET.
7. Press RUN.
8. DSGEN asks for a decimal number to be written on the disc label. This number is used for identification.....SYS GEN CODE?

   Operator responds with a decimal number....79

9. DSGEN requests the octal select code of the disc controller..... SYS DISC CHANNEL?

   Operator responds with an octal number.....*10*

10. DSGEN asks the number of sectors per hardware track on the disc (this is half the number of sectors on a software track).....# SECTORS/TRACK

    Operator responds with a decimal number....*12*

11. DSGEN requests the subchannel number (0 to 7) of the user disc to be formatted.....USER DISC SUBCHANNEL?

    Operator responds with a number between 0 and 7 inclusive.....*3*

12. DSGEN requests that the disc protect override switch be turned on (if it is not already) .....TURN ON DISC PROTECT OVERRIDE - PRESS RUN

    Operator turns on this switch and presses RUN.

13. DSGEN carries out formatting on the specified subchannel and halts with 102007_8 in the Memory Data Register (●/000/0●0/000/000/●●● = 102007_8).

    Operator can press RUN to format a new disc (switch 15 must still be up). DSGEN repeats from USER DISC SUBCHANNEL?

    Operator can set switch 15 down or off and press RUN to proceed to system generation.

This procedure should be followed for each proposed user disc.

CONFIGURING DSGEN

In order to use DSGEN, a version of DSGEN that is configured for a particular input/output arrangement must reside in core memory. This can be done in three ways:

1. Load DSGEN with BBL.
   Load and configure drivers with BBL.
   Ready

2. Load a configured DSGEN tape with BBL.
   Ready

   To produce a configured DSGEN:
   Load DSGEN.
   Load and configure drivers.
   Load and RUN the SIO System Dump.
   A configured tape is punched.

3. Load DSGEN.
   Load a configured driver tape.
   Ready.

   To produce a configured driver tape:
   Load and configure drivers.
   Load and RUN the SIO System Dump.
   A configured tape is punched.

Operating Procedures: Configuring DSGEN

1. Turn on all equipment.
2. Load DSGEN with the BBL, if desired.
3. Load and configure the SIO Teleprinter Driver. (See next page.)
4. Load and configure the SIO Tape Reader Driver and the SIO Magnetic Tape Driver, if desired. (See next page.)

5. Load SIO System Dump if a permanent copy is desired:

   a. Load SIO System Dump with BBL.

   b. Set the SWITCH REGISTER to 2_8 (0/000/000/000/000/0●0 = 2_8).

   c. Press LOAD ADDRESS.

   d. Set SWITCH REGISTER bit 15 to 1 (on) for a tape containing both DSGEN and the drivers or to 0 (off) for a tape containing only the drivers.

   e. Turn on the punch unit.

   f. Press RUN.

   g. The computer halts with the MEMORY DATA REGISTER set to 102077_8 when completed. Press RUN for another copy.

Loading SIO Drivers

1. Load the SIO Driver tape with BBL.
2. Set the SWITCH REGISTER to 2_8.
3. Press LOAD ADDRESS.
4. Set the I/O address (higher priority select code, lower number) of the device whose driver is being configured into SWITCH REGISTER bits 5 through 0. (For the teleprinter driver only, set switch 15 to 0 (off) for a model 2752 or 1 (on) for a 2754. Bits 14 through 6 should be set to 0 (off)).
5. Press PRESET.
6. Press RUN.

CONFIGURING DOS-M BOOTSTRAP

Once DOS-M has been generated onto a disc, it can be initiated using DOS-M Bootstrap.

The DOS-M Bootstrap must be configured before being used.

Follow these steps:

1. Load the SIO Tape Punch Driver or the SIO Teleprinter Driver with BBL.
   a. Set the SWITCH REGISTER equal to 2_8 (0/000/000/000/000/0●0 = 2_8).
   b. Press LOAD ADDRESS.
   c. Set switches 5 through 0 equal to the octal select code for the punch or teleprinter.
   d. Press RUN.
2. Load Bootstrap with BBL.
   a. Set the SWITCH REGISTER equal to 2_8 (0/000/000/000/000/0●0 = 2_8).
   b. Press LOAD ADDRESS.
   c. Set switches 5 through 0 equal to the octal select code of the disc controller. (Low number, high priority)
   d. Set switch 15 up (on) to punch a configured Bootstrap.
3. Press RUN. After Bootstrap punches the tape, it halts. Press RUN for another copy.

If the computer halts with the MEMORY DATA REGISTER equal to 102011_8 (●/000/0●0/000/00●/00● = 102011_8), a disc error has occurred. The disc status is in the A-register.

BASIC BINARY LOADER

To load a paper tape:

1. Turn on all equipment.
2. Place the tape in the reader.
3. For the 2114, press LOAD.
   For the 2116, set the SWITCH REGISTER equal to the starting address of BBL (017700 for 8K, 037700 for 16K, 077700 for 32K) and press LOAD ADDRESS; set the 2116 LOADER switch to ENABLED, press PRESET and RUN.
4. The BBL should read the paper tape and halt with 102077 (0/000/000/000/000/000 = 102077_8) in the T-register. For the 2116, set the LOADER switch to PROTECTED.
5. If the computer halts with 102011 (0/000/000/000/000/000 = 102011) in the T-register, a checksum error has occurred. If the computer halts with 102055 (0/000/000/000/000/000 = 102055_8) in the T-register, an illegal address has been read. Check for a tear in the paper tape and clean any dust out of the photoreader with an air brush. Then reposition the tape and reread the record. If this does not work, restart at step 1.

ERROR MESSAGES

The following messages may be printed on the teleprinter during execution of DSGEN:

Message	Meaning	Action
Messages During Initialization and Input Phase		
ERRØ1	Invalid response to initialization request.	Request is repeated. Enter valid reply.
ERRØ2	Checksum error on program input.	Computer halts; reposition tape to beginning of record and press RUN to reread.
ERRØ3	Record out of sequence.	Same as ERRØ2.
ERRØ4	Illegal record type.	Same as ERRØ2.
ERRØ5 *name*	Duplicate entry point.	Revise program by reloading the entry points (the current entry point replaces the previous entry point).
ERRØ6	Invalid base page length (must be zero).	Base page area is ignored, but memory protect error will occur if program is executed.
ERRØ7	Program name or entry point table overflow of available memory.	Irrecoverable error. Revise or delete programs.
ERRØ8 *name*	Duplicate program name.	The current program replaces the previous program.

Message	Meaning	Action
Messages During the Parameter Phase		
ERRØ9	Parameter name error (no such program).	Enter valid parameter statement.
ERR1Ø	Parameter type error.	Same as ERRØ9.
General Messages		
ERR13	User segment precedes user main program.	Irrecoverable.
ERR15	More than 63 subprograms called by a main program.	Revise main program (subsequent calls to subprograms are ignored).
ERR16	Base page linkage overflow.	Diagnostic printed for each word required. Revise order and composition of program loading to reduce linkage requirements.
ERR17	Current disc address exceeds number of available tracks.	Irrecoverable error.
ERR18	Memory overflow (absolute code exceeds LWA memory).	Diagnostic printed for each word required (absolute code is generated beyond LWA). Revise program.
ERR19	Program overlay (current word of absolute code has identical location to previous).	Current word is ignored (the address is printed).
ERR2Ø	Binary DBL record overflow of internal table.	Records overlay previous DBL records (diagnostic printed for each overflow record). Revise program.

Message	Meaning	Action
ERR21	Module containing entry point $CIC not loaded.	Irrecoverable error.
ERR22	Read parity/decode disc error. A-register bits 8-14 show track number; bits Ø-7 show sector number.	After ten attempts to read or write the disc sector, the computer halts. To try ten more times, press RUN
ERR23	EQT not entered for disc-resident I/O module.	Restart at $4ØØØ_8$.

Messages During I/O Table Entry

ERR24	Invalid channel number.	Enter valid EQT statement
ERR25	Invalid driver name or no driver entry points.	Same as ERR24.
ERR26	Invalid or duplicate D,R,U operands.	Same as ERR24.
ERR27	Invalid logical unit no.	Enter valid DRT statement.
ERR28	Invalid channel number.	Enter valid INT statement.
ERR29	Channel number decreasing.	Same as ERR28.
ERR31	Invalid EQT number.	Same as ERR28.
ERR35	Base page interrupt locations overflow into linkage area.	Restart Disc Loading Phase.
ERR36	Invalid number of characters in final operand.	Same as ERR28.

APPENDIX A
TABLES

Appendix A contains several useful tables and figures.

DOS-M BASE PAGE CONSTANTS

LOCATION	TYPE	VALUE
4Ø	DEC	-64
41	DEC	-1Ø
42	DEC	-9
43	DEC	-8
44	DEC	-7
45	DEC	-6
46	DEC	-5
47	DEC	-4
5Ø	DEC	-3
51	DEC	-2
52	DEC	-1
53	DEC	Ø
54	DEC	1
55	DEC	2
56	DEC	3
57	DEC	4
6Ø	DEC	5
61	DEC	6
62	DEC	7
63	DEC	8
64	DEC	9
65	DEC	1Ø
66	DEC	17
67	DEC	64
7Ø	OCT	17
71	OCT	37
72	OCT	77

LOCATION	TYPE	VALUE
73	OCT	177
74	OCT	377
75	OCT	1774ØØ
76	OCT	3777
77	OCT	1777ØØ

DOS-M BASE PAGE SYSTEM COMMUNICATION AREA

LOCATION	NAME	CONTENTS
1ØØ	UMLWA	Last word address of user available memory
1Ø1	JBINS	Start track/sector of job binary area
1Ø2	JBINC	Current Track/sector of job binary area
1Ø3	TBG	Time base generator I/O channel address
1Ø4-5	CLOCK	Current system clock time
1Ø6-7	CLEX	Execution clock time
11Ø	CXMX	Maximum allowable execution time
111	BATCH	Logical unit # of batch input device
112	SYSTY	Logical unit # of system teletype
113	DUMPS	Abort/Post Mortem dump flag
114	SYSDR	System directory track/sector
115	SYSBF	System buffer track/sector
116	SECTR	Number of sectors/disc track
117	EQTAB	First word address of Equipment Table
12Ø	EQT#	Number of Equipment entries
121	LUTAB	First word address of Logical Unit table
122	LUT#	Number of Logical Unit entries
123	JBUF	Job input buffer address
124	JFILS	Source file starting track/sector
125	JFILC	Source file current track/sector
126-4Ø	RONBF	User area file name information
141-53	EXPG	Directory entry for current program

LOCATION	NAME	CONTENTS
154	DISCO	Disc I/O channel/last track on disc
155	SYSSC	System subchannel
156	SCCNT	Number of subchannels on system
157	UDNTS	Next user disc track/sector
16Ø	SYNTS	Next system disc track.sector
161	CUISC	Current user disc subchannel
162	CRFLG	Current disc request flag: Ø for system, not Ø for user
163	CUILA	Current user disc last access
164	SDLA	System disc last access
165	CUMID	Computer identification
166-7Ø	DBUFR	System disc triplet parameter buffer
171-73	UBUFR	User disc triplet parameter buffer
174	TSONE	Last track/sector referenced +1
175	GUISC	Default user disc subchannel
176	SYSCD	System generation code
177	JFLSC	Source file subchannel
2ØØ	DISCL	User label track/sector
2Ø1	INTAB	First word address of interrupt table
2Ø2	INT#	Number of interrupt entries
2Ø3	EQT1	EQT1-EQT17 are addresses of current Equipment Table entry
2Ø4	EQT2	
2Ø5	EQT3	
2Ø6	EQT4	
⋮	⋮	
223	EQT17	
224	RQCNT	Number of request parameters.
225	RQRTN	Current request return address
226	RQP1	RQP8-RQP8 are addresses of current request parameters
⋮	⋮	
235	RQP8	

LOCATION	NAME	CONTENTS
236	NABRT	Illegal request code abort/no abort option
237	XA	A register contents at time of interrupt
24Ø	XB	B register contents at time of interrupt
241	XEO	E and O register contents at time of interrupt
242	XSUSP	Point of suspension at time of interrupt
243	EXLOC	Address of Exec module doublet table
244	EX#	Number of Exec module doublet table entries
245	EXMOD	Exec module # currently in Exec module overlay area
246-47	EXMAN	Exec module low and high main core addresses
25Ø-51	EXBAS	Exec module low and high base page core addresses
252	IODMN	First word address of I/O driver module main area
253	IODBS	First word address of I/O driver module base page area
254	UMFWA	First word address of user main area
255	UMFWA	First word address of user base page area
256	UBLWA	Last word address of user base page area
257	CHAN	Current DMA channel number
26Ø	OPATN	Operator/keyboard attention flag
261	OPFLG	Operator communication flag
262	SWAP	Job processor resident flag
263-65	JOBPM	Job processor disc address/number of words in main
265	JOBPB	Job processor base page number of words
266	TBSYG	Track/sector address of system track table
267	RTRK	Real time simulation track number
27Ø	$BUF	System input/output buffer
47Ø	$GOPT	Point of suspension continuation address
471	$IDCD	Input request code check
472	$MDBF	Exec module data buffer

LOCATION	NAME	CONTENTS
474	TEMP	System temporary
5Ø3	TEMPØ	System temporary
5Ø4	TEMP1	System temporary
5Ø5	TEMP2	System temporary
5Ø6	TEMP3	System temporary
5Ø7	TEMP4	System temporary
51Ø	TEMP5	System temporary
511	MSECT	Negative number of sectors/track
512	VADR	Address of instruction causing memory protect violation
513	IODMD	Current resident I/O driver module flag
514	RCODE	Current request code value
515	SXA	Operator attention restore A register value
516	SXB	Operator attention restore B register value
517	SXEO	Operator attention restore E and O register value
52Ø	SXSUS	Operator attention return address
521	SEQT1	Operator attention restore EQT table address
522	DSCLB	Disc track/sector of relocatable library
523	DSCL#	Number of relocatable library routines
524	LSTCH	Last disc referenced
525	FLFLG	User file table validity flag
526	XFLG	Entry address for disc not ready
527	SSFLG	System search flag
53Ø-31	CHARC	System temporary

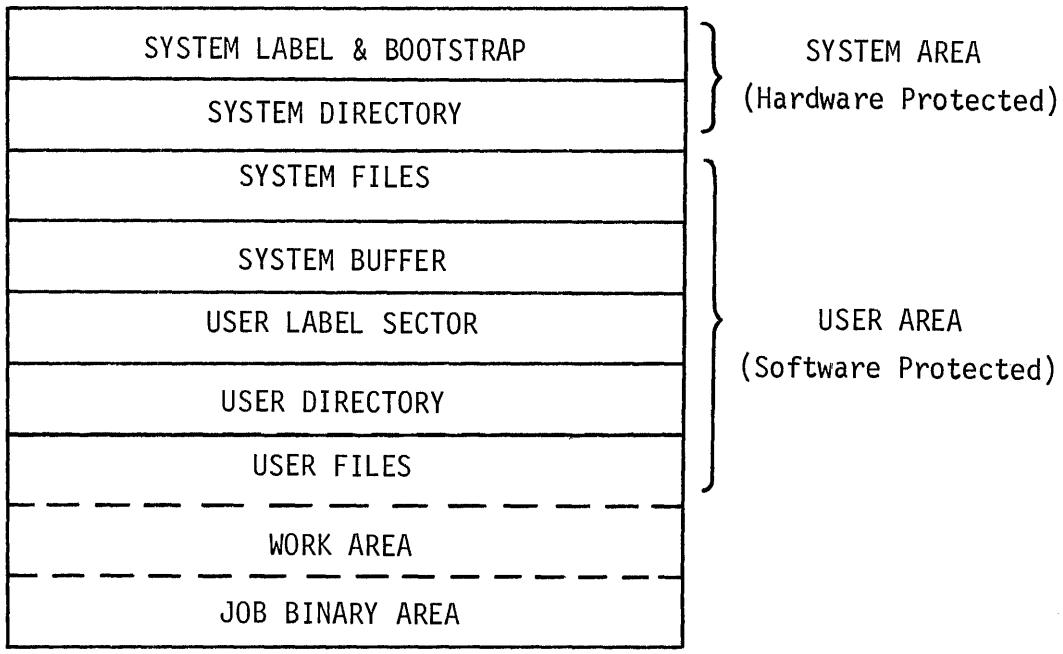


Figure A-1. General Disc Layout

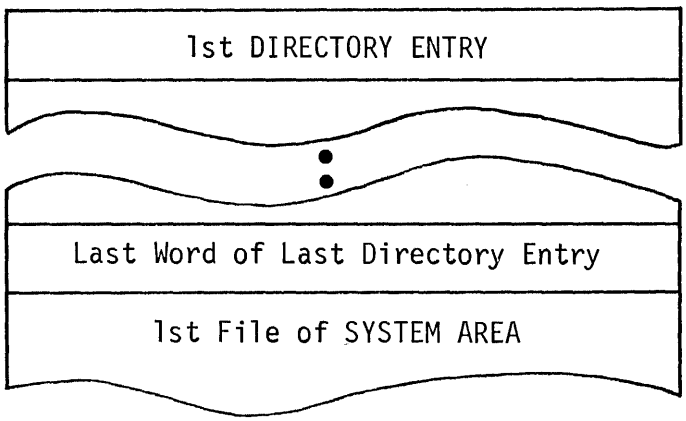


Figure A-2. System Directory Format

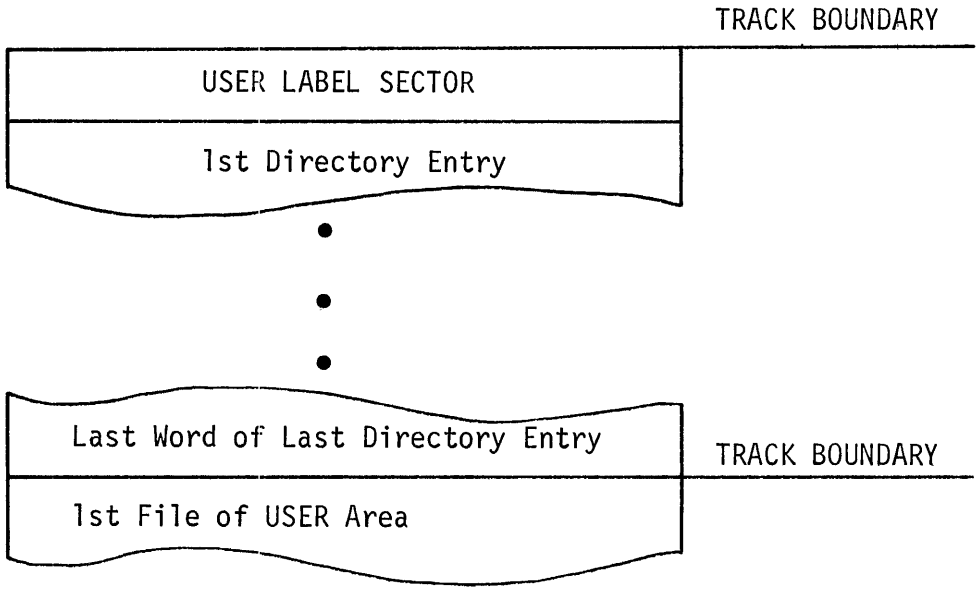


Figure A-3. User Directory Format

Word 1	F	N		
Word 2	A	M		
Word 3	E	P	Entry Type	
Word 4	Track	Sector		
Word 5	File Length (in sectors)			
Word 6	FWA Program			For System Generated Binary Programs Only
Word 7	LWA Program			
Word 8	FWA Base Page Linkage Area			
Word 9	LWA Base Page Linkage Area			
Word 10	Program Entry Point			
Word 11	FWA of LIB routine section			

The 1st five characters (in Words 1 through 3) contain the File

The lower character in Word 3 contains the Type Code and 'P' bit, as shown below.

Figure A-4. Directory Entry Format

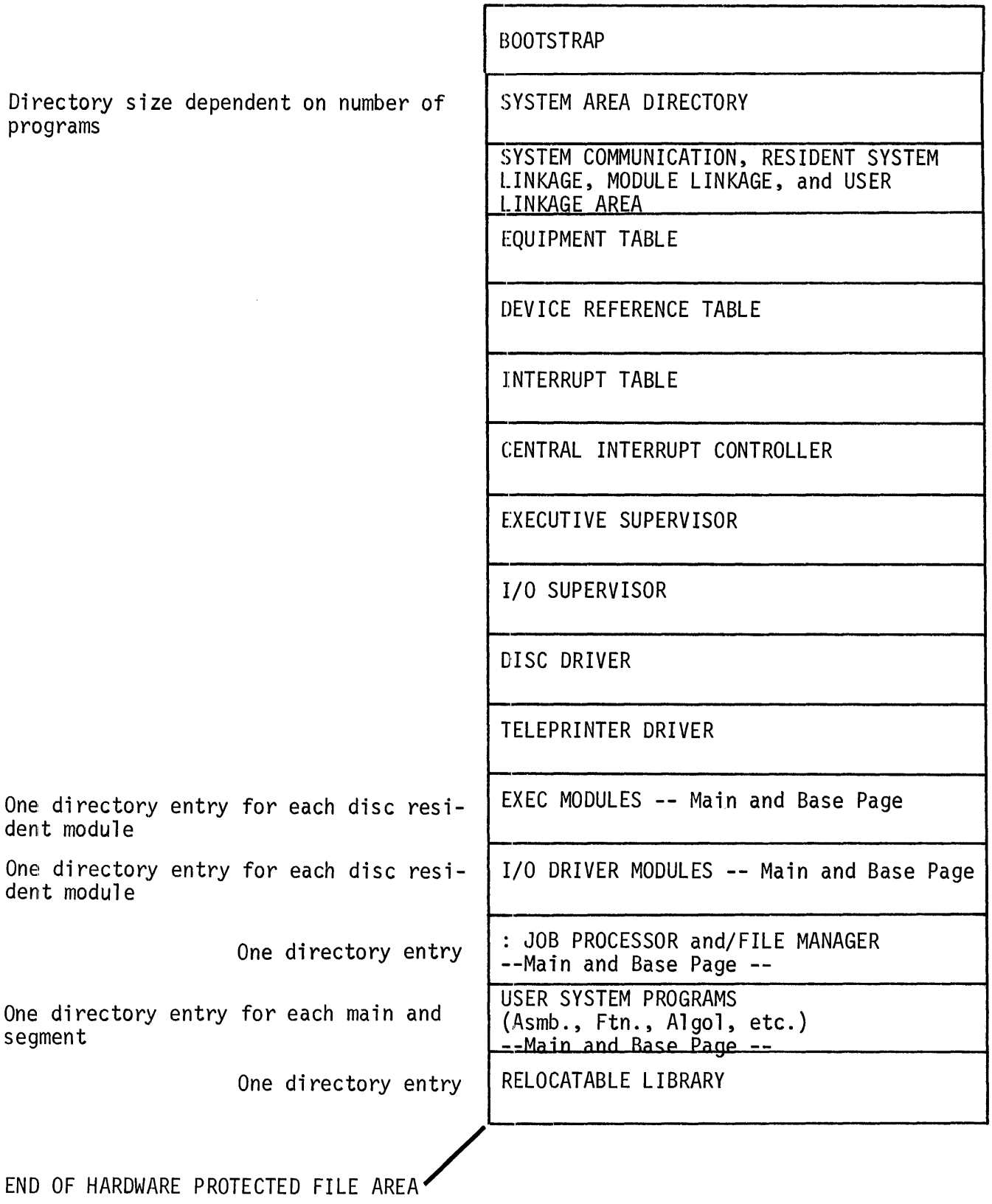
TYPE	FILE
Ø	System Resident
1	Disc Resident Executive Supervisor Module
2	Reserved for System
3	User Program, Main
4	Disc Resident Device Driver
5	User Program, Segment
6,7	Library
10_8	Relocatable Binary
11_8	ASCII Source Statements
12_8	Binary Data
13_8	ASCII Data

'P' Bit

Ø = No Action

1 = Purge this entry at the end of the JOB. This bit is set by the LOADER and cleared by a :STORE,P[,*file-name*] request

The last directory entry in each sector is followed by a word containing '-1'. The last entry in the directory is followed by a word containing zero.


BOOTSTRAP
Directory size dependent on number of programs
SYSTEM AREA DIRECTORY
SYSTEM COMMUNICATION, RESIDENT SYSTEM LINKAGE, MODULE LINKAGE, and USER LINKAGE AREA
EQUIPMENT TABLE
DEVICE REFERENCE TABLE
INTERRUPT TABLE
CENTRAL INTERRUPT CONTROLLER
EXECUTIVE SUPERVISOR
I/O SUPERVISOR
DISC DRIVER
TELEPRINTER DRIVER
One directory entry for each disc resident module
EXEC MODULES -- Main and Base Page
One directory entry for each disc resident module
I/O DRIVER MODULES -- Main and Base Page
One directory entry
: JOB PROCESSOR and/FILE MANAGER
--Main and Base Page --
One directory entry for each main and segment
USER SYSTEM PROGRAMS
(Asmb., Ftn., Algol, etc.)
--Main and Base Page --
One directory entry
RELOCATABLE LIBRARY
END OF HARDWARE PROTECTED FILE AREA

APPENDIX B
TYPICAL JOB DECKS

ASSEMBLE A PROGRAM AND STORE IN FILE

```
:JOB,ASMBS
:PROG,ASMB,5,6,4,56,99
ASMB,B,L
      NAM TEST,3
       :
      END ENTER
:STORE,R,AFILE
:JOB,NEXT JOB
```

Source Program

LOAD AND EXECUTE A RELOCATABLE FILE

```
:JOB,LOADE
:PROG,LOADR,2
AFILE,1E
:STORE,P,TEST
:RUN,TEST
1Ø
23
 :
51
:JOB,NEXT JOB
```

Data

STORE, EDIT, COMPILE, LOAD AND RUN A PROGRAM

```
:JOB,EVERY
:STORE,S,SOURC,5
FTN,B,L                    ┐
      PROGRAM ZOOM         │
      DIM  I(1Ø)           │  } Source Program
      :                    │
      :                    │
      ENDS$                ┘
::
:LIST,S,6,SOURC
:EDIT,SOURC,5              ┐
/I,2                       │
:                          │  } Edit List
:                          │
/E                         ┘
:JFILE,SOURC
:PROG,FTN,2,6,4,56,99
:PROG,LOADR
:RUN,ZOOM
123.62                     ┐
 :                         │  } Data for first run
 :                         │
ØØØØ1                      ┘
:RUN,ZOOM
321.5                      ┐
 :                         │  } Data for second run
 :                         │
Ø.56                       ┘
:JOB,NEXT JOB
```

APPENDIX C
SAMPLE DSGEN LISTINGS

This appendix contains the listings of two sample system generations. The first is for a minimum system: 2114 computer with 8K memory, and two teleprinters. The second is for a 2116 computer with 16K memory, card reader, lineprinter, magnetic tape, etc.

The values shown in these listings are *only* samples. The user must provide inputs appropriate to *his* system.

8K SYSTEM

```
SYS GEN CODE?
4321

SYS DISC CHNL?
10

# SECTORS/TRACK?
12

SYS DISC SIZE?
200

# DRIVES?
1

FIRST SYSTEM TRACK?
0
FIRST SYSTEM SECTOR?
3

SYS DISC SUBCHNL?
1

USER DISC SUBCHNL?
1

TIME BASE GEN CHNL?
0

IS 2114?
YES
```

```
LWA MEM?
17677

ALLOW :SS?
YES

PRGM INPT?
PT

LIBR INPT?
PT

PRAM INPT?
TY

*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT

NO UNDEF EXTS

ENTER PROG PARAMETERS

/E

#SYSTEM LINKS?
177

# USER LINKS?
5ØØ
```

```
SYSTEM

DISCM        Ø2ØØØ

DVRØØ        Ø4515

DVR31        Ø5275

* EQUIPMENT TABLE ENTRY

1Ø,DVR31,D,R,
14,DVRØØ,R
15,DVRØØ,R
/E

* DEVICE REFERENCE TABLE

 1 = EQT #?
2
 2 = EQT #?
1
 3 = EQT #?
1
 4 = EQT #?
3
 5 = EQT #?
3
 6 = EQT #?
3
 7 = EQT #?
Ø
 8 = EQT #?
/E

* INTERRUPT TABLE

11,1
14,2
15,3
/E

EXEC SUPERVISOR MODULES

$EXØ1        Ø615Ø
  $ADDR      Ø623I
```

```
$EXØ2          Ø615Ø
  $ADDR        Ø622Ø

$EXØ3          Ø615Ø

$EXØ4          Ø615Ø
  ASCII        Ø647l

$EXØ5          Ø615Ø
  $SRCH        Ø6317

$EXØ6          Ø615Ø
  $SRCH        Ø62Ø7
  $ADDR        Ø6534

$EXØ7          Ø615Ø
  $ADDR        Ø6327

$EXØ8          Ø615Ø
  ADDR         Ø63Ø5

$EXØ9          Ø615Ø
  ASCII        Ø642l

$EX1Ø          Ø615Ø

$EX11          Ø615Ø
  $SRCH        Ø6336

$EX12          Ø615Ø

$EX13          Ø615Ø
  ASCII        Ø65Ø7

$EX14          Ø615Ø
  ASCII        Ø6452

$EX15          Ø615Ø
  ASCII        Ø644O

$EX16          Ø615Ø

$EX17          Ø615Ø
  $LBL         Ø6544

$EX18          Ø615Ø

$EX19          Ø615Ø
  $LBL         Ø647Ø
```

```
$EX2Ø          Ø615Ø
  ASCII        Ø6453

I/O DRIVER MODULES

(NONE)

LWA SYS        Ø6663

FWA USER?
7ØØØ

USER SYSTEM PROGRAMS

JOBPR          Ø7ØØØ

LOADR          Ø7ØØØ

ASMB           Ø7ØØØ

ASMBD          14113

ASMB1          14113

ASMB2          14113

ASMB3          14113

ASMB4          14113

ASMB5          14113

FTN            Ø7ØØØ

FTNØ1          1Ø127
  SREAD        16241
  .OPSY        16774

FTNØ2          1Ø127

FTNØ3          1Ø127

FTNØ4          1Ø127
  %WRIT        14515
  FADSB        15213
  .OPSY        15356
  .FLUN        15413
  .PACK        15431
  DUMRX        15542
  DLDST        15626

*SYSTEM STORED ON DISC
```

16K SYSTEM

```
SYS GEN CODE?
1234

SYS DISC CHNL?
1Ø

# SECTORS/TRACK?
12

SYS DISC SIZE?
2ØØ

# DRIVES?
4

FIRST SYSTEM TRACK?
Ø
FIRST SYSTEM SECTOR?
3

SYS DISC SUBCHNL?
3

USER DISC SUBCHNL?
Ø

TIME BASE GEN CHNL?
2Ø

IS 2114?
NO

LWA MEM?
37677

ALLOW :SS?
YES

PRGM INPT?
PT

LIBR INPT?
MT

PRAM INPT?
TY

*EOT
*EOT
*EOT
```

```
NO UNDEF EXTS

ENTER PROG PARAMETERS

$EXØ1,Ø
$EXØ2,Ø
$EXØ3,Ø
$EXØ4,Ø
$EXØ5,Ø
$EXØ6,Ø
$EXØ7,Ø
$EXØ8,Ø
$EXØ9,Ø
$EX1Ø,Ø
$EX11,Ø
$EX12,Ø
$EX13,Ø
$EX14,Ø
$EX15,Ø
$EX16,Ø
$EX17,Ø
$EX18,Ø
$EX19,Ø
$EX2Ø,Ø
$ADDR,Ø
$LIBL,Ø
$SRCH,Ø
ASCII,Ø
DVRØØ,4
DVRØ2,Ø
DVR12,Ø
DVR15,Ø
/E

# SYSTEM LINKS?
177

# USER LINKS?
5ØØ

SYSTEM

DISCM          Ø2ØØØ
   *$CIC       Ø2ØØØ
   *$DYTY      Ø4355
   *$LDEX      Ø435Ø
   *EXEC       Ø2362
   *$DISC      Ø354Ø
   *$IDL1      Ø3717
   *$MUCT      Ø444Ø
   *$MDLD      Ø3167
   *$RQER      Ø3Ø1Ø
```

```
   *$JLOD    Ø4145
   *$MOVE    Ø3521
   *$TYPE    Ø3632
   *$SYIO    Ø4363
   *$BFND    Ø4467
   *$EFAD    Ø2657
   *$ABRT    Ø4211
   *$WAIT    Ø34Ø4
   *SETEQ    Ø35Ø5
   *$XCIC    Ø2Ø13
   *$CIC3    Ø2Ø57
   *SAVE$    Ø3771
   *$CLER    Ø4241
   *$OPER    Ø2361
   *ERRØ1    Ø3Ø26
   *ERRØ3    Ø3Ø32
   *ERRØ4    Ø3Ø34
   *ERRØ5    Ø3Ø36
   *ERRØ6    Ø3Ø4Ø
   *LUCHK    Ø2726
   *$DMA     Ø3354
   *$BLOP    Ø3341
   *$MBSY    Ø3347
   *$LDVR    Ø33Ø7
   *RQEQT    Ø2751
   *IO.LU    Ø3567
   *DRIVR    Ø3Ø53
   *ERRTN    Ø3ØØ4
   *IO.4Ø    Ø2671
   *$GDTK    Ø3274
   *MPY17    Ø4462
   *DISCX    Ø3264
   *..RRL    Ø4443
   *DEFØ4    Ø2615
   *DEF13    Ø4474
   *DEF19    Ø45Ø2
   *DEF2Ø    Ø45Ø3

$EXØ1        Ø4515
   *$EXØ1    Ø4515

$EXØ2        Ø4576
   *$EXØ2    04576

$EXØ3        Ø4646
   *$EXØ3    Ø4646

$EXØ4        Ø47Ø3
   *$EXØ4    Ø47Ø3

$EXØ5        Ø5224
   *$EXØ5    Ø5224
```

```
$EXØ6          Ø5373
   *$EXØ6      Ø5373

$EXØ7          Ø5432
   *$EXØ7      Ø5432

$EXØ8          Ø5611
   *$EXØ8      Ø5611

$EXØ9          Ø5746
   *$EXØ9      Ø5746

$EX1Ø          Ø6217
   *$EX1Ø      Ø6217

$EX11          Ø6363
   *$EX11      Ø6363

$EX12          Ø6551
   *$EX12      Ø6551

$EX13          Ø674Ø
   *$EX13      Ø674Ø

$EX14          Ø7277
   *$EX14      Ø7277

$EX15          Ø76Ø1
   *$EX15      Ø76Ø1

$EX16          1ØØ73
   *$EX16      1ØØ73

$EX17          1Ø21Ø
   *$EX17      1Ø21Ø

$EX18          1Ø6Ø4
   *$EX18      1Ø6Ø4

$EX19          11267
   *$EX19      11267

$EX2Ø          116Ø7
   *$EX2Ø      116Ø7

ASCII          12112
   *CNDEC      12112
   *CNOCT      12116

$ADDR          122Ø4
   *$ADDR      122Ø4
```

```
$SRCH          12221
   *$SRCH      12221
   *$CMPR      12461

$LBL           12546
   *LBLIO      12566
   *ISLBL      126Ø2
   *LBLMV      1256Ø
   *CHSUM      12546
   *MESSG      12616

DVRØ2          12636
   *I.Ø2       12636
   *C.Ø2       12715

DVRØ5          13Ø4Ø
   *I.Ø5       13Ø4Ø
   *C.Ø5       13113

DVR12          133Ø7
   *I.12       133Ø7
   *C.12       13514

DVR15          14Ø37
   *I.15       14Ø37
   *C.15       14175

DVR31          14741
   *I.31       14741
   *C.31       15Ø24

* EQUIPMENT TABLE ENTRY

1Ø,DVR31,D,R
12,DVR22,D
14,DVRØ5,R
15,DVRØ1
16,DVRØ2,R
17,DVR12,R
21,DVR15,R,D
22,DVRØØ
/E

* DEVICE REFERENCE TABLE

 1 = EQT #?
3
 2 = EQT #?
1
 3 = EQT #?
```

```
1
 4 = EQT #?
5
 5 = EQT #?
7
 6 = EQT #?
6
 7 = EQT #?
4
 8 = EQT #?
2
 9 = EQT #?

8
1Ø = EQT #?
Ø
11 = EQT #?
/E

* INTERRUPT TABLE

11,1
13,2
14,3
15,4
16,5
17,6
21,7
22,8
/E

EXEC SUPERVISOR MODULES

(NONE)

I/O DRIVER MODULES

DVRØØ         157Ø1
   *I.ØØ      157Ø1
   *C.ØØ      16Ø31

DVRØ1         157Ø1
   *I.Ø1      157Ø1
   *C.Ø1      15753

DVR22         157Ø1
   *I.22      157Ø1
   *C.22      16441

LWA SYS    16535
```

```
FWA USER?
17ØØØ

USER SYSTEM PROGRAMS

JOBPR          17ØØØ
   *JOBPR      17ØØØ

LOADR          17ØØØ

ASMB           17ØØØ
   *ASMB       23515
   *?ASCN      2Ø677
   *?ASMB      17554
   *?BNCN      215Ø7
   *?BPKU      22325
   *?CHOP      17646
   *?CHPI      226Ø7
   *?DCOD      22615
   *?ENDS      22227
   *?ERPR      22147
   *?MSYS      22666
   *?GETC      22653
   *?MOVE      2Ø437
   *?MSYM      21774
   *?RLUN      23374
   *?AFLG      23423
   *?LSTL      21716
   *?LUNI      23431
   *?RFLG      2342Ø
   *?Z         23441
   *?ASM1      2Ø371
   *?LABE      2Ø4Ø7
   *?OKOL      223Ø6
   *?ORRP      216Ø2
   *?PNLE      23436
   *?SETM      22673
   *?SUP       223Ø2
   *?LPER      223Ø5
   *?PERL      2227Ø
   *?LOUT      22335
   *?LTFL      22274
   *?DRFL      23426
   *?LTSA      22557
   *?LTSB      2256Ø
   *?ORGS      223ØØ
   *?CNTR      22367
   *?TSTR      23427
   *?ASII      23445
   *?ICSA      22145
   *?FLGS      23415
   *?BFLG      23416
   *?LFLG      23417
```

```
   *?TFLG    23421
   *?X       2344Ø
   *?MESX    175Ø5
   *?ASCI    23444
   *?LINC    221Ø7
   *?LINS    21764
   *?LIST    21652
   *?LUNP    23433
   *?OPLK    176ØØ
   *?OPER    22637
   *?PKUP    2232Ø
   *?PLIT    224Ø5
   *?PNCH    2Ø631
   *?PRNT    22Ø32
   *?RSTA    2Ø1Ø5
   *?LWA     23437
   *?RDSC    234ØØ
   *?WEOF    23Ø2Ø
   *?WRIF    231Ø1
   *?LGFL    23425
   *?SEGM    17541
   *?SYMK    2Ø5Ø6
   *?V       22632
   *?ARTL    22471
   *?LST     22273
   *?PLIN    2343Ø
   *?PCOM    22111
   *?SECT    23414
   *?NEAU    17443
   *?HA38    22346
   *?XRFI    1754Ø

ASMBD        24113
   *ASMBD    24435

ASMB1        24113
   *ASMB1    24361
   *?LITI    25Ø23
   *?CMQ     24553
   *?INSR    24721
   *?HA3Z    24522
   *?ENP     24655
   *?EXP     2464Ø

ASMB2        24113
   *ASMB2    24344
   *?ART     25Ø13
   *?BREC    2447Ø
   *?LKLI    25527
   *?SKPR    24434
   *?SPCR    24437
```

```
ASMB3          24113
   *ASMB3      24623

ASMB4          24113
   *ASMB4      24361
   *?INS?      24534

ASMB5          24113
   *ASMB5      24344

FTN            17000
   *%WLIC      20042
   *%FTN0      17000
   *%WPRN      17735
   *%ERRR      17701
   *%RDIS      17557
   *%WDIS      17244
   *%SEGN      17224
   *%WTRA      17236
   *%WSEC      17237
   *%RTRA      17347
   *%RSEC      17350
   *%RBFA      17352
   *%LUNO      17203
   *%LUNI      17204
   *%LUNP      17205
   *%NAMA      17226
   *%RTYP      17227
   *%WLIN      20017
   *%WPAG      20043
   *%TILT      20074
   *%RDSI      20044
   *%WDSI      20056
   *%WOUT      17334
   *%RBFW      17623
   *%LABL      17733
   *%CONA      17734
   *%ENDP      20105
   *%WDLU      17241
   *%RDLU      17346
   *%RFLG      17554
   *%WBFW      17341
   *%WBFA      17232
   *%HEDN      20010
   *%DUP8      17373
   *%NXDV      17402
   *%NELM      17352
   *%STYP      17433
   *%LGO       17202
```

```
FTNØ1          2Ø127
   *%FTN1      23550
  SREAD        26241
   *%READ      26241
   *%JFIL      267Ø7
   *%RDSC      26663
  .OPSY        26774
   *.OPSY      26774

FTNØ2          2Ø127
   *%FTN2      2Ø741

FTNØ3          2Ø127
   *%FTN3      22117

FTNØ4          2Ø127
   *%FTN4      2Ø7Ø2
  %WRIT        24515
   *%WRIT      247ØØ
   *%WRIF      24577
   *%WBUF      24777
  FADSB        25213
   *.FAD       25213
   *.FSB       2522Ø
  .OPSY        25356
   *.OPSY      25356
  .FLUN        25413
   *.FLUN      25413
  .PACK        25431
   *.PACK      25431
  DUMRX        25542
   *$LIBR      25542
   *$LIBX      2557Ø
  DLDST        25626
   *.DLD       25626
   *.DST       25636

ALGOL          17ØØØ
   *HPAL       27517
   *%HPST      27733
  MPY          31275
   *.MPY       31275
  %WRIT        3142Ø
   *%WRIT      316Ø3
   *%WRIF      315Ø2
   *%WBUF      317Ø2
  SREAD        32116
   *%READ      32116
   *%JFIL      32564
   *%RDSC      3254Ø
  .OPSY        32651
   *.OPSY      32651
```

```
ALGL1          327Ø6
   *ALGL1      3335Ø
   *%LNAL      327Ø7
   *%ABAL      327Ø6

*SYSTEM STORED ON DISC
```

APPENDIX D
RELATION TO OTHER SOFTWARE

The Hewlett-Packard 2116 and 2114 are general-purpose computers; as such, they can handle other HP software when the Moving-Head Disc Operating System is inactive. Every computer is shipped with the software and documentation appropriate to the system configuration.

Prepare Tape System can be used to store the relocatable modules of DOS-M on a magnetic tape. DSGEN can then read from this magnetic tape to generate a system.

In an attempt to make DOS-M compatible with the Real-Time Executive, DOS-M simulates the Real-Time EXEC requests as follows (See *REAL-TIME SOFTWARE*, 02116-9139):

READ/WRITE	Identical for work area of disc and I/O devices.
I/O CONTROL	Identical
I/O STATUS	Status word 2 returns transmission log instead of Real-Time Equipment Table word 5.
DISC ALLOCATION	Simulates request in work area.
DISC RELEASE	No action; tracks cannot be released.
PROGRAM COMPLETION	Identical
PROGRAM SUSPENSION	Identical
PROGRAM SEGMENT LOAD	Identical
PROGRAM SCHEDULE	Treated as segment load.
CURRENT TIME	Word 5 set to Ø, other words identical.
EXECUTION TIME (TIMER)	Not accepted See N option of RUN request.

NOTE: The RTE System runs on a 2116 only and uses a fixed-head disc or drum memory with a sector size of 64 words.

APPENDIX E
LINE PRINTER FORMATTING

When a user program makes a READ/WRITE EXEC call to the line printer (HP 2778A or HP 2778A-01), the line printer driver DVR12 interprets the first character in the line as a carriage control character and prints it as a space. The control characters have the following meanings:

Character	Meaning
blank	Single space (print on every line),
Ø	Double space (print on every other line),
1	Eject page,
*	Suppress space (overprint next line),
others	Single space.

Each printed line is followed by an automatic single space unless suppressed by the asterisk (*). Double spacing requires an additional single space prior to printing the next line. If the last line of a page is printed and the following line contains a "1", then a completely blank page occurs.

When a user program makes an I/O CONTROL EXEC call and the function code equals 11_8 (see Section IV, I/O CONTROL EXEC CALL), then the optional parameter word defines a format action to be taken by the line printer. The parameter word has these meanings:

Parameter Word (Dec)	Meaning
< Ø	Page eject,
Ø to 55	Space Ø to 55 lines ignoring page boundaries,
56 to 63	Use carriage control channel equal to the word - 55,
64	Set automatic page eject mode,
65	Clear automatic page eject mode.

*DVR12 checks for certain program names (FTN, ASMB, ALGOL LOADR, JOBPR); for these programs, it prints the first character of each line and generates a single space.

If the parameter word equals zero, the automatic single space is to be suppressed on the next print operation only.

CARRIAGE CONTROL CHANNELS

If the parameter word is between 55 and 64, then the printer spaces using the standard carriage control channels, which have the following meanings:

Channel 1	Single space with automatic page eject.
Channel 2	Skip to next even line with automatic page eject.
Channel 3	Skip to next triple line with automatic page eject.
Channel 4	Skip to next 1/2 page boundary.
Channel 5	Skip to next 1/4 page boundary.
Channel 6	Skip to next 1/6 page boundary.
Channel 7	Skip to bottom of the page.
Channel 8	Skip to top of next page.

AUTOMATIC PAGE EJECT

During non-automatic page eject mode, if the parameter word is equal to 56, then it is interpreted as equal to 1. Automatic page eject mode applies only to single space operations.

APPENDIX F
SUMMARY OF DIRECTIVES

DIRECTIVE	DESCRIPTION
:ABORT	Terminate the current job.
:ADUMP[,*FWA*[,*LWA*][,B],L]	Dump a program if it aborts
:BATCH, *logical unit*	Switch from keyboard to batch mode, or reassign batch device.
:COMMENT *string*	Print a message.
:DATE,*day*[,*hour*,*min*]	Set the date and the time (if time-base is present).
:DD	Dump on entire disc onto a disc on another subchannel.
:DD,X	Dump the system area only to another disc.
:DD,U[,*file*[,(*name*)],*file*[,(*name*)]...]	Dump all or specified files of the user disc to another disc, optionally assigning new file names.
:DN,*n*	Declare an I/O device down.
:DUMP,*log.unit*,*file*[,S_1[,S_2]]	Dump all or part of a user file to a peripheral I/O device.
:EDIT,*file log.unit*],*new*]	Edit a source statement file stored on disc, optionally creating a new file.
:EJOB	Terminate the current batch and/or job normally.
:EQ[,*n*]	List the equipment table.
:GO[,P_1,P_2...P_5]	Continue processing a suspended program.
:IN,*label*	Label or unlabel ("*") the current user disc.

DIRECTIVE	DESCRIPTION
:JFILE,*file*	Specify a source file on the disc for the assembler or compiler.
:JOB[,*name*]	Initiate a user job.
:LIST,S,*log.unit*,*file*[,*m*[,*n*]]	List all or part of a source statement file.
:LIST,U,*log.unit*[,$file_1$,...]	List the user directory.
:LIST,X,*log.unit*[,$file_1$,...]	List the system directory.
:LU[,n_1[,n_2]]	Assign or list logical units.
:OFF	Abort the currently executing program or operation without terminating the job.
:PAUSE	Suspend the current job.
:PDUMP[,*FWA*[,*LWA*]][,B][,L]	Dump a program after normal completion.
:PROG,*name*[,P_1,P_2...P_5]	Turn on a system or user program.
:PURGE[,$file_1$,$file_2$,...]	Delete user files.
:RUN,*name*[,*time*][,N]	Run a user program
:SA,*track*,*sector*[,*number*]	Dump disc In ASCII to standard list device.
:SO,*track*,*sector*[,*number*]	Dump disc in octal to standard list device.
:SS	Set up system search for file names over all subchannels.
:SS,n_1,n_2...	Set up system search for file names over specified subchannels.
:SS,99	Restrict search for file names to current user disc (plus system directory for RUN & PROG).

DIRECTIVE	DESCRIPTION
:STORE,A,*file,sectors*	Reserve space for an ASCII data file.
:STORE,B,*file,sectors*	Reserve space for a binary data file
:STORE,P[,$name_1$, $name_2$,...]	Store temporary Loader generated programs as permanent files.
:STORE,R,*file* [,*log.unit*]	Store a relocatable file from a peripheral I/O device or from the JBIN area of disc after an assembly or compilation.
:STORE,S,*file,log.unit*	Store a source statement file from a peripheral I/O device.
:TRACKS	Print the disc track status of the current user disc.
:TYPE	Return to keyboard mode from batch mode.
:UD[,[*label*][,*n*]]	Change the subchannel assignment for the user disc, or request label & subchannel information for a user disc.
:UP,*n*	Declare an I/O device up.

APPENDIX G
SUMMARY OF EXEC CALLS

Consult Section III for the complete details on each EXEC call.

For each EXEC call, this appendix includes only the parameters (P_1 through P_n) of the assembly language calling sequence.

READ/WRITE: Transfer input or output.

```
RCODE   DEC   1 or 2        1 = read or 2 = write
CONWD   OCT   c            (See Section III for control information.)
BUFFR   BSS   n            (n-word buffer)
BUFFL   DEC   n or -2n     (buffer length, words (+), characters (-).)
DTRAK   DEC   p            (disc track; optional)
DSECT   DEC   q            (disc sector; optional)
```

I/O CONTROL: Carry out control operations.

```
RCODE   DEC   3
CONWD   OCT   c            (See Section III for control information.)
PARAM   DEC   n            (Optional parameter required by some CONWDs.)
```

PROGRAM COMPLETION: Signal end of program.

```
RCODE   DEC   6
```

PROGRAM SUSPEND: Suspend calling program.

```
RCODE   DEC   7
```

PROGRAM SEGMENT LOAD: Load segment of calling program.

```
RCODE   DEC   8
SNAME   ASC   3,xxxxx      (xxxxx is segment name)
```

TIME REQUEST: Request the 24-hour time and day.

```
RCODE   DEC   11
ARRAY   BSS   5            (Time values; tens of milliseconds, seconds,
                            minutes, hours, returned in that order.)
```

I/O STATUS: Request device status.

```
RCODE   DEC   13
CONWD   DEC   n            (Logical unit number)
STATS   NOP                (Status returned here)
TLOG    NOP                (Transmission log returned here)
```

FILE READ/WRITE: Read or write a user data file.

```
RCODE   DEC   14 or 15     (14 = read or 15 = write.)
CONWD   OCT   c            (See Section III for control information.)
BUFFR   BSS   n            (Buffer of n words.)
BUFFL   DEC   n or -2n     (Length of buffer in words (+) or
                            characters (-).)
FNAME   ASC   3,xxxxx      (User file name = xxxxx.)
RSECT   DEC   m            (Relative sector within file.)
```

WORK AREA STATUS: Ascertain if *n* contiguous work tracks are available.

```
RCODE   DEC   16
NTRAK   DEC   n            (Number of consecutive tracks desired.)
TRACK   NOP                (Desired first track; from LIMITS call.)
STRAK   NOP                (Actual starting track, or Ø if n not
                            available.)
```

WORK AREA LIMITS: Ascertain first and last tracks of work area.

```
RCODE   DEC   17
FTRAK   NOP       (Returns first work track number here.)
LTRAK   NOP       (Returns last work track number here.)
SIZE    NOP       (Returns number of sectors per track here.)
```

SEARCH FILE NAMES: Ascertain if a file name exists in the directory.

```
RCODE   DEC   18
FNAME   ASC   3,xxxxx   (xxxxx is the file name.)
NSECT   NOP             (Number of sectors in file returned here, or
                         Ø if not found.)
```

CHANGE USER DISC: Change the current user disc subchannel.

```
RCODE   DEC   23
LABEL   ASC   3,xxxxx   (Disc label = xxxxx or ASCII 1, * for unlabel.)
SUBCH   DEC   (Ø to 7)  (Subchannel number; optional parameter.)
```

APPENDIX H
MESSAGES

During the operation of DOS-M certain messages may be printed on the system teleprinter. These messages may be error reports or simply informative; they are generated by various parts of DOS-M. Appendix H lists these messages alphabetically including where they originated, what they mean, and what response, if any, the operator must make. Messages that begin with a variable name or a non-alphabetic character are listed by the first non-variable, alphabetic character.

Message	Source	Description
BAD CONTROL STATE.	JOBPR	Directive just entered is not acceptable in DOS-M. Enter correct directive on system teleprinter.
BEGIN 'DEBUG' OPERATION	DEBUG	Any legal DEBUG operations may now be entered. Enter any legal DEBUG operations.
CHECKSUM ERROR	JOBPR	Checksum error in INPUT to :ST,R,file, LU directive. Correct tape.
CW *nnnnn*		In a READ/WRITE EXEC call at *nnnnn*, buffer is out of memory bounds. Correct program.
DEVICE #*nn* DOWN	JOBPR	EQT # *nn* is unavailable (down). Use the UP,*nn* directive to make the device available (UP). (Then use the GO directive if needed.)
DICTIONARY OVERFLOW	JOBPR	No room is left for entries in the user file dictionary. Put file on another disc or remove some of the files.
??? DISC	DISCM	Informs user that user disc was labeled by a non-DOS-M system. May be made a DOS-M disc by labeling or unlabeling with :IN.
DISC GEN CODE *nnnn* NOT SYS GEN CODE *nnnn* ERR POSS	DISCM	Informs the user that the disc being requested was initialized (labeled) by a system with a different system Generation Code. Generation code on disc may be updated by labeling or unlabeling using IN.

Message	Source	Description
DISC NOT ON SYSTEM	DISCM	No disc pack with the currently requested label can be found on the system. Mount disc pack with correct label or ready drive containing disc.
DONE?	JOBPR	Thirty feed frames (paper tape) or an end-of-file (magnetic tape) have occurred during input. Enter YES for end of input; NO for more input.
??? LABEL *xxxxxx* DOS LABEL *xxxxxx* TSB LABEL *xxxxxx*	DISCM	Attempting to label (or unlabel) an already labeled disc pack. Enter YES to relabel the disc pack or NO to drop the request to relabel the disc pack.
DUPLICATE FILE NAME	JOBPR	Double defined file name found in a :STORE directive (other than STORE,P), or an EDIT directive with a new file name; or on DD,U. Remove file or rename file.
$END ALGOL	ALGOL	End of ALGOL compilation. No response required.
$END ASMB	ASMB	Assembly as completed. No response required.
$END ASMB CS	ASMB	Assembly has ended because of an error in the Assembler Control statement. Correct the control Statement.
$END ASMB NPRG	ASMB	Assembly has terminated because no JFILE was found when required. Define the file using a JFILE directive.
$END ASMB PASS	ASMB	Another pass of the source program through the input device is required. Printed on the system after Pass 1. Replace the program in the input device and type: :GO.
$END ASMB XEND	ASMB	Assembly stops. An EOF occurred in the source program before an END statement. Add an END statement to the program.
END FILE	JOBPR	During an EDIT, (1) the master file ended before completion of editing or (2) a colon occurred in column 1 of a source statement. Check input to the EDIT program.

Message	Source	Description
$END FTN	FTN	Compilation has completed. No response required.
END JOB *xxxx* [RUN = *xxxx* MIN. *xx.x* SEC EXEC = *xxxx* MIN. *xx.x* SEC]	JOBPR	End of current job. Total job time and execution time of the job are printed on the system teleprinter and standard list device if a time-base generator is present. Enter next job.
ENTER FILE NAME(S) OR /E	LOADR	Enter list of relocatable program files. To terminate list of file names type "/E".
ENTRY ERROR	DEBUG	DEBUG operation entered was illegal. Correct entry.
EQT *xx* CH *xx* DVR*xx* D R U*x* S*x*	JOB	Equipment table entry printed by the directive :EQ. No action required
EXTRA PARAMETERS	JOBPR	More than 15 parameters in a directive. Reduce the number of parameters.
FI *nnnnn*	DISCM	In a FILE READ/WRITE EXEC call, the file requested at *nnnnn* cannot be found. Calling program is aborted. Check for File name requested at *nnnnn*. If the File *nnnnn* is not present, enter the File *nnnnn*.
HPAL	ALGOL	Control statement error. Correct control statement.
IB *nnnnn*	DISCM	Illegal buffer address in EXEC call at location *nnnnn*. Program is aborted. Correct buffer program address.
IE *nnnnn*	DISCM	If a colon occurs in the first column of input entered through the batch device during a program execution, the program is aborted, and control is given to the JOBPR. *nnnnn* is the memory location of the input request.
IGNORED	DISCM	Input from system teleprinter during program execution cannot be processed. Correct input.

Message	Source	Description
*IGNORED	JOBPR	All directives following EJOB and before next JOB except BATCH, TYPE, TRACKS, and OFF are ignored. Enter acceptable directive.
file ILLEGAL	JOBPR	On a source file LIST directive, the requested file was not a source file. Retype LIST directive using source file. A file name begins with a non-alphabetic character. Rename the file.
ILLEGAL DIGIT	JOBPR	In a decimal number, character is other than Ø-9. Enter correct decimal number. In an octal number, digit is other than Ø-7. Enter correct octal number.
ILLEGAL LUN	JOBPR	Logical unit requested is equal to zero, greater than the number of logical units in the system, not the correct type (i.e., input type for output device), etc. Enter a correct logical unit.
ILLEGAL PROGRAM RUN LIMITS	DISCM	Attempt to run a user main or segment whose user area limits or base page limits will not fit within the limits of the current system. Recreate user mains or segments on current system using LOADR.
ILLEGAL PROGRAM TYPE	JOBPR	Program requested in a RUN or PROG is not legal. Enter correct name.
INPUT ERROR	DISCM	Equipment table entry number of logical unit number in EQ, LU, UP or DN is illegal. Enter correct equipment table entry number.
INPUT:DATE, XXXXXXXXXX[,H,M,]	DISCM	When system is initiated from the disc, DOS-M requires a DATE directive. The "HM" is ignored in DOS-M if a Time Base Generator is not in the system. Enter a DATE directive.
I/O ERR ET EQT #*mm*	DISCM	End-of-tape on device #*mm*. EQT #*mm* is unavailable. To make the device available (up) use the UP,*n* directive.
I/O ERR NR EQT #*mm*	DISCM	The device #*mm* is not ready. To make the device available (up), use the UP,*n* directive.

Message	Source	Description
I/O ERR PE EQT *#mm*	DISCM	Parity error on device *#mm* returns to program return address with A set to status, B set to Ø. Call maintenance.
IT *nnnnn*	DISCM	Illegal disc track or sector address in EXEC call from location *nnnnn*. Program is aborted. Correct the track or sector address in EXEC call.
I/O ERR{PE / NR}USER DISC	DISCM	A parity error or device not ready occured when attempting to assign a user disc. Disc may not be formatted; format it with DSGEN.
JBIN OVF	FTN, ASMB	Overflow of job binary area during assembly or compilation. Reduce size of job or purge user files.
JOB ABORTED!	JOBPR	Correct problem and start new job.
JOB *xxxxx ddddddddd* [TIME = *xxxx* MIN. *xx.x* SECS EXEC = *xxxx* MIN. *xx.x* SEC.]	JOBPR	Message printed at the beginning of each job. The time information is deleted in DOS-M if a Time Base Generator is not included in the system. Start job.
LØ1	LOADER	Checksum error on tape.
LØ2		Illegal record.
LØ3		Memory overflow.
LØ4		Base page overflow.
LØ5		Symbol table overflow.
LØ6		Duplicate main or segment name (may be caused by attempting to run the loader twice in one job).
LØ7		Duplicate entry point.
LØ8		No main or segment transfer address.
LØ9		Record out of sequence.
L1Ø		Insufficient directory work area, or user area space.

Message	Source	Description
L11		Program name table overflow.
L12		User file specified cannot be found.
L13		Program name duplication.
L14		Non-zero base page length.
L15		Segment occurred before main.
L16		Program overlay (illegal ORG).
LBL = *llllll*	DISCM	Disc subchannel referenced is labeled *llllll*. If attempting to change user disc subchannel, enter UD with correct label.
LIMIT ERROR	JOBPR	In a directive, source statement numbers are out of order (EDIT), dump limits are incompatible (PDUMP, ADUMP), sector numbers are illegal (DUMP), or beginning source statement number is greater than final statement number (EDIT). Correct directive and re-enter.
xxxx LINES	JOBPR	Total number of statements stored by a STORE,S directive. No response required.
****LIST END****	JOBPR	Terminates list of source statements generated by a LIST directive. No response required.
LN *nnnnn*	DISCM	Logical unit requested by an EXEC call at *nnnnn* is unassigned. Program is aborted. Reassign logical unit.
LOADR COMPLETED	LOADR	Loading has completed. No responses required.
LOADR SUSP	LOADR	Loader has suspended (usually at EOT). Type :GO,*n* to restart the Loader with proper parameter value.
LOADR TERMINATED	LOADR	Loader has terminated because of an error. Check input.
LOAD TAPE	LOADR	In conjunction with LOADR SUSP, this message requests that next relocatable tape be loaded before GO. Load the next relocatable Tape and enter :GO to read next tape or :GO,1 to indicate that all tapes are read in.

Message	Source	Description
LU *nnnnn*	DISCM	Illegal logical unit in EXEC call at *nnnnn*. Program is aborted. Enter correct logical unit number.
LU*xx* EQT*xx*	JOBPR	Logical unit table entry; EQT #*xx* assigned to LU#*xx*. No response required.
LUN UNASSIGNED	JOBPR	Logical unit requested in a directive is unassigned. Assign logical unit number requested in the directive.
xxxxx MISSING	DISCM	Segment *xxxxx* requested by an EXEC call in system or user directory. Job is aborted. Correct job.
MISSING PARAMETER	JOBPR	A parameter is missing in a directive. Retype the directive correctly.
MP *nnnnn*	DISCM	Memory protect violation at location *nnnnn*. Program is aborted. Correct the program.
NAME *IGNORED	JOBPR	Illegal JOB *name;* non-alphabetic first character. Retype correct Job name.
NEXT AVAIL TRACK=*tt* BAD=*n*	JOBPR	In TRACK directive, *tt* = first track beyond end of current user area; *n* = number of bad tracks. "BAD=*n*" returned only if bad tracks do exist. *tt* = "NONE" if no tracks are available
NO BIN END	JOBPR	No END record detected when storing a relocatable binary program.
NO PROGRAM LOADED	LOADR	No programs were loaded by the LOADR. Loading terminates.
NO SOURCE	JOBPR	No source statements following a /R or /I in an EDIT directive. Job is aborted. Enter source statements after the /R or /I.
NO SOURCE	ALGOL	Source file from disc not pre-set.
NUMBER OVERFLOW	JOBPR	An integer is too large.

Message	Source	Description
OR *nnnnn*	DISCM	I/O operation requested by EXEC call at *nnnnn* is rejected. Program is aborted. Check program
OVERFLO JBIN	JOBPR	There is not enough room in the user area for storing the relocatable binary from the user area.
PARAMETER ILLEGAL	JOBPR	A parameter of a directive is illegal. Re-enter directive.
PARITY ERROR/SC=*n*,TRK=*ttt*,SCTR=*sss*	JOBPR	Parity error during disc read or write. Call maintenance.
PAUSE *xxxx*	DISCM	Program has temporarily suspended itself. *xxxx* is an octal number. Restart program using the GO directive
RE-ENTER STATEMENT ON TTY	JOBPR	Follows most error messages that do not cause abort. Type in the correct statement.
RQ *nnnnn*	DISCM	Illegal request code in EXEC call at *nnnnn*. Program is aborted. Correct the program.
SPARE TRK OVERFLOW	JOBPR	Defective cylinder detected and no spare tracks available for reassignment.
STOP *xxxxx nnnnn*	LIBR	Program *xxxxx* has terminated at location *nnnnn*.
SUBCHAN = *n*	DISCM/ JOBPR	Given in response to :UD information request or when :SS makes new subchannel assignment. No response required.
xxxxx SUSP	DISCM	Program *xxxxx* suspended by EXEC call or PAUSE directive. Restart program using the GO directive.
TAPE END	JOBPR	EOT flag set on magnetic tape or paper tape device during output via JOBPR directives :DUMP and :LIST or output of a JOB or EJOB statement. If a magnetic tape, it is rewound with standby, if paper tape a trailer is punched. The JOBPR will then pause to allow new tape to be set up. Mount a new magnetic tape. Enter :GO to continue the output.

Message	Source	Description
TM *nnnnn*	DISCM	Maximum execution time exceeded. The program is currently at *nnnnn* and is aborted. Increase execution time.
#TRACKS UNAVAILABLE	DISCM	There are not enough work tracks for the compiler. Purge disc of unecessary files.
TRAC # TOO BIG	JOBPR	Track requested is higher than last available disc track (track may be in JBIN area). Redefine the track request or purge files or use different disc.
TSB DISC	DISCM	Informs user that the user disc was labeled by a non-DOS-M system. May be made DOS-M disc by labeling or unlabeling with :IN.
UD *nnnnn*	DISCM	Unable to find user disc requested by EXEC call at *nnnnn*. Mount required disc and type :GO; or terminate program with :AB or :OF.
UNLBL	DISCM	User disc specified in UD is unlabeled. If trying to change user disc assignment, enter UD,*[,*n*].
file name UNDEFINED	JOBPR	Undefined file name in PURGE, LIST, RUN, STORE or DD,U,*file*. Retype correct file name on the system teleprinter.
UNDEFINED EXTS	LOADR	Undefined external references exist in programs loaded. The external references are listed one per line. To load additional programs from paper tape type :GO,Ø[,*n*].
WRONG INPUT	JOBPR	Relocatable binary input furnished for a source file request or vice-versa. Put in a correct input.
nn xx	ERRØ	Library routine error code.
@	JOBPR/ DISCM	Directives may be entered. Enter desired directive.
*	DISCM	Operator attention directives may be entered. Enter desired directive.

APPENDIX I
MAGNETIC TAPE USAGE

Input/output transfers to and from a HP 3Ø3Ø magnetic tape unit can be programmed using the standard READ/WRITE EXEC call. (See Section III.) When specifying the data buffer length, the programmer must know that a buffer length of zero (Ø) causes the driver to take no action on a write or an ASCII read. Only the amount of data that fits within the buffer is transmitted to the user on read. A zero (Ø) buffer length on binary read causes a forward skip one record.

In the I/O STATUS EXEC call, bits 7-Ø of the second status word contain the status of the magnetic tape unit. The bits have the following meaning when they are set (i.e., equal to one):

BIT	MEANING
7	End-of-file record encountered while reading, forward spacing, or backward spacing.
6	Start-of-tape marker sensed.
5	End-of-tape marker sensed.
4	Timing error on last read/write operation.
3	I/O request rejected by magnetic tape unit.
2	No write enable ring, or the tape unit is rewinding.
1	Parity error on last read/write operation.
Ø	Tape unit busy, or in local mode.

The status bits are stored in the EQT entry; they are updated everytime the driver is called. A dynamic status request is processed as soon as the magnetic tape EQT entry is available (availability bits equal to ØØ), and returns the actual status of the device (obtained from the driver) to the calling program in the A-register and to the EQT entry.

Buffers of less than six words are padded to six words. The maximum buffer length is 16,384.

ERROR RECOVERY PROCEDURES

On a read parity error, the driver rereads the record three times before setting the parity error status bit and returning to the calling program. The final read attempt is transmitted to the program buffer.

On a write parity error, the driver continues to retry the write until one of these two conditions occurs:

a) The record is successfully written, or

b) The end-of-tape is encountered.

On a write without the write enable ring, the magnetic tape unit is made unavailable (magnetic tape not ready). DOS-M prints a message:

I/O ERR NR EQT#*n*

and waits for the operator to correct the unit and enter :GO.

At the end-of-tape there are only two legal forward motion requests:

a) Write end-of-file, or

b) Read record.

All other forward motion requests (write, forward space) cause the unit to be made unavailable. In addition, only one of the legal motion requests may be made after an end-of-tape. Backward motion requests clear the end-of-tape status.

APPENDIX J
DISC LABELS

Sector 0 of track 0 of each disc is used for label information. In addition, if the user area is on the system disc, a label also occurs in Sector 0 of the first track after the system area.

The contents of the label include:

Word 0: Label presence code (ASCII "LB").

Word 1: System Proprietary Code:

1. "DO" for DOS-M
2. "TS" for Time-Shared Basic

Word 2: System generation code assigned at system generation time. The code can be any 4 decimal digits.

Words 3-5: A six-character disc label. If the first character equals * the disc is unlabeled. This label can only be set using :IN (for user areas) or by DSGEN (set to "SYSTEM" for system discs).

Word 31: Checksum of words 0-30.

The first 64 words are reserved for label information. Word 65 contains the next available track and sector. Words 66 and 67 contain the number of bad tracks and the next available space track.

INDEX

CUT ALONG LINE

HEWLETT hp PACKARD

READER COMMENT SHEET

MOVING-HEAD
DISC OPERATING SYSTEM

02116-91779 AUGUST, 1970

Hewlett-Packard welcomes your evaluation of this text. Any errors, suggested additions, deletions, or general comments may be made below. Use extra pages if you like.

FROM PAGE__OF__

NAME: _____

ADDRESS: _____

_____

NO POSTAGE NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AS SHOWN ON OTHER SIDE AND TAPE

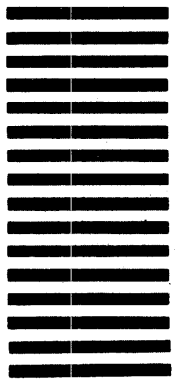
FOLD FOLD

FIRST CLASS
PERMIT NO. 141
CUPERTINO
CALIFORNIA

BUSINESS REPLY MAIL

No Postage Necessary if Mailed in the United States Postage will be paid by

MANAGER, SOFTWARE PUBLICATIONS
HEWLETT - PACKARD
CUPERTINO DIVISION
11000 Wolfe Road
Cupertino, California
95014



FOLD FOLD