HEWLETT PACKARD

HP

24307B

DOS-III
Disc Operating System
reference manual

HP 2000 COMPUTER SYSTEMS

HP 24307B
DOS-III
Disc Operating System
reference manual

HEWLETT PACKARD

HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, 95014

MANUAL PART NO. 24307-90006
MICROFICHE PART NO. 24307-90007

Printed: FEB 1975
Printed in U.S.A.

List of Effective Pages

Pages	Effective Date
Title	Feb 1975
iii to xiii	Feb 1975
1-1 to 1-17	Feb 1975
2-1 to 2-60	Feb 1975
3-1 to 3-46	Feb 1975
4-1 to 4-4	Feb 1975
5-1 to 5-35	Feb 1975
6-1 to 6-3	Feb 1975
7-1 to 7-4	Feb 1975
8-1 to 8-25	Feb 1975
9-1 to 9-22	Feb 1975
10-1 to 10-46	Feb 1975
11-1 to 11-8	Feb 1975
12-1 to 12-10	Feb 1975
13-1 to 13-34	Feb 1975
14-1 to 14-3	Feb 1975
15-1 to 15-16	Feb 1975
A-1 to A-15	Feb 1975
Index 1, 1 to 3	Feb 1975
Index 2, 1 to 2	Feb 1975
Index 3, 1 to 3	Feb 1975

Preface

This manual is a programming guide to DOS-III, a Hewlett-Packard Disc Operating System for 2000-series computer systems. Programmers using this manual should be familiar with the functions of batch-processing operating systems and one of the programming languages supported by the DOS-III Operating System.

The Hewlett-Packard programming languages and program libraries that can operate under control of DOS-III are described in the following reference manuals:

- *HP ALGOL (02116-9072)*
- *HP ASSEMBLER (24307-90014)*
- *HP FORTRAN (02116-9015)*
- *HP FORTRAN IV (5951-1321)*
- *RELOCATABLE SUBROUTINES (02116-91780)*

Other information, which may be useful to the programmer, is included in the SMALL PROGRAMS MANUAL, the MANUAL OF DIAGNOSTICS and the SOFTWARE OPERATING PROCEDURES. These manuals contain custom-assembled modules pertaining to each customer's software and hardware configurations, and are supplied with each Hewlett-Packard computer system.

This manual is divided into six functional parts:

- Part 1. DOS-III OPERATING SYSTEM

  Part 1 defines the standard capabilities of DOS-III. It includes a summary of DOS-III organization, hardware and software; definitions of DOS-III directives, EXEC calls and I/O routines; a description of the interaction of DOS-III and its subsystems; and a set of sample job decks.

- Part 2. DOS-III EXTENDED FILE MANAGEMENT PACKAGE (EFMP)

  Part 2 describes the capabilities of the DOS-III Extended File Management Package (EFMP), which allows the programmer to extend the file-handling capabilities of the DOS-III Operating System. Part 2 contains sections on EFMP organization, EXEC calls and use of UTIL, the EFMP Utility Program.

- Part 3. GENERATING AND LOADING DOS-III

  Part 3 gives complete instructions for generating and loading a DOS-III System.

- Part 4. DOS-III SYSTEMS PROGRAMMING

  Part 4 contains information which will help the advanced programmer to write his own EXEC modules, plan I/O drivers and use the DOS-III privileged mode capabilities.

- Part 5. ERROR CODES AND MESSAGES

  Part 5 is a complete set of all DOS-III Operating System error codes and messages.

- Part 6. APPENDIX AND INDEXES

  Part 6 contains an appendix of DOS-III system tables and three indexes: the first two are convenient summaries of DOS-III directives and EXEC calls; the third refers to terms discussed in the manual.

Contents

TABLES

FIGURES

PART 1
DOS-III Operating System

SECTION I
DOS-III Organization

The DOS-III supervisory software consists of a Disc Monitor (DISCM) that resides in main memory; EXEC modules which may reside either in main memory or on disc; and a Job Processor (JOBPR) that is disc-resident. Together these modules manage I/O processing, interrupt processing, executive processing, job processing, and file handling.

Other DOS-III software consists of a series of relocatable binary software modules. Since each module is an independent, general-purpose program, the hardware and software configuration of the system is flexible. Modules can either reside in main memory or on the disc, at the user's option (specified during system generation). In a system with a small main memory, the modules can reside on the disc to save main memory space; in a large main memory system, modules can reside in main memory for greater efficiency.

MAIN MEMORY LAYOUT

When DOS-III is active, the main memory is divided into a User 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 main memory by the supervisor when needed. The User Area provides space for execution of user programs.

In addition, large DOS-III software modules, such as the FORTRAN Compilers, Assembler, Relocating Loader, and Job Processor, reside on the disc and execute in the User Area. (See Appendix A for figures on disc and main memory layout.)

If the memory protect option is present, a memory protect boundary is set between the System Area and the User Area. This boundary interrupts whenever a user program attempts to execute an I/O instruction (including a HALT) or to modify the System 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.

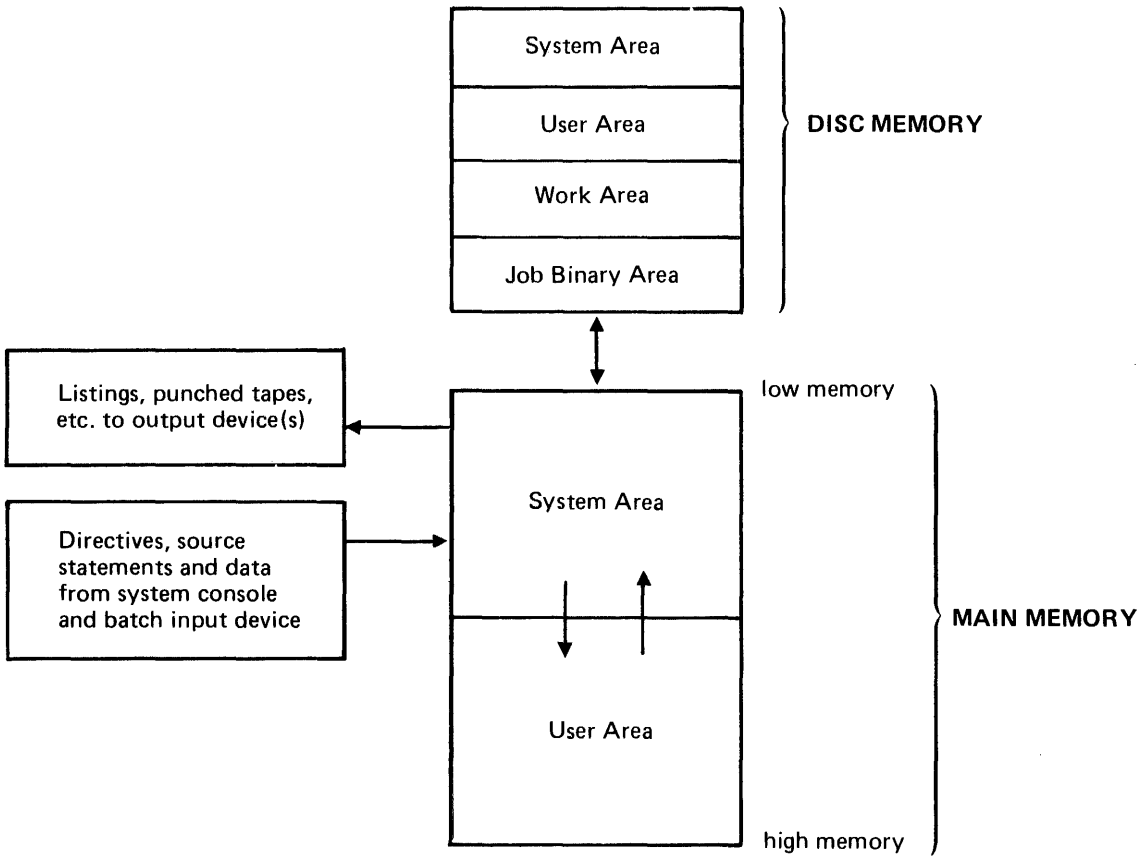


Figure 1-1. Functional Diagram of DOS-III

DOS-III OPERATION

DOS-III operates in either keyboard or batch mode. In keyboard mode, the user enters statements and commands to the system (called directives) to control his programming job through a keyboard device (system console). Each line entered must terminate with a *return* and a *linefeed.* In batch mode, the user enters directives through a batch input device, sometimes integrated with a source program on punched cards, paper tape or magnetic tape, thus forming a job deck. Jobs can be stacked one upon another in a queue.

Deleting Keyboard Errors

To delete an entire line of input, strike *rubout* then *linefeed.* To delete the character just entered, strike *Control-A* (simultaneous "A" and *control* key striking). Each *Control-A* deletes one additional preceding character.

Batch Abort

Some errors when encountered in batch mode cause a batch abort. When such an error occurs (mostly in response to a directive) DOS-III takes the following action:

1. The offending directive and an error message is printed on the list device.
2. JOB ABORTED is printed on both the system console and the list device.
3. The offending statement and subsequent statements are ignored until a JOB, EJOB, or TYPE directive is encountered. The current operation is aborted and the next input is processed.

DOS-III DIRECTIVES

The DOS-III 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-III. They are entered in one of the two modes of DOS-III operation: keyboard or batch.

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

- Create, rename, edit, list, and dump user files (relocatable, absolute, loader-generated, source statements, and ASCII or binary data)
- Search the various disc subchannels for specified file names
- Check status of user disc tracks
- Turn on user programs or system programs such as FORTRAN and Assembler
- Examine and modify the logical organization of the I/O; rewind magnetic tapes and output end-of-file commands to magnetic tapes; output top-of-form commands to list devices
- Start and stop a job; type comments; suspend operations; resume execution of suspended programs

- Assemble or compile, load and execute a user program
- Dump main or disc memory
- Set the date; abort programs; transfer to batch mode (from keyboard mode or batch mode); return to keyboard mode (from batch mode)
- Change the subchannel of the user disc
- Initialize (label) a disc subchannel
- Dump all (or part of) a disc to another disc
- Purge file name entries from the user file directory
- Repack discs to eliminate purged user files
- Reserve logical memory space for specific subsystems (Memory Management)

DOS-III directives are described in Section II.

DOS-III EXEC CALLS

After being translated and loaded, an executing user program communicates with DOS-III by means of EXEC calls. An EXEC call is a JSB instruction which transfers control to the DOS-III 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 or main programs
- Request the time
- Control of memory protect
- Store values into base page memory locations
- Memory Management
- Programmatic file control

DOS-III EXEC calls are described in Section III.

DOS-III INPUT/OUTPUT

All I/O operations and interrupts are channeled through the DISCM section of the DOS-III Supervisor. DISCM is always main-memory resident and maintains ultimate control of the computer resources.

I/O programming is device-independent. Programs written in FORTRAN, ALGOL, and Assembler 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 initially are assigned to appropriate devices by the operator during system generation, depending upon what is available and can be assigned during a job. Thus, the programmer need not worry about the type of input or output device performing the actual operation.

PRIVILEGED INTERRUPT

For DOS-III system interrupt processing, the I/O channel select codes are assigned decreasing priority. Channel 10_8 has the highest priority and channel 37_8 has the lowest. When an interrupt occurs on an I/O channel, system interrupt processing is disabled on all channels having a lower priority (higher number) until the higher priority interrupt processing is completed.

DOS-III provides an optional capability which permits privileged interrupts on specific I/O devices (channels). These devices have their own user-supplied interrupt routines and have their interrupts processed without going through the system's central interrupt processor ($CIC). The system guarantees a response time of 100 microseconds for privileged device interrupts. (For a description of privileged interrupt driver routines, see Section 13.)

The privileged interrupt capability is obtained by including a "fence" board in the system hardware configuration and notifying the system software of the existence of the fence during system generation (see Section 10). The privileged interrupt fence physically separates privileged devices from system devices. Privileged devices are those with interface boards in I/O channels of a lower number (higher priority) than the fence. System devices are those with interface boards in I/O channels with a higher number than the fence.

The DMA channels are always considered system devices although they reside on the privileged side of the fence. When the privileged interrupt option is included in the system, any DOS-III drivers which require DMA interrupts must explicitly inform the system of this fact. This is accomplished by issuing the following subroutine call from the driver before returning control to the system:

```
EXT $SDMA
JSB  $SDMA
```

When the last DMA interrupt has been received, the driver should inform the system that no further DMA interrupts are expected by issuing the following subroutine call:

```
EXT $CDMA
JSB  $CDMA
```

When the privileged interrupt fence is installed in the system and necessary privileged interrupt drivers are included, the user can access his privileged devices with standard I/O calls (JSB EXEC).

TIMING CAPABILITIES

A library subroutine called $TIME is available to both system programs and user programs. The Time Base Generator is required to use this subroutine (see "Hardware Options"). $TIME provides the capability to set, reset, or release a timer (100 millisecond resolution).

> *Note: Upon return from the $TIME subroutine, Memory Protect is disabled until a system request (JSB EXEC) is issued.*

When setting (activating) a timer, an initial time value is placed into a user-supplied buffer and this timer buffer is added to a linked list of currently active timers. When the timer expires, the subsystem, driver, or user receives temporary control from the system. A timer is reset by placing a new time value into an active timer buffer. A timer is released (deactivated) by removing the timer buffer from the linked list of active timers. It is possible to remove all timer buffers from the list with one calling sequence.

To use $TIME, the program must include a timer buffer, a time-out processor routine, and a calling sequence.

Timer Buffer

A 4-word timer buffer must be available to $TIME. The address of this buffer is passed to $TIME to identify the desired timer. Timer buffer format is:

Word 1:	16-bit buffer identifier	
Word 2:	Address of time-out processor routine	
Word 3:	Current time value	System use only
Word 4:	Address of next timer buffer in linked list	System use only

Program must not modify word 3 or 4.

Time-out Processor Routine

Control is passed to the time-out processor routine when a specified timer expires. Unless the system was generated with the privileged interrupt option, the interrupt system will be OFF and should remain OFF during execution of the time-out processor routine. If the privileged interrupt option is included in the system, the interrupt system will be ON upon entry into the time-out processor. To prevent further privileged interrupts from occurring during execution of the time-out processor, the time-out processor must disable the interrupt system.

> **Caution: Interrupts should not be disabled for more than 100 microseconds.**

On entry into the time-out processor routine, the timer buffer is released from the timer list and the A- and B-registers set as follows:

A = 16-bit identifier of the timer just expired (this allows one time-out processor to service many timers).

B = 15-bit address of the timer buffer associated with the expired timer.

Calling Sequence

To set or reset a timer:

```
        EXT   $TIME
          .
          .
          .
        LDA   VALUE     (Time specified in -100 milliseconds)
        LDB   ATMBF     (Address of timer buffer)
        JSB   $TIME     (Set/reset timer)
        SZA             (If A = 0, no error; A = 1, illegal address)
        JMP   ERROR
          .
          .
          .
VALUE   DEC   -2        (Set timer for 200 milliseconds)
```

When this request is received, the list of timers is scanned for a matching timer buffer. If no match is found, a set request is assumed and the new entry is placed in the timer list. If a match is found, a reset request is assumed and the new value is stored into the existing timer buffer.

On return from $TIME, the contents of the A-register indicate the termination condition:

A = 0; normal termination

A = 1; illegal timer buffer address

To release a timer:

```
            EXT   $TIME
              .
              .
              .
            CLA                 (Indicates release request)
            LDB   ATMBF         (Release a specific timer)      }
                or              or                              } Choose one
            CLB                 (Release all timers)            }
            JSB   $TIME         (Release timer)
            <return point>
ATMBF       DEF   TMBUF[,I]     (Address of timer buffer)
TMBUF       OCT   n             (16-bit identifier)             }
            DEF   TOP[,I]       (Address of time-out processor) } Time Buffer
            BSS   2             (Reserved for system)           }
              .
              .
              .
TOP         NOP                                                 }
              .                                                 } Time-out Processor
              .                                                 }
              .                                                 }
            JMP   TOP,I                                         }
```

Note: Routines using $TIME must remain main-memory resident during program execution because the system uses a linked list mechanism to keep track of the timers.

DOS-III FILES

Two types of files can be included in the DOS-III system: standard files (created by the STORE or EDIT directives) and files created under the Extended File Management Package (if EFMP is included in the system).

Standard Files

The disc provides quick access and mass storage for user files consisting of source statements, relocatable, absolute and loader-generated object programs, or 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 system library subroutines.

DOS-III Extended File Management Package

DOS-III installations can use the DOS-III Extended File Management Package (EFMP). This set of optional EXEC modules allows the user to exploit a more powerful file structure than that provided by DOS-III. EFMP files allow logical record sizes of varying lengths for different files, security codes, flexible buffering, sequential reads and writes with a pointer, and detailed status information. In addition, a utility program (UTIL) is available that operates in the User Area. UTIL makes those EFMP functions (except reads and writes), normally only usable through EXEC calls, usable from the keyboard. For more information on EFMP, see Part 2.

DOS-III MEMORY MANAGEMENT

A memory management EXEC module allows user and system programs to allocate and release buffer space within memory. The following memory management capabilities are provided:

- A directive (:MMGT) to specify and list subsystem names and block sizes.
- An initialization call (RCODE=35) to reserve a block of memory under a unique block name.
- A status call (RCODE=36) to interrogate the state of various blocks of memory.
- A buffer allocation call (RCODE=38) to subdivide blocks of memory into individual buffers. A unique buffer identification is assigned each buffer allocated.
- A buffer release call (RCODE=41) to release previously allocated buffer space.

GENERATING A DOS-III SYSTEM

DOS-III is generated and loaded using two programs:

- Configured DSGEN (the system generator)
- BMDL (a bootstrap loader which loads the configured DOS-III from the disc into main memory); or an equivalent program contained on a ROM.

First, DSGEN outputs instructions to the operator asking for information about the system. At the appropriate point in the dialogue, the operator loads in the relocatable binary modules which make up DOS-III and specifies whether the modules are to be disc- or main-memory resident. Finally, DSGEN stores the configured DOS-III system on the disc in absolute form. (The disc is protected from alteration by a hardware override switch.)

DOS-III then resides as a System Area and User Area on the disc. Each area is labeled and contains a directory of all the files contained within the area. The System Area contains system main-memory resident and disc-resident modules, while the User Area contains user files.

To load DOS-III into main memory and begin system execution, the user executes a disc loader. The Loader loads all the modules designated main memory resident into main memory. (The disc-resident modules are brought into main memory when needed by the main-memory resident modules.)

DISC STORAGE

Disc storage is divided into subchannels. Each subchannel is a logical disc, i.e., the dimensions do not necessarily correspond to the physical characteristics of the disc. Each subchannel contains 203 tracks — typically three of which are reserved as spares. The smallest addressable unit on a disc is a sector. One sector contains 128 sixteen-bit words of storage.

HP 2883/2884

During system generation, the HP 2883 disc drives can be configured for one of two modes — four subchannels per drive or two subchannels per drive. In either case, the controller supports one or two drives (one drive is required).

For the four subchannel per drive mode, each drive contains a removable pack of twenty disc surfaces divided into four subchannels. Thus, the controller can support up to eight subchannels.

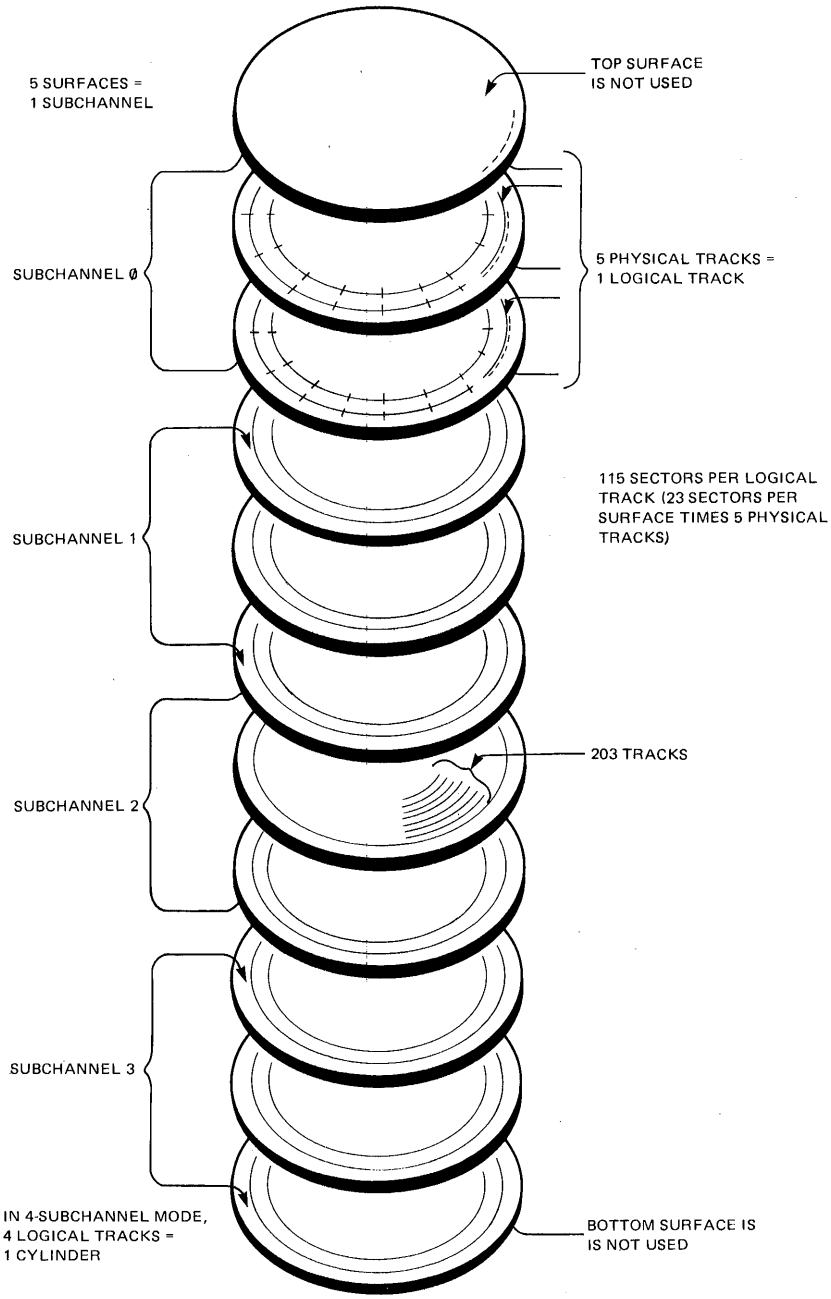
For the two subchannel per drive mode, each drive contains a removable pack of twenty disc surfaces divided into two subchannels. One controller supports up to four subchannels. A second controller (optional) can be added to provide support for up to eight subchannels. Subchannel assignments follow:

2883/2884 with four Subchannels per drive			**2883/2884 with two Subchannels per drive**			
0	1	Disc Drive Numbers	0	1	2	3
0	4	Subchannel Assignments	0	2	4	6
1	5					
2	6		1	3	5	7
3	7					

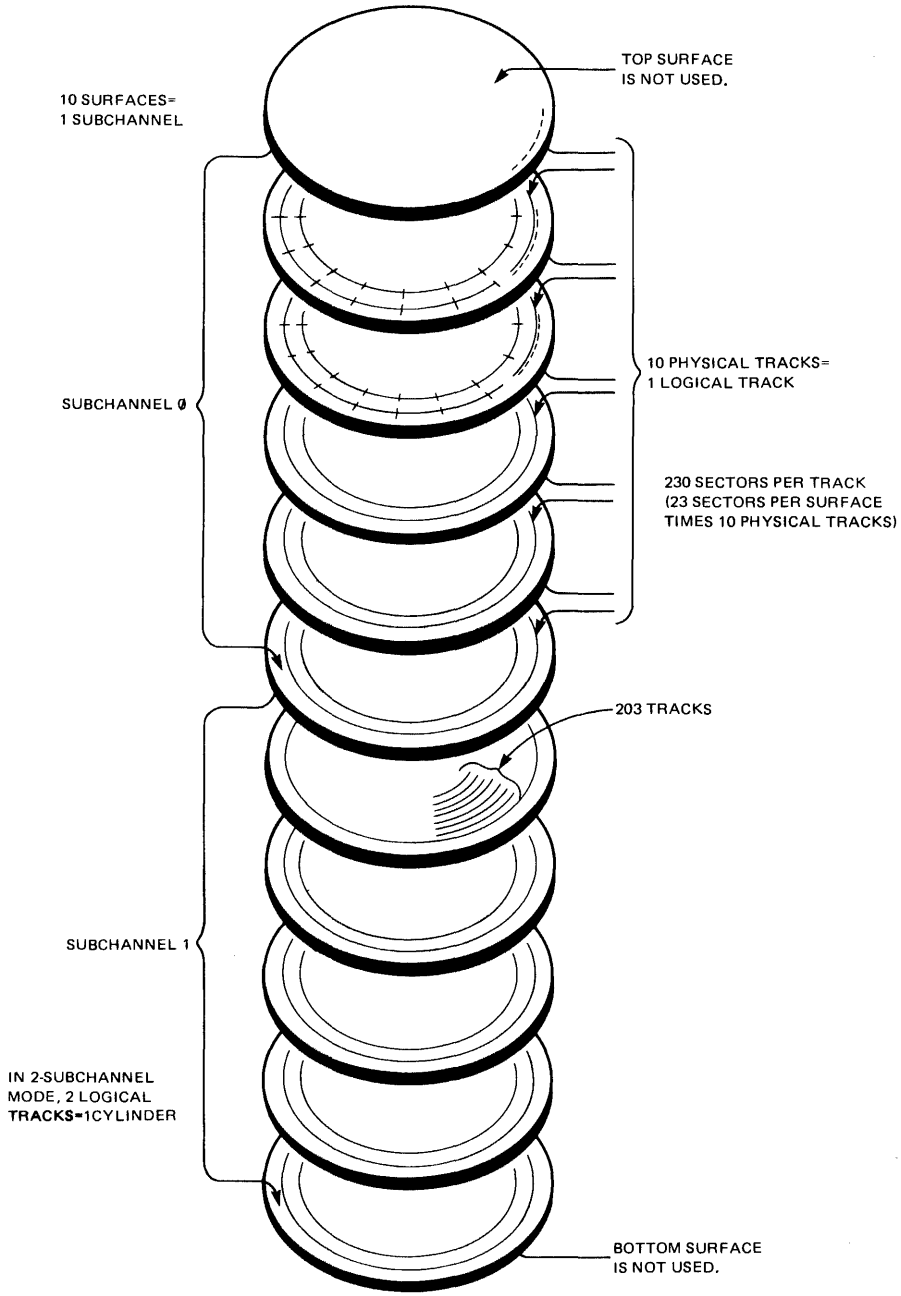
When two controllers are used (two subchannels per drive mode only) they must reside in contiguous I/O channel slots. In addition, the subchannels associated with the second controller (subchannels 4 through 7) can contain only user discs — no generation or bootstrap operations are permitted on these subchannels.

When an HP 2883/2884 is configured to the four-subchannel mode, each track contains 115 sectors. If it is configured to the two-subchannel mode, each track contains 230 sectors. Perhaps the concept of logical disc organization can be more clearly understood by studying the accompanying illustrations.

HP 2883/2884
4-SUBCHANNEL MODE
LOGICAL ORGANIZATION



HP 2883/2884
2-SUBCHANNEL MODE
LOGICAL ORGANIZATION

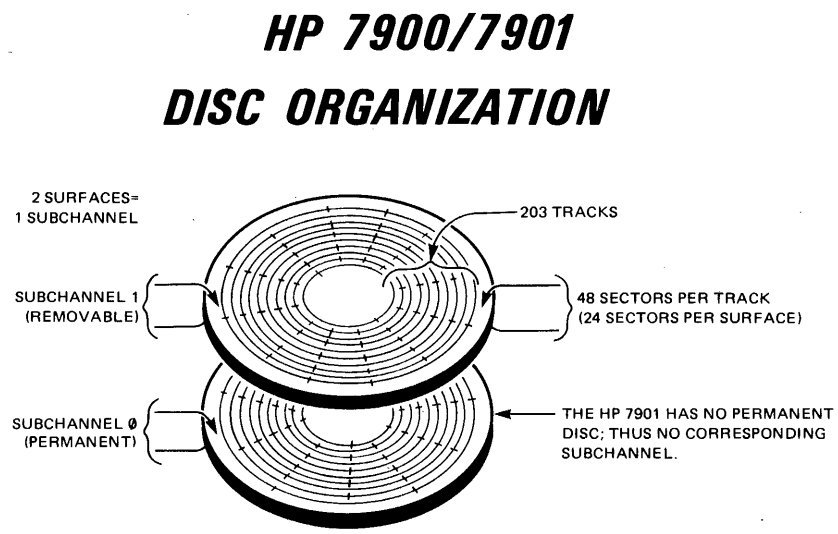


HP 7900/7901

The controller for the moving-head disc supports up to four disc drives (one is required). Each 7900 drive contains two discs: a fixed disc and a removable cartridge. Each 7901 drive contains one disc: a removable cartridge. Each disc is referenced through a subchannel of the controller. Therefore, the controller has a maximum of eight subchannels (numbered 0 to 7). The subchannels are assigned as follows:

7900					7901			
0	1	2	3	Disc Drive Numbers	0	1	2	3
1	3	5	7	Removable Subchannels	1	3	5	7
0	2	4	6	Permanent Subchannels	None			

On the HP 7900/7901 disc drive, each track on the disc contains 48 sectors as shown in the following illustration.



DISC USAGE

DOS-III normally allows two subchannels to be available to the user: one subchannel contains the system disc and the other contains the user disc (which may be the same subchannel as the system disc). The user subchannel assignment 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-III 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 subchannel, 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.

DOS-III HARDWARE REQUIREMENTS

DOS-III controls the operation of HP 2100A and HP 2100S Computer systems, and HP 21MX Computer Series systems. Minimum hardware requirements depend on the type of computer system selected.

The minimum hardware required for DOS-III operation is:

1. a) An HP 2100A or HP 2100S Computer, with 16,384 words of main memory, and DMA; or,

   b) An HP 21MX-series Computer with 16,384 words of main memory, and a Dual-Channel Port Controller.

2. Moving-head Disc device (HP 7900 Moving-head Disc Drive with fixed disc and removable cartridge; or HP 7901 Moving-head Disc Drive with removable cartridge; or HP 2883 Disc File with one removable pack).

3. System Console device.

4. Paper Tape Reader.

Hardware Options

The following hardware options are available:

1. Time-base Generator (provides accounting times and time-of-day).
2. Privileged Interrupt Fence.
3. Floating-point hardware (standard for 21MX Computer Series).
4. Additional main memory to a total of 24,576 or 32,768 words.
5. Using extenders, additional I/O channels (up to channel 37_8).
6. Memory Protect (not available for the HP 2105 Processor).
7. Paper Tape Punch.
8. Line Printer.
9. Card Reader.
10. Magnetic Tape Unit.
11. Additional Disc Drives. (Maximum is four on HP 7900/7901; two on HP 2883 with four subchannels per drive; and four on HP 2883 with two subchannels per drive.)
12. CRT Display Console.
13. Writable Control Store.
14. Fast FORTRAN Processor.

DOS-III SOFTWARE

Required Software

The minimum software requirements for DOS-III are

1. Absolute Programs
   a. DOS-III System Generator (DSGEN)
   b. DOS-III Bootstrap Loader
   c. SIO Drivers
2. Relocatable Programs
   a. DOS-III Disc Monitor (DISCM)
   b. DOS-III Exec Modules
   c. DOS-III Job Processor (JOBPR)
   d. DOS-III Disc Driver (DVR31)
   e. DOS-III System Console Driver (DVR00, DVR05 or DVR26)
   f. DOS-III Paper Tape Reader Driver (DVR01)

Software Options

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

1. DOS-III Relocating Loader
2. DOS-III Assembler
3. DOS-III FORTRAN Compiler
4. RTE/DOS FORTRAN IV Compiler
5. RTE/DOS FORTRAN IV Compiler — 10K Compiler Area
6. RTE/DOS ALGOL Compiler
7. RTE/DOS Relocatable Library (EAU, or floating point)
8. RTE/DOS FORTRAN IV Library (extended-precision arithmetic)

9. RTE/DOS FORTRAN Formatter

10. DOS-III Standard Drivers (either main-memory or disc resident):

    Paper Tape Punch Driver (DVR02)

    Digital Plotter Driver (DVR10)

    Card Reader Driver (DVR11) — uses DMA or Dual Channel Port Controller

    Line Printer Driver (DVR12)

    Optical Mark Reader Driver (DVR15)

    Magnetic Tape Driver (DVR23) — uses DMA or Dual Channel Port Controller

    Terminal Printer Driver (DVR26)

    Writable Control Store Driver (DVR33) — uses DMA

    Card Reader Punch Driver (DVR34)

    Hardwired Serial Interface Driver (DVR67)

11. DOS-III Physical Drivers

    Synchronous Data Set Interface Driver (DVR70)

    Synchronous Modem Interface Driver (DVR71)

    Asynchronous Data Set Interface Driver (DVR72)

    Asynchronous Multiplexer Interface Driver (DVR73)

    Buffered Asynchronous Data Set Interface Driver (DVR74)

12. DOS-III Logical Drivers

    Asynchronous Terminal Driver Number One (ATD01)

    Asynchronous Terminal Driver Number Two (ATD02)

    Asynchronous Card Reader Driver Number One (ACR01)

    Page Mode Terminal Driver Number One (PMT01)

    Page Mode Terminal Driver Number Two (PMT02)

    Synchronous Line Control (SLC)

13. DOS-III Extended File Management Package

14. RTE/DOS Fast FORTRAN Processor Subroutine Library

SECTION II
DOS-III Directives

Directives are the direct line of communication between the keyboard or batch input device and DOS-III. Directives may enter DOS-III in two modes: keyboard and batch. In either mode, all directives are listed on the system console. Certain directives can be used in one mode only; others can be used in both modes. In keyboard mode, the operator manually inputs the directives through the system console keyboard. In batch mode, the programmer prepares the directives (commonly on punched cards, paper tapes, or magnetic tape) and inputs them along with programs, data, etc., in a complete job.

FORMAT FOR DIRECTIVES

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

:PURGE,FILE1,FILE2,FILE3

When the sequence and position of parameters is significant, missing parameters 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-III.

Note: The total length of an input string cannot exceed 72 characters.

ENTERING DIRECTIVES

DOS-III has two conventions for notifying the operator that directives may be entered:

1. DOS-III outputs a "commercial at" sign (@) and rings a bell (at the system console). At this time, the operator may enter any directive.

2. DOS-III outputs an asterisk (at the system console). At this time the operator may enter an "operator attention" directive only. The "operator attention" directives are

:ABORT

:DN

:EQ

:LU

:OFF

:PAUSE

:TRACKS

:TYPE

:UP

Should the operator type any other directive, DOS-III outputs the following message:

IGNORED

and returns to the executing program.

To attain control of DOS-III (to enter an "operator attention" directive) the operator can strike any system console keyboard key. If the system console is available, DOS-III immediately outputs an asterisk (*); if the system console is busy, DOS-III will output the asterisk as soon as it releases the system console.

Notes: 1. Operator attention is disabled during the completion phase of :EDIT and during :PURGE.

2. Some system conditions restrict allowable directives; e.g., after an I/O ERR NR EQT# nn, the system is waiting for an :UP,nn, followed by :GO. Under such conditions, otherwise legitimate directives will be ignored.

3. Some operations, such as editing, require perceptible waits while DOS-III processes the directive.

ORDER OF DIRECTIVES

The DOS-III directives described in this section are presented in alphabetic order (by function name). If a directive must be used in keyboard mode only, a note to that effect is placed at the top of each page describing the directive. A quick cross-reference index of DOS-III directives, "Summary of Directives," is included at the back of this manual.

ABORT

Purpose

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

Format

:ABORT

Comments

Abort carries out all the operations of a batch mode EJOB directive. All I/O devices are cleared.

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 logical unit number of the desired batch input device.

Comments

A BATCH, JOB, TYPE, OR TRACKS directive must be the first directive entered following system start.

See "TYPE" in this section for the opposite procedure of returning batch mode to keyboard mode. Assigning a null device or logical unit numbers 2 or 3 as the batch device results in an ILLEGAL LUN error (see LOGICAL UNIT directive).

CLEAR

Purpose

To clear the Job Binary Area on the disc, or to issue a clear command to an I/O device.

Format

:CLEAR[,logical unit]

where *logical unit* is the logical unit number of the device to be cleared. If *logical unit* is omitted, the disc Job Binary Area is cleared.

Comments

Using logical units 1, 2, or 3 results in an LU error.

The effect of clearing an I/O device is the transmittal of a clear function to the appropriate driver.

COMMENT

Purpose

To print a message on the system console.

Format

:COMMENT character string

where *character string* is a message to be printed on the system console.

Comments

A space (but not a comma) is required between the directive word and the comment string.

The programmer can use :COMMENT or :PAUSE to send a message to the operator at the system console; using :COMMENT causes no suspension of processing. Use :PAUSE when a processing delay is desired, for example to request that the operator mount a magnetic tape.

EXAMPLES:

:COMMENT BEGINNING OF PAYROLL JOB

DATE

Purpose

To set the date and time for accounting purposes whenever DOS-III is activated.

Format

:DATE,day[,hour,min]

where *day* is any string of ten or fewer characters (commas not permitted) chosen by the operator (such as 7/10/69, 10.JULY.69, etc.);

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

Comments

The DATE directive is legal only as the first directive in a start-up procedure. The directive is not accepted any other time.

EXAMPLES:

:DATE,7/10/69,12,23

:DATE,WEDNESDAY,7,45

:DATE,10JULY1969

:DA,,

DOWN

Purpose

To declare an I/O device unavailable for use during the remainder of a job.

Format

:DN,n

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

Comments

The system console and the disc (logical units 1, 2, and 3) cannot be set down.

DUMP (DISC-TO-DISC)

Purpose

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

2. To dump the System Area (including system buffer) onto another subchannel (:DD,X)

3. To dump all or specified files of the User Area (optionally assigning some new file names) onto another subchannel (:DD,U ...) or, onto the current subchannel (assigning new file names).

Formats

1. *:DD*

2. *:DD,X*

3. *: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).

Note: No more than 14 parameters can be specified after :DD,U.

The destination disc must be specified by a :UD immediately following the :DD. Any other directive will negate the :DD. (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 "INITIALIZE")

The SS directive does not apply to :DD.

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

TRAC # TOO BIG

If DOS-III cannot find some of the files specified to be dumped, the message

file

UNDEFINED

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

If a file specified to be dumped has the same name (after the optional renaming) as an existing file on the destination disc, the message

file

DUPLICATE FILE-NAME

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

Caution: A DOS-III system created through the :DD directive (disc-to-disc dump) cannot be protected with the Protect/Override switch on the disc drive because the protect bits on the system portion of the original disc are not copied during the dump operation.

DUMP (FILE)

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 logical unit number of 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 (including magnetic tape). The dump format varies with the type of file and the type of device. See Table 2-1.

Table 2-1. :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
Absolute binary	Absolute binary records	8 octal words/line
Relocatable binary	Relocatable binary records (loadable)	8 octal words/line
Source statements	1 statement/record	1 statement/line

Note: Sector numbers on listings are not related to the S1 and S2 parameters.

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.

An error message occurs when *S1* > *S2*, or when either *S1* or *S2* is greater than the length of the file.

Source statements, relocatable binary and absolute binary files can be dumped to a punch device and later restored by using the appropriate STORE directive. In general, however, this cannot be done with ASCII data and binary data files.

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 system console:)

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

(On the list device:)

```
001   000000  062125  072121  114535  010010  010075  010156  010100
      002400  052100  026014  026036  062006  042154  072023  114535
      010025  010076  010077  010006  010153  114535  010033  010076
      010077  010101  010117  102501  002002  026056  062006  072046
      114535  010050  010123  010076  010127  010124  010006  010122
      114535  010056  010076  010077  010126  010153  036006  036006
      036006  036121  026003  114535  010071  010076  010077  010106
      010120  114535  010074  010074  000006  000022  000002  000001
      000000  020116  047524  020106  047525  047104  020120  051117
      043522  040515  020103  047515  050114  042524  042504  000005
      000011  000000  000000  000016  000002  177746  020040  020040
      020040  020040  020040  020040  020040  020040  020040  020040
      020040  020040  020040  020040  020040  020040  020040  020040
      020040  020040  020040  000003  177777  020040  020501  040440
      020040  041102  041040  020040  041503  041440  020040  042104
      042040  020040  042505  042440  020040  043106  043040  020040
```

DUMP (PROGRAM)

Purpose

To request that a user program be dumped to the standard list device (logical unit 6) 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] [,S]
:ADUMP[,FWA[,LWA]] [,B] [,S]

where *FWA* is the octal address, relative to the program origin, of the first word to be dumped
LWA is the octal address, relative to the program origin, of the last word to be dumped
B means dump the base page linkage area of the program
S means dump the entire system area.

If *LWA* is missing, the entire program, starting with *FWA*, is dumped. *B* alone dumps all the main program, plus base page linkages, but not the system routines. *S* alone dumps only the system.

If no parameters are given, everything except the system area is dumped.

Comments

The dump directives, PDUMP and ADUMP, must precede the RUN or PROG request in a job. They implicitly refer to the next program to be executed. DOS-III sets a flag when it encounters either PDUMP or ADUMP, then checks the flag the next time a program is executed. Only one of the requests will be honored, depending upon whether the program runs normally or is aborted. The dump is labeled accordingly. These flags are cleared when a program terminates.

Any parameter following *S* in the directive is ignored. If *FWA* is greater than *LWA*, this message is output:

LIMIT ERROR

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*
adr_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 octal 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 main memory). 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	*item-1*	*item-2*	*item-1*	*item-2*	*item-1*	*item-2*	*item-1*	*item-2*

Note: :OFF before a program executes clears the dump flags.
:OFF during a program execution causes an abort dump.
:OFF during a dump terminates the dump.

EXAMPLE:

```
:ADUMP,0,15,B  (Set up dump flag)
:RUN,PRG9,6    (Run program)
LU     012140

(Main program dump)
ADUMP
12000   160001  002002  130573  170574  006004  160001  002003  026012
12010   130575  170576  006004  160001  170577  006004  160001  170600

(Page Eject)
(Base page dump)

00570   010137  002045  010711  003237  010763  002045  017014  000300
00574   017641  000000  017015  000400  017641  000406  017601  000000
00600   017650  000000  017615  000000  017664  000000  017662  000573
00604   017637  000573  017571  177205  017563  001204  017714  017715
00610   017562  021121  017534  021122  017536  021122  017633  160656
00614   017544  037626  017546  037626  017673  000000  017605  000040
```

DUMP (SECTOR)

Purpose

To dump any specified sector or sectors of the current user disc on the standard list device (logical unit 6) 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

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 system console:)

:SO,0,1

@

(On the list device:)

```
001    000000  067767  017570  067744  077743  017613  017613  017613
       017613  064120  007004  077310  064117  044055  160001  044051
       010072  073773  053774  077761  053775  077762  077304  044056
       160001  001727  013733  073305  050060  027460  053763  027445
       067304  044066  037310  027415  027505  044052  160001  023773
       033774  170001  063773  073302  002004  073303  063774  073773
       067304  160001  073766  164000  017570  063305  050060  027440
       006004  160001  033773  170001  006004  063730  170001  006004
       003004  170001  067304  077311  027440  060154  001722  013765
       033774  001727  001723  070154  063761  067302  017606  063762
       067303  017606  002400  067774  017606  063311  067775  017606
       067761  006003  027540  044055  160001  023774  033302  170001
       067762  006003  027546  023775  033303  170001  063776  001200
       067777  006003  002004  064155  070155  054175  070175  006400
       050175  064115  074200  047740  074157  064175  074161  124003
       000000  057766  127570  037766  163766  002021  027571  013764
```

EDIT

Purpose

To perform listed edit operations on a user source file (follows the :SS condition).

Format

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

where *file* is the name of a source file (the primary file) 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 destructively replaces the old file. (Follows :SS in searching for duplicate file names.)

Comments

An edit list consists of one or more edit commands and, optionally, a series of associated source statement (i.e., following REPLACE, INSERT). Edit operations are executed when they are entered. When using the system console, the operator must not enter the next operation until the "@" prompt is output on the console.

All edit operations begin with a slash(/), and only the first character following the slash is required. The rest are ignored (until a comma is reached).

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 statement of the operation, inclusively. 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. Sequence numbers refer to the original sequence of the unedited file; inserted statements cannot be referenced until the current editing process is completed and the file automatically resequenced prior to another EDIT directive.

Source statements following /REPLACE or /INSERT on the current batch device cannot contain a colon (:) in column 1, although those entered from the system console can, with the exception of :OFF and :ABORT (which are interpreted as directives instead of data). Source statements can never contain a slash (/) in the first column. Source statements on any device other than the system console and the current batch device can contain anything else in column 1 (including :OFF or :ABORT).

Input is terminated only by an /END.

If the edit file is entered on the system console and either a

PARAMETER ILLEGAL

or

NO SOURCE

error occurs, the user merely re-enters the statement in error. If the edit list is entered on any other device, the EDIT directive is aborted (if the EDIT directive was entered in keyboard mode) or the entire job is aborted (in batch mode).

EDIT OPERATIONS

/DELETE,m[,n]

Deletes source statements *m* through *n*, inclusively, from the source file. If only *m* is specified, that one statement is deleted.

/INSERT,m

Inserts the source statements in the edit list immediately following this command into the primary file following statement *m*.

/MERGE[,k],secondary file[,m[,n]]

Merges source statements from the secondary file into the primary file named in the EDIT directive.

k is the sequence number of the primary file (named in the EDIT directive) after which source statements of the secondary file are merged. If *k=0*, the secondary file source statements are merged at the beginning of the primary file; if *k* is omitted, the secondary file source statements are merged at the end of the primary file.

Secondary file is the name of the source file to be merged with the primary file. If *m* and *n* are specified, then only lines *m* through *n* of the secondary file are merged. If only *m* is specified, then only that one line is merged.

/REPLACE,m[,n]

Replaces source statements *m* through *n* (inclusively) in the primary file with source statements following the /R in the edit list. If *n* is omitted, then only statement *m* is replaced.

Note: Directives cannot be inserted or replaced but can be merged from another file.

/SUPPRESS

Suppresses echoing of the edit operations on the system console, providing that the logical unit specified in the EDIT directive was not the system console. Normally, echoing occurs after each EDIT directive unless /S is entered.

/UNSUPPRESS

Resumes echoing of the edit operations on the system console.

/END

Terminates the edit file and returns DOS-III to its previous mode for further directives. (The last edit command must be /END.)

EXAMPLES:

If a file named SOURC contains:

```
Statement 1     ASMB,R,B,L
Statement 2             NAM START
Statement 3     A       EQU 30
Statement 4     B       EQU 20
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 100
B       NOP
/D,4
/I,6
        STA B
/E
```

then the new file SOURC equals:

```
Statement 1     ASMB,R,B,L
Statement 2             NAM START
Statement 3     A       EQU 100
Statement 4     B       NOP
Statement 5     START NOP
Statement 6             LDA A
Statement 7             STA B
Statement 8             END
```

Assume now that there exists a source file named FILE2:

```
Statement 1     ALF,ALF
Statement 2     JMP START
```

To merge FILE2 into the new SOURC, the following EDIT directive, along with its edit list, is required:

```
:ED,SOURC,5
/M,7,FILE2
/E
```

The new file SOURC looks like this:

```
Statement 1     ASMB,R,B,L
Statement 2             NAM START
Statement 3     A       EQU 100
Statement 4     B       NOP
Statement 5     START NOP
Statement 6             LDA A
Statement 7             STA B
Statement 8             ALF,ALF
Statement 9             JMP START
Statement 10            END
```

END-OF-FILE

Purpose

To write an end-of-file mark on a magnetic tape.

Format

:EF[,logical unit]

where *logical unit* is the logical unit number of the desired magnetic tape (default is 8).

END-OF-JOB

Purpose

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

Format

:EJOB

Comments

The EJOB directive outputs a message recording the total run time of the job and execution time, then returns to keyboard mode.

If :SS condition is active, :EJOB purges temporary files on all specified *user* subchannels. If :SS condition is not active, :EJOB purges temporary files on the current user subchannel. (See STORE directive and "DOS-III Relocating Loader," Section V.) All directives except :TRACKS, :OFF, :TYPE 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 that subchannel 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 = 0007 MIN. 52.6 SEC. EXEC = 0001 MIN. 21.0 SEC.

or

END JOB START

This message is printed on the system console and on the standard list device (logical unit 6). A top-of-form is issued on the list device prior to the message.

EQUIPMENT TABLE

Purpose

To list one or all entries in the equipment table on the system console (see Appendix A for equipment table format).

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 DVRmm d r Uu Ss

where *nn* is the decimal number of the entry
vv is the octal channel number of the device
mm is the I/O driver number for the device
d specifies DMA if equal to D, no DMA if zero
r specifies main-memory resident if equal to R, disc-resident if zero
u is a single decimal digit used for subchannel addressing
s is the availability status of the device:
0 for not busy, and available,
1 for disabled (down),
2 for busy

EXAMPLE:

Following is a listing of a DOS-III Equipment Table.

```
:EQ
EQT 01 CH 11 DVR05 0 R U0 S0
EQT 02 CH 13 DVR01 0 0 U0 S0
EQT 03 CH 14 DVR31 D R U0 S0
EQT 04 CH 16 DVR02 0 R U0 S0
EQT 05 CH 20 DVR12 0 R U0 S0
EQT 06 CH 21 DVR11 D 0 U0 S0
EQT 07 CH 22 DVR23 D 0 U0 S0
0
```

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 0 and 32767.

Comments

When a program suspends itself (see "Program Suspension" in Section III), 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 EXEC (7)
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, and to destroy an existing System Area (and, optionally, a User Area).

Format

:IN,label

where *label* is a six-character name to be written on the disc, or "*" which means unlabel the disc.

Comments

Four basic cases are possible:

1. *:IN,** An unlabeled disc (a disc containing only a User Area). The user directory and all user files are destroyed.

2. *:IN,** A labeled disc. The message

   DOS (or TSB) LABEL xxxxxx

   OK TO PURGE?

   is output. To purge both the System and User Areas, the operator must respond with

   YES

   If the existing label is SYSTEM (the disc contains a DOS or TSB system), the Override/Protect switch must be in the override position (if the disc was created using DSGEN); otherwise, a HLT 31 will occur. If the operator responds with

   NO

   the directive is ignored.

3. *:IN,label* An unlabeled disc. Only the label is changed; no files are destroyed.

4. *:IN,label* A labeled disc. The message

??? LABEL xxxxxx

OK TO PURGE?

is output. To purge an existing DOS or TSB system, move the user files to the beginning of the disc, and assign the new label to the User Area, respond with

YES

If the existing label is SYSTEM (the disc contains a DOS or TSB system), the Override/Protect switch must be in the override position (if the disc was created using DSGEN); otherwise, a HLT 31 will occur. If the operator responds with

NO

the directive is ignored.

Initialization does not affect the protect bits. They remain set.

Refer to Section VI for an example of how to copy the System from one subchannel to another.

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 a non-numeric character) which identifies the job.

Comments

A JOB, BATCH, TYPE, or TRACKS directive must be the first directive entered following system start.

When DOS-III processes the JOB directive, it issues a top-of-form to the list device (logical unit 6), prints an accounting message on the system console 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).

For example,

```
:JOB,START
JOB START MON 6.16.9        TIME = 0013 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.

LIST

Purpose

To list file information recorded in the user or system directories; or to list and sequentially number the contents of all or part of a source file.

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.)

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

where *X* specifies the System Area directory

U specifies a User Area directory

S specifies a user source file

logical unit specifies the list device

file$_1$, . . . names up to 13 entries to be listed (if none is specified, the entire directory is listed)

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

Comments

A top-of-form is issued to the list device prior to listing.

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 FWAM PB
SUBCHAN = *n*

The following lines are then printed:

name type sctrs trk sec $lower_p$ $upper_p$ $lower_b$ $upper_b$ *entry fwam p-b*

where *name* identifies the file,
type tells what kind of file *name* is

AB = absolute binary program AD = ASCII data BD = binary data RB = relocatable binary program SS = source statements	}	User File Only
LB = library XS = supervisor module	}	System File Only
DR = disc resident I/O driver 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 AB, 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 upper 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,
fwam is the octal address of the first word of available memory following the program, and
p-b is equal to T if the file is temporary and will be purged by :EJOB unless stored by :STORE,P.

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

file UNDEFINED

SOURCE LISTING OUTPUT

Each source statement is preceded by a four-digit decimal sequence number.

If the requested file is not a source file, the following message appears,

file
ILLEGAL

The list is terminated by the message

***** LIST END *****

EXAMPLES:

(on the system console:)
:LI,U,6
@

(On the list device:)

NAME	*TYPE*	*SCTRS*	*DISC*	*ORG*	*PROG LIMITS*		*B.P. LIMITS*		*ENTRY*	*FWAM*	*PB*
SUBCHAN=4											
EX9	*SS*	*00080*	*T001*	*000*							
EXM	*RB*	*00063*	*T004*	*008*							
BBB	*SS*	*00001*	*T006*	*023*							
SRCH	*RB*	*00003*	*T007*	*000*							
SSERH	*UM*	*00002*	*T007*	*003*	*10000*	*10271*	*00713*	*00713*	*10000*	*10271*	*T*
ASCII	*AD*	*00200*	*T007*	*005*							
BINRY	*BD*	*00300*	*T015*	*013*							

Note: T in the "PB" column means that the entry is temporary.

(On the system console:)

: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  FWAM  PB
SUBCHAN=4
EX9    SS    00080   T001  000
EXM    RB    00063   T004  008
BBB    SS    00001   T006  023
SRCH   RB    00003   T007  000
SSERH  UM    00002   T007  003  10000 10271  00713 00713 10000  10271
ASCII  AD    00200   T007  005
BINRY  BD    00300   T015  013
```

Note: "PB" no longer equals T.

(On the system console:)

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

@

(On the list device:)

```
0926  ASMB,L,R,X,C,N,B
0927        HED DUMMY $LIBR AND $LIBX FOR RTS SIMULATION ON DOS
0928        NAM DUMRX,6
0929        ENT $LIBR,$LIBX
0930        SPC 2
0931  *     CALLING SEQUENCES:   ENTRY       TERMINATION
0932  *
0933  *
0934  *     PRIVILEGED           JSB $LIBR   JSB $LIBX
0935  *                          NOP         DEF (PROGRAM ENTRY POINT)
0936  *
      ****  LIST END ****
```

LOGICAL UNIT

Purpose

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

Format

:LU[,n_1[,n_2]]

where n_1 and n_2 (if present) are decimal numbers.

If neither n_1 nor n_2 is present: the entire device reference table is printed.

If only n_1 is present: the equipment table entry number assigned to logical unit number n_1 is printed. (See EQUIPMENT TABLE directive.)

If both n_1 and n_2 are present (and n_2 does not equal zero): the device recorded in equipment table entry n_2 is assigned to logical unit n_1.

If both n_1 and n_2 are present (and n_2 does equal zero): the logical unit specified by n_1 becomes a null device, and any I/O request on that device is ignored.

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 generated. This insures a standard I/O organization for all users.

If n_2 = 0 (that device is to be made null), the logical unit specified by n_1 may not be equal to 1, 2, 3, or the logical unit number of the current batch device.

EXAMPLE:

```
:LU
LU01  EQT03
LU02  EQT01
LU03  EQT01
LU04  EQT05
LU05  EQT04
LU06  EQT06
LU07  EQT07
LU08  EQT02
LU09  EQT00   (null device)
@

:LU,9,5       (Logical unit 9 becomes punch)
:DU,9,FILE1   (Dumps FILE1 to punch)

:LU,9         (Checks EQT for LU9)
LU09  EQT05
@

:LU,6,0       (Assigns line printer to null device)
:PR,FTN4,99   (Reads from paper tape reader, no list, object to JBIN)
```

MMGT

Purpose

To reserve logical memory address space for specific subsystems.

> *Note: This directive applies to memory associated with system programs only. Memory associated with user programs is strictly under program control.*

In addition, this directive may be used to obtain a report of memory space previously reserved for subsystems.

Format

:MMGT[,subsystem-name$_1$*, wwwww*$_1$*, subsystem-name*$_2$*, wwww*$_2$*, . . ., subsystem-name*$_n$*, wwwww*$_n$*]*

subsystem-name is a 4-character ASCII name defined for a subsystem at system generation.

wwwww is the number (decimal) of logical words to be reserved for the associated subsystem.

If no parameters are entered, the directive is interpreted as an inquiry request and a list of subsystem names and the number of reserved words previously set is printed on the console. The list appears in the following form:

SUBSYSTEM	*WORDS*
subsystem-name$_1$	*wwwww*$_1$
subsystem-name$_2$	*wwwww*$_2$
.	.
.	.
.	.
subsystem-name$_n$	*wwwww*$_n$

Comments

The :MMGT directive is entered just prior to the :PROG or :RUN directive and reserved memory space is released at program termination. If the subsystem name specified was not defined at system generation, the system prints:

subsystem-name — UNDEFINED

where *subsystem-name* is the 4-character subsystem name. Any defined subsystem names is included in the parameter string are accepted.

If an attempt is made to update or display the subsystem table and no subsystems were defined when the system was generated, the system prints:

NO SUBSYSTEMS DEFINED

If the cumulative sum of words requested for subsystems exceeds the amount available, the system prints:

LIMIT ERROR

Any requests up to the available space limit are accepted. If more than one subsystem name is included in the parameter string, the user may determine which requests have been accepted by entering the :MMGT directive with no parameters. This causes a list of subsystem names together with the number of words reserved for each name to be printed on the console.

Note: The subsystem names discussed here must be included as entry points (ENT) within the associated subsystem routines which are included as part of the system at generation time.

OFF

Purpose

To abort the currently executing user program or 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 cancels any pending DD, ADUMP, or PDUMP directives, unless a program is running, in which case, a pending :ADUMP is executed.

PAUSE

Purpose

To interrupt the current job, optionally print a comment on the system console, and return to the system console for operator action.

Format

:PAUSE [character string]

Comments

PAUSE may be entered through the keyboard even when DOS-III 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-III to the job in the previous mode.

EXAMPLE:

```
:PAUSE MOUNT MAG TAPE    (Operator mounts magnetic tape)
:GO
```

PROGRAM

Purpose

To turn on (i.e., load from the disc and begin executing) a program from the System Area or a program from the User Area which was generated with the DOS-III 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, FTN4 for the RTE/DOS FORTRAN IV Compiler, ASMB for the DOS-M Assembler, LOADR for the DOS-III Relocating Loader, or ALGOL for the RTE/DOS ALGOL Compiler.

A user program is specified via the file name assigned by the DOS-III Relocating Loader (the name specified in the program's PROGRAM, HPAL, or NAM statement).

P_1 through P_5 are optional parameters which DOS-III 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 V for the parameters required by FTN, FTN4, ASMB, ALGOL, and LOADR. Additional programs may be added during system generation, if desired.

Note: User programs can be run using :PROG or :RUN. :PROG is useful when the program needs parameters. DOS-III first searches the user files for the program, then the system files. :RUN is useful when an execution time limit is desired (and a Time-base Generator is present).

EXAMPLES:

:PROG,FTN,2,99

:PROG,MYFIL,0,3,84

PURGE

Purpose

To remove the directory entry associated with a user file. (Follows the :SS condition.)

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. The directory entry for the specified file name is purged (marked for removal)

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

Comments

After the directory entries are purged, the remaining User Area files may be repacked for efficiency by entering the :RPACK directive. However it should be noted that the repacking function is performed automatically each time an :EJOB directive is executed.

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's directory entry is purged, DOS-III prints its name on the system console.

The presence of undefined files in the list has no effect on the purging of named (and existing) entries. However, if an entry cannot be found, this message is output to the system console:

file UNDEFINED

The fastest way to purge all files on a single disc is to use :IN,* (see "Initialize" in Section 2).

CAUTION: OPERATOR ATTENTION IS DISABLED DURING :PURGE.

EXAMPLE:

Original contents of user directory: *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

RENAME

Purpose

To rename a specified user file and, optionally, change its program type. (Follows the :SS condition.)

Format

:RNAME,oldname,newname[,type]

where *oldname* is the name of the user file to be renamed
newname specifies the new name for the file
type specifies the new type for the file.

Comments

If a file name on one of the active subchannels is the same as *newname*, the message

DUPLICATE FILE NAME

is output and the file name is not changed. If the file named *oldname* cannot be found on any of the active subchannels, the message

oldname UNDEFINED

is output.

The *type* parameter must be a decimal number from 3 to 12. File types 3-5 require 11-word directory entries and types 6-12 require 5-word directory entires. If the file *type* is incompatible in this respect, a

PARAMETER ILLEGAL

message results. (File type numbers are described in Appendix A.)

Note: It is the users responsibility to insure that the format and structure of the file contents are compatible with its new file type.

REWIND

Purpose

To rewind a magnetic tape.

Format

:RWND[,logical unit]

where *logical unit* is the logical unit number of the desired magnetic tape (default is 8).

RPACK

Purpose

To repack the disc, eliminating purged files.

Format

:RPACK

Comments

When a :PURGE directive is issued, the directory entry for specific or implied files is purged. The :RPACK directive is used to search the directory for purged entries. If any are detected, the user file area is repacked, eliminating those files.

Note: This repacking function is automatically performed at the end of each job.

EXAMPLE:

:RPACK

scans the user directory for purged entries and repacks the disc to eliminate files associated with those entries.

RUN

Purpose

To run a user or system 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 (default is five minutes). DOS-III ignores *time* if a Time-base Generator is not present.

N, if present, tells DOS-III 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 a program executes longer than the time limit, the current job is aborted and DOS-III 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-III ultimately to execute with other HP software not having the same request codes.

EXAMPLE:

:RUN,ROUT,15

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

TM 13421
@
System responds, indicating a time-out condition.

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 source 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 valid for the current job, and can be reassigned by another JFILE directive.

When using a 21MX Assembler, up to fifteen files may be specified in the :JFILE directive so long as these files constitute *one* program having *one* END statement.

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

Example 3 in Section VI illustrates using the JFILE directive.

STORE

Purpose

To create a user file on the current user 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), binary data files (type-B), and absolute binary program files (type-X). (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[,file$_1$,file$_2$ ...]*
TYPE-S	*:STORE,S,file,logical unit [,C]*
TYPE-A	*:STORE,A,file,sectors*
TYPE-B	*:STORE,B,file,sectors*
TYPE-X	*:STORE,X,file,logical unit*

Note: Control @ should not be used in file names.

Comments

TYPE-R FILES. The directive format is

:STORE,R,file[,logical unit]

where *file* is a name consisting of five (or fewer) characters and must not duplicate another *name* already present in the user files.

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 disc if none is specified. The Job Binary Area remains as it was before the STORE,R directive.

If DOS-III 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.*

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-III Relocating Loader had relocated into executable format during the current job. A program is stored in a file of the same name. Up to 14 programs per directive are allowed. If none are specified, all programs loaded during the current job are stored. DOS-III finds these temporary programs in the user file and converts them to permanent user files by removing their "temporary" flags (see the description of the LIST,U directive).

Programs loaded during the current job but not stored as permanent files (as shown above) may be executed normally (RUN or PROG directive) and appear in the user file directory. 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 [,C]

where *file* is the name of the user file to be filled with source statements from the *logical unit* specified. *File* is a name of five or fewer characters, and must not duplicate a name already present in the user files. The source statement input must be terminated by a record containing a double colon (::) if the *C* option is omitted; or a triple colon (:::) if the *C* option is included. If the termination record is omitted, DOS-III stores the succeeding data on the disc as if it were source statements.

If DOS-III comes to an end-of-tape before finding the termination record (:: or :::), it outputs

DONE?

on the system console.

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

When DOS-III completes the STORE,S it outputs

nnnn LINES

where *nnnn* is the number of statements stored.

If the *C* parameter is included in the STORE directive, statements with a colon in column 1 are interpreted as data and transferred to the designated source file. In this case, input is terminated with a triple colon (:::). When the *C* parameter is omitted in the STORE directive, those statements with a colon in column 1 will attempt execution. The *logical unit* specified in the STORE, S directive (when the *C* parameter is used) must not be the current batch device logical unit. If it is, DOS-III outputs the message

ILLEGAL LUN

If the user is in keyboard mode, DOS-III outputs an @ and waits for a new directive. If the user is in batch mode, a batch abort occurs.

If the *C* parameter is used and the *logical unit* specified is the system console, then all input received prior to ::: is transferred to the designated source file, except OFF and ABORT directives. If either of the two are encountered during keyboard entry, they are interpreted as directives and executed. (:OFF returns control to keyboard mode without terminating the job. :ABORT aborts the current job if the directive was entered from the keyboard, or DOS-III performs a batch abort if the STORE, S directive was entered from the batch device.) Files containing :OFF and :ABORT can be created by storing from a device other than the system console or the current batch device.

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 must 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-III checks only that the file name exists and that it contains the sector specified.

EXAMPLE:

:STORE,A,ASCII,20

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

TYPE-X FILES. The directive format is

:STORE,X,file,logical unit

where *file* is the name of the user file to be filled with absolute binary programs from the device specified by *logical unit.*

When an end-of-tape is encountered, DOS-III outputs

DONE?

To continue loading tapes, place the next tape in the reader and type *NO;* otherwise, type *YES.*

SYSTEM SEARCH

Purpose

To specify a list of disc subchannels which may be searched for file names. This is the :SS condition which 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. (See "Generating and Loading DOS-III," Part 3.) Otherwise, 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, *file* 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 original 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 through the appropriate EXEC call), the system outputs a message:

SUBCHAN = n

EXAMPLE:

```
:UD
SUBCHANNEL = 1
LABEL = UNLBL
: RUN MYPRG
FILE NAME UNDEFINED        (file not on subchannel 1)
:SS                        (search all subchannels for file MYPRG)
:RUN
SUBCHANNEL = 0
(MYPRG now begins execution)
```

TOP-OF-FORM

Purpose

To issue a top-of-form command to a list device.

Format

:TOF[,logical unit]

where *logical unit* is the logical unit number of the desired list device. If *logical unit* is omitted, then logical unit 6 receives the command.

TRACKS

Purpose

To output information about the next available track on the current user disc.

Format

:TRACKS

Comments

A TRACKS, JOB, BATCH, or TYPE directive must be the first directive entered following system start.

The decimal number corresponding to the first track beyond the end of the current user area (and the number of faulty tracks encountered, if any) is output to the system console.

Faulty 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 = 0010
           @                              (End of directive processing)
```

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

```
(INPUT)    :TRACKS
(OUTPUT)   NEXT AVAIL TRACK = 0012
           BAD = 2
           @                              (End of directive processing)
```

In this example, the system reports that there are no more tracks available in the user area.

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

TYPE

Purpose

To return from batch mode to keyboard mode.

Format

:TYPE

Comments

A TYPE, JOB, BATCH or TRACKS directive must be the first directive entered following system start.

Control is returned to the system console. :TYPE may be entered through the batch device or the keyboard device; when it is entered from the keyboard, DOS-III waits until the currently 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.

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 one of the following messages from DOS-III:

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 table entry number.

If the incorrect *n* is entered, DOS-III outputs a list of all the down devices.

USER DISC CHANGE

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 new subchannel.

Comments

Discs are labeled by the INITIALIZE directive.

Each form of the UD directive has a different purpose.

EXAMPLES:

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

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

If the UD directive specifies a subchannel with an incorrect system proprietary code (see "Disc Labels" in Appendix A), DOS-III still makes the assignment, and outputs

TSB DISC or ??? DISC

If the UD directive specifies a subchannel whose system generation code does not match that of the current system disc, DOS-III still makes the assignment but outputs

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.

Notes: 1. Before executing a :DD or :DD,X to a TSB or ??? DISC, the disc should be initialized with :IN,; otherwise, bad tracks may be reported erroneously.*

2. If a disc pack is changed on a DOS-III system, the subchannel assigned to that pack must be explicitly reassigned using a :UD directive or EXEC call.

Refer to item 5 in Section VI for an example of copying a System from one subchannel to another.

SECTION III
DOS-III EXEC Calls

DOS-III EXEC calls are the line of communication between an executing program and DOS-III. 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-III. DOS-III then determines the type of request (from the parameter list) and, if it is legally specified, initiates processing of the request.

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 EXEC, 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; presents each EXEC call in detail; and concludes with a discussion of how parameters are passed to and from a program.

The EXEC calls detailed in this section are presented alphabetically, according to their function. The Request Code (RCODE) value they have in the Assembly-language calling sequence appears at the top of each page.

Note: DOS-III may include two user-created EXEC modules, loaded along with the DOS-III system EXEC modules during system generation. The purpose of the EXEC modules (called $EX36 and $EX37) and the number of parameters needed in the EXEC call are defined by the user. User EXEC module calling sequences are defined in Section XII, "User-written EXEC Modules."

ASSEMBLY LANGUAGE EXEC CALLS

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

```
EXT EXEC          (Used to link program to DOS-III)
    .
    .
    .
JSB EXEC          (Transfer control to DOS-III)
DEF *+n+1         (Defines point of return from DOS-III, 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 may occur any-
    .             where in program; may be multi-level indirect. Seven is
    .             the maximum number of allowable parameters for any
    .             EXEC call.)
DEF Pn
return point      (Continue execution of program)
    .
    .
    .
    .
    .
    .
P1 - - -          (Actual parameter values)
 .
 .
 .
Pn - - -
```

ALGOL EXEC CALLS

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 not be declared VALUE. Third, when arrays are passed as parameters, the first element of the array (not just the array name) must be passed as a type INTEGER and not by VALUE. Fourth, since ALGOL requires that the format of each procedure call be defined, a program must declare a dummy external procedure for each EXEC call requiring a different number of parameters. (These dummy procedures must be compiled as separate procedures to provide proper linkage in the Loader.)

EXAMPLE:

The program below (DXFER) reads one sector from the work area and writes the information into a different location in the work area. DXFER calls EXEC through the CODE procedure EXECX (compiled externally). EXECX is compiled in the program DSKIO, although that program name is irrelevant to the linkage between DXFER and EXECX.

MAIN PROGRAM

```
HPAL,B,L,"DXFER"
BEGIN
    INTEGER ARRAY BUFFER[1:128];
    BOOLEAN READX;
    INTEGER TRACK,SECTOR;
    FORMAT F1("SOURCE TRACK,SECTOR?"),
           F2("DESTINATION TRACK,SECTOR?");
    PROCEDURE EXECX(RD,TRK,SCTR,BFR);
      VALUE RD,TRK,SCTR;
      BOOLEAN RD;
      INTEGER TRK,SCTR,BFR;
      CODE;
  WRITE(1,F1);
  READ(1,*,TRACK,SECTOR);
  READX←TRUE;
  EXECX(READX,TRACK,SECTOR,BUFFR[1]);
  WRITE(1,F2);
  READ(1,*,TRACK,SECTOR);
  READX←FALSE:
  EXECX(READX,TRACK,SECTOR,BUFFR[1]);
END$
```

PROCEDURE

```
HPAL,P,B,L,"DSKIO"
PROCEDURE EXECX(RD,TRK,SCTR,BFR);
      VALUE RD,TRK,SCTR;
      BOOLEAN RD;
      INTEGER TRK,SCTR,BFR;
BEGIN
  PROCEDURE EXEC(IO,LU,BFR,BFSZ,TRK,SCTR);
        INTEGER IO,LU,BFR,BFSZ,TRK,SCTR;
        CODE;
  INTEGER REQCD;
  IF RD THEN REQCD←1 ELSE REQCD←2;
  EXEC(REQCD,2,BFR,128,TRK,SCTR);
END;
```

FORTRAN EXEC CALLS

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 integer values or integer variables defined elsewhere in the program.

EXAMPLE

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

} *Equivalent calling sequences*

Some EXEC call functions are generated automatically by the FORTRAN compiler or special subroutines. (Refer to "FORTRAN," in Section V and the specific EXEC calls in this section.)

BASE PAGE STORE

Purpose

To store values into base page memory locations.

Assembly Language

```
            EXT   EXEC
             .
             .
             .
            LDA   NUMB
            LDB   ADDR
            JSB   EXEC      (Transfer control to DOS-III)
            DEF   *+2       (Point of return from DOS-III)
            DEF   RCODE     (Request code)
            return point    (Continue execution)
             .
             .
             .
RCODE       DEC   -19       (Request code = -19)
NUMB        DEC   n         (n is value to be stored)
ADDR        DEF   LOC       (LOC is a base page location)
```

FORTRAN

This feature must not be invoked by a FORTRAN program.

Comments

Base Page Store stores values into base page locations normally protected by memory protect. Prior to using the calling sequence specified above, the user loads the value to be stored into the A register and the absolute address of the base page location in the B register. Base Page Store then performs a store indirect through the B register.

CAUTION: CARE MUST BE TAKEN NOT TO MODIFY SYSTEM-ESSENTIAL BASE PAGE LOCATIONS.

FILE CREATE

Purpose

To allow the user to create a user disc file under program control.

CAUTION: Because of the relationship between disc space used for the work area and disc space used for creating new files, care must be taken to create all files before issuing requests that access the disc work area (work area limits requests, disc allocation requests, work area I/O requests).

Assembly Language

```
        EXT   EXEC
          .
          .
          .
        JSB   EXEC      (Transfer control to DOS-III)
        DEF   *+6       (Point of return from DOS-III)
        DEF   RCODE     (Request code)
        DEF   RSTAT     (Return status)
        DEF   FNAME     (File name)
        DEF   TYPE      (Program type)
        DEF   DSKLN     (File length)
        return point    (Continue execution)
          .
          .
          .
RCODE   DEC   32        (Request code = 32)
RSTAT   BSS   1         (Return status from system:
                           -4  =  illegal parameter
                           -3  =  invalid file name
                           -2  =  invalid file type
                           -1  =  insufficient file space
                            0  =  normal termination
                           >0  =  duplicate file name — content is
                                  address of old directory entry)
```

```
FNAME  ASC   3,xxxxx      (5-character file name)
TYPE   OCT   nnnnnn       (Program type:
                               bit 7     = 0; permanent
                                         = 1; temporary
                               bits 5-0  = 6-14(8); program type as defined
                                           in Disc Directory "Entry Type,"
                                           Appendix A)
DSKLN  DEC   s            (Length in sectors)
```

FORTRAN

```
DIMENSION INAM(3)     (File name)
INAM(1) = xxxxxB      (First two characters)
INAM(2) = xxxxxB      (Next two characters)
INAM(3) = xxxxxB      (Last character and blank)
ITYPE   = n           (n is numeric program type)
IDSK    = s           (s is disc length in sectors)
IRCDE   = 32          (Request code)

CALL EXEC(IRCDE,IRST,INAM,ITYPE,IDSK)
```

EXAMPLE:

```
  DATA NAME/2HDA,2HIL,2HY/
        .
        .
        .
C CREATE TEMPORARY ASCII FILE OF 72 SECTORS
  CALL EXEC(32,LSTAT,NAME,213B,72)
  IF (LSTAT .NE. 0) GO TO error routine
  continue normal program path
```

FILE NAME SEARCH

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-III)
         DEF   *+4 (or 5)  (Point of return from DOS-III)
         DEF   RCODE       (Request code)
         DEF   FNAME       (File name)
         DEF   NSECT       (Number of sectors)
         DEF   IPRAM       (Optional parameter)
         return point      (Continue execution)
          .
          .
          .
RCODE    DEC   18          (Request code = 18)
FNAME    ASC   3,xxxxx     (xxxxx is the file name)
NSECT    NOP               (Number of sectors returned here; 0 if not
                           found)
IPRAM    DEC   n           n = 0   user area with wait
                           n = 1   user area without wait
                           n = 2   system area with wait
                           n = 3   system area without wait
```

FORTRAN

```
DIMENSION NAME (3)          (File name)
IPRAM     = 2               (System search, with wait)
IRCDE     = 18              (Request code)
NAME (1)  = xxxxxB          (First two characters)
NAME (2)  = xxxxxB          (Next two characters)
NAME (3)  = xxxxxB          (Last character and blank)

CALL EXEC (IRCDE, NAME, ISECT, IPRAM)
```

Comments

File searches can be performed on either the system or user area, with or without wait, according to the value of IPRAM. If IPRAM is omitted, the search is performed on the user area with wait. If the search is requested *with* wait, the A register contains the track/sector address of the file, and the B register contains the memory address of the track/sector address, upon return to the user program.

Before executing a File Name Search without wait, NSECT should be initialized to some value other than zero (for example, –1) to distinguish between "file not found" and "operation still in process" conditions on completion of the search. EXEC calls issued while the File Name Search is still in progress are queued by DOS-III and the system goes into the wait loop until the search is completed.

EXAMPLE:

```
EQUIVALENCE (AREG,IREG(1))
DATA NAME/2HFI,2H1/
      .
      .
      .
AREG = EXEC(18,NAME,ISECT,0)
IF (ISECT .NE. 0) GO TO error routine
IREG(1) = track/sector address of the file
ISECT = number of sectors in FILE1
```

Note: The FORTRAN function variable (AREG) is a copy of the A-register or the A- and B-registers.

FILE PURGE

Purpose

To allow the user to purge a user disc file directory entry or to purge all temporary file entries.

Assembly Language

```
            EXT   EXEC
              .
              .
              .
            JSB   EXEC          (Transfer control to DOS-III)
            DEF   *+3 (or 4)    (Point of return from DOS-III)
            DEF   RCODE         (Request code)
            DEF   RSTAT         (Return status)
            DEF   FNAME         (Optional file name)
            return point        (Continue execution)
              .
              .
              .
RCODE       DEC   33            (Request code = 33)
RSTAT       BSS   1             (Return status from system:
                                      -4  =  illegal parameter
                                      -3  =  invalid file name
                                      -1  =  undefined file name
                                       0  =  normal termination

FNAME       ASC   3,xxxxx       (5-character file name)
```

FORTRAN

```
DIMENSION INAME(3)          (File name)
INAME(1)  =   xxxxxB        (First two characters)
INAME(2)  =   xxxxxB        (Next two characters)
INAME(3)  =   xxxxxB        (Last character and blank)
IRCDE     = 33              (Request code = 33)

CALL EXEC(IRCDE,IRST,INAME)
```

Comments

If the file name parameter is omitted, all temporary file entries are deleted from the directory.

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-III)
        DEF  *+7 (or 8)    (Point of return from DOS-III)
        DEF  RCODE         (Request code)
        DEF  CONWD         (Control information)
        DEF  BUFFR         (Buffer location)
        DEF  BUFFL         (Buffer length)
        DEF  FNAME         (File name)
        DEF  RSECT         (Relative sector within file)
        DEF  IPRAM         (Area which could have been legally transferred
                            if an overflow occurred-optional parameter)
        return point       (Continue execution)
         .
         .
RCODE   DEC  14 or 15      (Request code:  14 = read, 15 = write)
CONWD   OCT  conwd         (See Comments, I/O 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)
IPRAM   NOP                (Optional parameter; see Comments)
```

FORTRAN

```
DIMENSION NAME (3), IBUF(10)
NAME(1) = xxxxxB          (First two characters of file name)
NAME(2) = xxxxxB          (Second two characters)
NAME(3) = xxxxxB          (Last character and blank)
ICRDE   = 14 (or 15)      (Request code)
ICON    = conwd           (See comments)
IRSCT   = 0               (Relative sector number)

CALL EXEC (IRCDE, ICON, IBUF, 10, NAME, IRSCT, IPRAM)
                          or
CALL EXEC (IRCDE, ICON, IBUF, 10, NAME, IRSCT)
```

Comments

See the Comments under I/O READ/WRITE EXEC call (RCODE = 1 or 2) for a description of the *conwd* fields needed in the above calling sequences.

To read or write on the *m*th sector of a file, set RSECT = *m*−1. To determine the size of a file, use the FILE NAME SEARCH EXEC call (RCODE = 18).

Data files to be written (or read) should be created with a STORE directive before executing the EXEC call.

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

If the DOS-III installation is likely to have more than one user disc, the program should use the USER DISC CHANGE EXEC call (RCODE = 23) 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.

This call provides an optional parameter, IPRAM, to provide the user with information concerning a file read/write overflow (where the buffer length exceeds the sector contents). If IPRAM is omitted, an overflow causes an IT error. If IPRAM is included and an overflow occurs, control is returned to the user program with IPRAM set equal to the number of words (+) or characters (−) (as defined by BUFFL) that could legally have been transferred. If an overflow occurs, no disc transfer takes place, whether IPRAM is included or not. If IPRAM is included and no overflow occurs, the value of the parameter is set to zero.

EXAMPLE:

```
DATA  NAME /2HFI,2HLE,2H1/
DIMENSION IBUF(128)
      .
      .
      .
CALL EXEC(14,3,IBUF,128,NAME,0)
```

Read the first sector of FILE1.

FILE RENAME

Purpose

To allow the user to change a file name (and optionally, its type) under program control.

Assembly Language

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC          (Transfer control to DOS-III)
            DEF   *+5 (or 6)    (Point of return from DOS-III)
            DEF   RCODE         (Request code)
            DEF   RSTAT         (Return status)
            DEF   ONAME         (Old file name)
            DEF   NNAME         (New file name)
            DEF   NTYPE         (Optional new file type)
            return point        (Continue execution)
             .
             .
             .
RCODE       DEC   34            (Request code = 34)
RSTAT       BSS   1             (Return status from system:
                                    -4  =  illegal parameter
                                    -3  =  invalid old or new file name
                                    -2  =  invalid old or new file type
                                    -1  =  undefined old or new file name
                                     0  =  normal termination
                                    >0  =  duplicate new file name; content
                                           is address of duplicate directory
                                           entry)
ONAME       ASC   3,xxxxx       (5-character file name to be changed)
NNAME       ASC   3,xxxxx       (5-character new file name)
NTYPE       OCT   nnnnnn        (New program type:
                                    bit 7     =  0; permanent
                                              =  1; temporary
                                    bits 5-0  =  6-14(8); program type as defined
                                                 in Disc Directory "Entry Type,"
                                                 Appendix A)
```

FORTRAN

```
DIMENSION INAMO(3), INAMN(3)    (Old file name, new file name)
INAMO(1)  =    xxxxxB           (First two characters)
INAMO(2)  =    xxxxxB           (Next two characters)
INAMO(3)  =    xxxxxB           (Last character and blank)
INAMN(1)  =    xxxxxB           (First two characters)
INAMN(2)  =    xxxxxB           (Next two characters)
INAMN(3)  =    xxxxxB           (Last character and blank)
IRCDE = 34                      (Request code = 34)
ITYPE = n                       (File type)

CALL EXEC(IRCDE,IRST,INAMO,INAMN,ITYPE)
```

Comments

The specified old name may match the new name — no error message is returned, the new program type (if specified) will be changed.

I/O CONTROL

Purpose

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

Assembly Language

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC          (Transfer control to DOS-III)
            DEF   *+3 (or 4, or 5)  (Point of return from DOS-III)
            DEF   RCODE         (Request code)
            DEF   CONWD         (Control information)
            DEF   PRAM1         (First optional parameter)
            DEF   PRAM2         (Second optional parameter)
            return point        (continue execution)
             .
             .
             .
RCODE       DEC   3             (Request code = 3)
CONWD       OCT   conwd         (See Comments)
PRAM1       DEC   n             (Optional value parameter; see "Comments")
PRAM2       BSS   m             (Optional buffer address)
```

FORTRAN

Use the specific FORTRAN auxiliary I/O statements (see Comments) or an EXEC calling sequence.

```
DIMENSION IPRM2(m)          (Define buffer of m words)
IRCDE = 3                   (Request code)
ICNWD = conwd               (See Comments)
IPRAM = n                   (Optional; see Comments)
CALL EXEC (IRCDE,ICNWD,IPRAM)

                or

CALL EXEC (IRCDE,ICNWD)

                or

CALL EXEC (IRCDE,ICNWD,IPRAM,IPRM2)
```

Comments

CONWD

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

	0	0	W	FUNCTION CODE (see below)						LOGICAL UNIT NUMBER					
BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

WAIT FIELD (W)

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

If W = 0, DOS-III waits until the control request is complete before returning.

FUNCTION CODE FIELD

Function codes are defined programatically within the various I/O drivers. Thus the following list of standard function codes is general in nature. Detailed information on specific peripheral-associated function codes is available in the *DOS-III Standard Drivers Reference Manual* (24307-90073).

Function Code (Octal)	Action
000	Clear the device (all drivers)
001	Write end-of-file (magnetic tape), select hopper (optical mark reader)
002	Backspace one record (magnetic tape)
003	Space forward one record (magnetic tape)
004	Rewind (magnetic tape), bell request (optical mark reader)
005	Rewind standby (magnetic tape)
006	Dynamic status (all drivers)
007	Set end-of-paper tape (paper tape punch)
010	Generate paper tape leader (paper tape punch)
011	List output line spacing (line printers) (PARM1 or IPRAM required)
012	Write file gap (magnetic tape)
013	Space forward one file (magnetic tape)
014	Backspace one file (magnetic tape)
017	Extended function code present (card reader punch)

For function code values 000 through 077_8, no DMA is assigned. For function code values 100 through 177_8, DMA is assigned if required by the I/O driver.

LOGICAL UNIT FIELD

This field specifies the logical unit number of the device which is to receive the control request.

OPTIONAL PARAMETERS

Specification of $Parameter_1$ (PRAM1 or IPRAM) or $Parameter_2$ (PRAM2 or IPRM2) depends on the contents of the function code field in the control word. Function code 11_8 requires $Parameter_1$. This parameter designates the number of lines to be spaced on the specified logical unit. A negative value specifies a page eject on a line printer or the number of lines to be spaced on the System Console. For details on line printer formatting, refer to Section IV in the *DOS-III Standard Drivers Reference Manual* (24307-90073). When $Parameter_1$ is specified, its value is passed to EQT10 prior to entering the driver. If $Parameter_2$ is specified, $Parameter_1$ must be specified. The value of $Parameter_2$ is passed to the driver via EQT11.

Compiler Considerations

Within FORTRAN and ALGOL programs, various control operations for magnetic tape may be performed by the following auxiliary I/O statements:

BACKSPACE
ENDFILE
REWIND

Refer to the appropriate compiler manual for a detailed description of these statements.

EXAMPLES:

```
C CLEAR I/O DEVICES 1 TO N
  DO 10 LU=1,N
  10 CALL EXEC (3,LU)

C SPACE 5 LINES ON THE LINE PRINTER
  CALL EXEC (3,1106B,-5)

C SPACE FORWARD ONE FILE MARK
  CALL EXEC (3,1310B)

C FOR DATA COMMUNICATION — SET TERMINAL OPTION ENABLE AUTO L.F.
  CALL EXEC (3,4000B+LU,1)
```

I/O READ/WRITE

Purpose

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

Assembly Language

```
                EXT   EXEC
                 .
                 .
                 .
                JSB   EXEC        (Transfer control to DOS-III)
                DEF   *+5 (or 7)  (Point of return from DOS-III; 7 is for disc request)
                DEF   RCODE       (Request code)
                DEF   CONWD       (Control information)
                DEF   BUFFR       (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)    (Request code: 1 = read, 2 = write)
CONWD           OCT   conwd       (conwd is described in comments)
BUFFR           BSS   n           (Buffer of n words)
BUFFL           DEC   n (or -2n)  (≤n; words (+) or characters (-))
DTRAK           DEC   f           (Work area track number, decimal)
DSECT           DEC   g           (Work area sector number, decimal)
```

Note: Single I/O transfers within the DOS-III environment cannot exceed 16K words.

RCODE = 1; RCODE = 2

FORTRAN

DIMENSION IBUF (n)	*(Define buffer of n words)*
IRCDE = 1 (or 2)	*(Request code)*
ICON = conwd	*(See Comments)*
IBUFL = n (or -2n)	*(Buffer length in words (+) or characters (-1))*
ITRAK = 150	*(Disc track number)*
ISECT = 0	*(Disc sector number)*

CALL EXEC (IRCDE, ICON, IBUF, IBUFL, ITRAK, ISECT) *for disc transfers*
CALL EXEC (IRCDE, ICON, IBUF, IBUFL) *for non-disc transfers.*

Comments

CONWD

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

	0	0	W	J		A		K	V	M	LOGICAL UNIT #					
BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

FIELD	FUNCTION
W	If 1, tells DOS-III to return to the calling program after starting the I/O transfer. If W = 0, DOS-III waits until the transfer is complete before returning.
J	If 1, and logical unit number is 2 or 3 (disc), a backward track increment will be performed (for example, JBIN read/write). (This field is applicable only to RCODE = 1 or RCODE = 2.)
A	When transferring variable length binary records (M = V = 1), A = 1 indicates absolute binary format.
K	1) When used with console keyboard input, if K=0 "no printing" is specified. If K=1 printing the input as received is specified. 2) When used with disc write requests, if K=0 execute cyclic check after disc write. If K=1 eliminate cyclic check after disc write.
V	1) When reading variable length records from punched tape devices in binary format (M = 1), if V = 0 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 read in. 2) When outputting ASCII records to a list device (M = 0), if V = 0 the first character in the buffer is interpreted as a carriage control character (see Section IV). If V = 1, single spacing occurs, and the entire buffer (including the first character is output to the list device.
M	Determines the mode of data transfer. If M = 0, transfer is in ASCII character format, and if M = 1, binary format.

"Waiting and No Waiting"

If the program requests the "waiting" option in the *conwd* (W = 0), DOS-III will return the transmission log in the B register upon completion. (The transmission log is a positive number, representing the number of words or characters transmitted, depending upon which was originally requested.)

If the program requests the "no waiting" option in the *conwd* (W = 1), it can check for the completion of the I/O operation with the I/O STATUS EXEC call (RCODE = 13). When the operation is complete (STATS $\geqslant$ 0), the transmission log can be retrieved from the TLOG parameter.

Notes: When using "no waiting" I/O and loading program segments:

1. *Under :RUN, DOS-III waits for all I/O to complete before loading the segment.*
2. *Under :PROG, DOS-III does not wait.*

If a read or write is issued to a disc address that does not lie in the Work Area, the message *IT nnnnn* is output and the program is terminated.

Compiler Considerations

Within FORTRAN and ALGOL programs, I/O transfers to standard devices are programmed by the READ and WRITE statements.

I/O transfers to the Work Area and the disc may be done through the BINRY library routine. 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 number, track number, sector number, and offset in words within the sector. (If the offset equals 0, the transfer begins on the sector boundary. If the offset equals N, then N words of the sector are skipped before starting the transfer.) BINRY has two entry points, BREAD and BWRIT, for read and write operations respectively. An example below gives the calling procedure.

```
DIMENSION IBUF(10), BUF(20)
LUN = 2
ITRK = 120
ISECT = 36
IOFF = 0
CALL BREAD (BUF, 40, LUN, ITRK, ISECT, IOFF)
                    or
CALL BWRIT (IBUF, 10, LUN, ITRK, ISECT, IOFF)
```

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-III)
            DEF  *+4 (or 5)    (Point of return from DOS-III)
            DEF  RCODE         (Request code)
            DEF  LUN           (Logical unit)
            DEF  STATS         (Status returned)
            DEF  TLOG          (Transmission log returned, optional)
            return point       (Continue execution)
             .
             .
             .
RCODE       DEC  13            (Request code = 13)
LUN         DEC  n             (Logical unit number)
STATS       NOP                (Status returned here)
TLOG        NOP                (Transmission log returned here)
```

FORTRAN

```
IRCDE = 13                     (Request code)
LUN = n                        (n is decimal logical unit number)
CALL EXEC (IRCDE, LUN, 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 number. The transmission log in the B register and in TLOG contains the amount of information which was last transferred (a positive number of words or characters, depending on which was requested by the call initiating that transfer).

MEMORY MANAGEMENT (BUFFER ALLOCATION)

Purpose

To allocate buffer space within an area reserved under a block name identifier (see "Memory Management (Initialize)") or from unassigned available memory.

Assembly Language

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC      (Transfer control to DOS-III)
            DEF   *+6 (or 7)(Point of return from DOS-III)
            DEF   RCODE     (Request code)
            DEF   RSTAT     (Return status)
            DEF   LENG      (Desired buffer length)
            DEF   SADR      (Buffer starting address is returned here)
            DEF   ID        (Buffer identifier is returned here)
            DEF   BID       (Optional block name identifier)
            return point    (Continue execution)
             .
             .
             .
RCODE       DEC   38        (Request code = 38)
RSTAT       BSS   1         (Return status from system:
                                -4  =  illegal parameter
                                -3  =  BID not present
                                -1  =  no memory available
                                 0  =  normal return
                                >0  =  requested amount not available;
                                       contents is actual number of
                                       words available)
LENG        DEC   n         (Buffer length in words)
SADR        BSS   1         (Actual starting address from system)
ID          BSS   1         (Buffer identifier from system 1 ≤ ID ≤ 1023)
BID         ASC   2,xxxx    (4-character unique memory management block
                             name identifier)
```

Comments

If a block name identifier is specified, the buffer will be allocated space within the area reserved for that identifier. If the block name identifier is omitted, space is allocated from unassigned available memory.

MEMORY MANAGEMENT (BUFFER RELEASE)

Purpose

To release reserved buffer space.

Assembly Language

```
            EXT   EXEC
              .
              .
              .
            JSB   EXEC      (Transfer control to DOS-III)
            DEF   *+4       (Point of return from DOS-III)
            DEF   RCODE     (Request code)
            DEF   RSTAT     (Return status)
            DEF   ID        (buffer identifier)
            return point    (Continue execution)
              .
              .
              .
RCODE       DEC   41        (Request code = 41)
RSTAT       BSS   1         (Return status from system:
                                -4  =  illegal parameter
                                -1  =  illegal ID
                                 0  =  normal return
ID          DEC   n         (Buffer identifier 1 ≤ ID ≤ 1023)
```

Comments

This request releases space allocated to buffers. If the specified buffer resides within the area reserved under a block name identifier, the logical address space remains reserved. Otherwise, the released space is returned to the system.

MEMORY MANAGEMENT (INITIALIZE)

Purpose

To reserve a block of memory under a block name identifier specified by the user.

Assembly Language

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC      (Transfer control to DOS-III)
            DEF   *+6 (or 7)(Point of return from DOS-III)
            DEF   RCODE     (Request code)
            DEF   RSTAT     (Return status)
            DEF   LENG      (Desired block length)
            DEF   SADR      (Block starting address is returned here)
            DEF   BID       (Block name identifier)
            DEF   LADR      (Optional starting address parameter)
            return point    (Continue execution)
             .
             .
             .
RCODE       DEC   35        (Request code = 35)
RSTAT       BSS   1         (Return status from system:
                               -4  =  illegal parameter
                               -2  =  another block name identifier
                                      assigned to area specified by LADR
                               -1  =  no memory available
                                0  =  normal termination
                               >0  =  space requested not available;
                                      content is number of words
                                      available)
LENG        DEC   n         (Block length in words)
SADR        BSS   1         (Actual starting address of block—from system)
BID         ASC   2,xxxx    (4-character memory management block name
                             identifier)
LADR        OCT   n         (Requested starting address—0 = don't care)
```

Note: A non-zero LADR value must be an address between ending program address and last word of available memory.

Comments

This request reserves a block of memory under the block name identifier (BID) specified by the user. Subsequent user requests for allocation of buffer space within this area may be made. If the memory management initialize request (RCODE=35) is not included in a user program prior to buffer allocation requests (RCODE=38) for buffers within the specified BID, an error return condition results. If LADR is specified and is non-zero, the value must be an address between the end of program address and the last word of available memory.

MEMORY MANAGEMENT (STATUS REQUEST)

Purpose

To determine the number of words reserved under a block name identifier or the number of unallocated words remaining.

Assembly Language

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC      (Transfer control to DOS-III)
            DEF   (+3 (or 4) (Point of return from DOS-III)
            DEF   RCODE     (Request code)
            DEF   LENG      (Word count from system)
            DEF   BID       (Optional block name identifier)
            return point    (Continue execution)
             .
             .
             .
RCODE       DEC   36        (Request code = 36)
LENG        BSS   1         (Number of words allocated to BID or number of
                             available words if BID is not present. If BID
                             parameter is specified but not found, a -3 value
                             is returned)
BID         ASC   2,xxxx    (Unique memory management block name identi-
                             fier)
```

Comments

When the BID parameter is specified, this request returns the number of words reserved under a user-specified block name identifier (BID). If the BID parameter is specified but not found, a –3 value is returned. If the BID parameter is not specified, the request returns the number of unallocated words remaining in the system.

MEMORY PROTECT CONTROL

Purpose

To enable or disable the memory protect option from a user program.

CAUTION: THE SYSTEM IS NOT PROTECTED WHEN MEMORY PROTECT IS IS DISABLED.

Assembly Language

```
        EXT   EXEC
          .
          .
          .
        JSB   EXEC      (Transfer control to DOS-III)
        DEF   *+3       (Point of return from DOS-III)
        DEF   RCODE     (Request code)
        DEF   MPTK      (Define the memory protect parameter flag)
        return point    (Continue execution)
          .
          .
          .
RCODE   DEC 30          (Request code = 30)
MPTK    DEC n           (If n = 0, memory protect is activated, and
                         is activated following any interrupt
                         completion. If n ≠ 0, then memory protect
                         is deactivated and remains off after
                         interrupt completion)
```

FORTRAN

```
IRCDE = 30
MPTK = 0 (or 1)
CALL EXEC (IRCDE,MPTK)
```

Comments

Any program termination, either normal or aborted, enables memory protect. Program segments can make memory protect EXEC calls to turn memory protect on or off, but calling and exiting from segments has no effect on memory protect settings.

PROGRAM COMPLETION

Purpose

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

Note: Every program must terminate and return to DOS-III using this EXEC call, whether the EXEC call is explicitly coded or indirectly generated by a compiler.

Assembly Language

```
        EXT  EXEC
         .
         .
        JSB  EXEC          (Transfer control to DOS-III)
        DEF  *+2           (Define end of parameter list)
        DEF  RCODE         (Request code)
         .
         .
         .
RCODE   DEC  6             (Request code = 6)
```

FORTRAN

```
IRCDE = 6
CALL EXEC (IRCDE)
```

Compiler Considerations

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

PROGRAM LOAD

Purpose

To load a main program from the disc into main memory and transfer control to its entry point. Follows the :SS condition.

Assembly Language

```
          EXT  EXEC
           .
           .
          JSB  EXEC          (Transfer control to DOS-III)
          DEF  *+3 (to 8)    (Determine number of parameters)
          DEF  RCODE         (Request code)
          DEF  PNAME         (Program name)
          DEF  PRAM1         (First optional parameter)
           .
           .
           .
          DEF  PRAM5         (Fifth optional parameter)

RCODE     DEC  10
PNAME     ASC  3,xxxxx       (Program name)
PRAM1     ---                (Up to 5 words of parameter information
  .                           passed to the program. See "Parameter
  .                           Processing" at the end of this section.)
  .
PRAM5
```

FORTRAN

```
DIMENSION NAME(3)            (Program name)
IRCDE = 10
NAME(1) = xxxxxB             (First two characters)
NAME(2) = xxxxxB             (Next two characters)
NAME(3) = xxxxxB             (Last character and blank)
CALL EXEC (IRCDE,NAME[,p1...])
```

Comments

During main program loading, the system interrogates a system flag called AEPF (location 135_8). This flag is normally zero unless specifically set by a user program. If AEPF is not zero, the contents of AEPF are treated as an alternate entry point address. The system transfers control to the alternate entry point by performing a JMP AEPF,I (jump indirect). AEPF is then cleared. If AEPF = 0, control transfers to the program main entry point.

The Assembly language user can alter the contents of AEPF (and any other base page location) by using the BASE PAGE STORE EXEC call (RCODE = -19).

PROGRAM SUSPENSION

Purpose

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

Assembly Language

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

FORTRAN

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

Comments

DOS-III prints a message on the system console when it processes the PROGRAM SUSPENSION EXEC call:

name SUSP

When the operator restarts the program with a :GO, up to five parameters may be passed to the suspended program. (See "Parameter Processing" at the end of this section.)

Compiler Considerations

The FORTRAN and ALGOL compilers automatically generate a PROGRAM SUSPENSION EXEC call when they compile a PAUSE statement.

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 V, "DOS-III Subsystems," for information on segmented programs.) Follows the :SS condition.

Assembly Language

```
            EXT  EXEC
             .
             .
             .
            JSB  EXEC          (Transfer control to DOS-III)
            DEF  *+3 (to 8)    (Determine number of parameters)
            DEF  RCODE         (Request code)
            DEF  SNAME         (Segment name)
            DEF  PRAM1         (First optional parameter)
             .
             .
             .
            DEF  PRAM5         (Fifth optional parameter)
             .
             .
             .
RCODE       DEC  8             (Request code = 8)
SNAME       ASC  3,xxxxx       (xxxxx is the segment name)
PRAM1       ---                (Up to 5 words of parameter information
PRAM5       ---                passed to the segment. See "Parameter
                               Processing" at the end of this section.)
```

FORTRAN

```
DIMENSION NAME (3)             (Segment name)
IRCDE = 8
NAME (1) = xxxxxB              (First two characters)
NAME (2) = xxxxxB              (Next two characters)
NAME (3) = xxxxxB              (Last character and blank)
CALL EXEC (IRCDE, NAME [,p1...] )
```

Comments

In the FORTRAN or ALGOL calling sequence, the user must convert the name of the segment from ASCII to octal and store it in the NAME array, two characters per word. The RTE/DOS FORTRAN IV Compiler, however, can convert this automatically through Hollerith constants.

During program segment loading, the system interrogates a system flag called AEPF (location 135_8). This flag is normally zero unless specifically set by a user program. If AEPF = 0, control transfers to the program segment main entry point. If AEPF is not zero, the contents of AEPF are treated as an alternate entry point address. The system transfers control to the alternate entry point by performing a JMP AEPF,I (jump indirect). AEPF is then cleared. (The Assembly language user can alter the contents of AEPF (and any other base page location) by using the BASE PAGE STORE EXEC call (RCODE = -19).)

See "Segmented Programs," in Section V, for a description of segmented programs.

SEGMENT RETURN

Purpose

To return control from a segment to the main program at the instruction immediately following the program segment load call. (This provides a subroutine-like return from a segment to a main program.)

Assembly Language

```
            EXT   EXEC
              .
              .
              .
            JSB   EXEC        (Transfer control to DOS-III)
            DEF   *+2 (to 7)  (Point of return from DOS-III)
            DEF   RCODE       (Define the request code)
            DEF   PRAM1       (Define the first parameter)
              .
              .
              .
            DEF   PRAM5       (Define the fifth optional parameter)
              .
              .
              .
RCODE       DEC 29            (Request code = 29)
PRAM1       ...               (Up to five words of parameter information
  .                           are passed from the segment to the main
  .                           program. See "Parameter Processing" at
                              the end of this section)
PRAM5       ...
```

FORTRAN

```
IRCDE = 29
CALL EXEC (IRCDE [,P1, . . . ,P5] )
```

TIME REQUEST

Purpose

To request the current time.

Assembly Language

```
            EXT  EXEC
              .
              .
              .
            JSB  EXEC          (Transfer control to DOS-III)
            DEF  *+3           (Point of return from DOS-III)
            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

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

Comments

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

ARRAY	or ITIME (1)	=	Tenth of seconds
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 = 0)

If DOS-III does not contain Time-base Generator, all values in the time array are set to zero.

WORK AREA LIMITS

Purpose

To ascertain the first and last tracks of the Work Area on the system or current user disc and the number of sectors per track.

Assembly Language

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC          (Transfer control to DOS-III)
            DEF   *+5 (or 6)    (Point of return from DOS-III)
            DEF   RCODE         (Request code)
            DEF   FTRAK         (First track)
            DEF   LTRAK         (Last track)
            DEF   SIZE          (Number of sectors/track)
            DEF   DISC          (Optional parameter — see Comments)
            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)
DISC        DEC   n             (n = 0 for system disc; n ≠ 0 for current user disc)
```

FORTRAN

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

Comments

This call returns the limits of the Work Area, which is that area of the system or user disc which programs use for temporary storage with the I/O READ/WRITE EXEC call (RCODE = 1 or 2). If the DISC parameter is omitted from the calling sequence, or if DISC = 0, the system disc information is returned. If DISC ≠ 0, user disc information is returned.

WORK AREA STATUS

Purpose

To ascertain whether a specified number of consecutive operable tracks exist in the Work Area of the system disc.

Assembly Language

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC      (Transfer control to DOS-III)
            DEF   *+5       (Point of return from DOS-III)
            DEF   RCODE     (Request code)
            DEF   NTRAK     (Number of tracks desired)
            DEF   TRACK     (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, 0 if n tracks
                             not available)
```

FORTRAN

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

Comments

This call is used with the WORK AREA LIMITS EXEC call (RCODE = 17) to establish the nature of the Work Area. The READ/WRITE EXEC call (RCODE = 1 or 2) then transmits information to and from this area, using the track numbers determined by this call. DOS-III handles track switching automatically.

DOS-III checks whether there are *n* consecutive tracks starting at the track specified. If *n* tracks are available, DOS-III returns the starting track number to the program. If DOS-III does not locate *n* consecutive tracks, it returns 0 in STRAK or ISTRK.

USER DISC CHANGE

Purpose

To change the subchannel assignment for the user disc.

Assembly Language

```
            EXT   EXEC
              .
              .
              .
            JSB   EXEC          (Transfer control to DOS-III)
            DEF   *+3 (or 4)    (Point of return from DOS-III)
            DEF   RCODE         (Request code)
            DEF   LABEL         (Disc label)
            DEF   SUBCH         (Disc subchannel; optional)
            return point        (Continue execution)
              .
              .
              .
RCODE       DEC   23            (Request code = 23)
LABEL       ASC   3,xxxxxx      (Label = xxxxxx)
SUBCH       DEC   (0 to 7)
```

FORTRAN

```
DIMENSION LABEL (3)          (New label)
IRCDE    = 23
LABEL (1) = xxxxxB           (First two characters)
LABEL (2) = xxxxxB           (Next two characters)
LABEL (3) = xxxxxB           (Last two characters)
ICHNL     = M                (0 through 7)


CALL EXEC (IRCDE, LABEL, ICHNL)
                or
CALL EXEC (IRCDE, LABEL)
```

Comments

If both the label and subchannel are specified, DOS-III checks whether the subchannel has that label. If it does, the assignment is made and DOS-III returns. If not, DOS-III outputs

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 input

:GO

to return 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.

If only a label is specified, DOS-III searches for the label, starting with the highest subchannel. If DOS-III finds the label, it makes the assignment. If DOS-III cannot find the label, it suspends the program and outputs

DISC NOT ON SYS
UD nnnnn
xxxxx SUSP

The operator can then abort the program or load a properly labeled disc then input

:GO

to return to the *beginning* of the EXEC call.

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

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 then input

:GO

to return to the *beginning* of the EXEC call.

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

DISC NOT ON SYS
UD nnnnn
xxxxx SUSP

The operator can then abort the program or load an unlabeled disc then input

:GO

to return to the beginning of the EXEC call.

Notes: 1. *If the EXEC call specifies a subchannel with an incorrect system proprietary code (see Appendix A), DOS-III still makes the assignment but outputs*
TSB DISC or ??? DISC

2. *If the EXEC call specifies a subchannel whose system generation code (see Section VII) does not match that of the system disc, DOS-III still makes the assignment, but outputs*
DISC GEN CODE nnnn NOT SYS GEN CODE mmmm ERR POS

3. *The changes made by this EXEC call are only temporary, and will be reset at the end of each job to the user subchannel specified during system generation.*

4. *If the specified subchannel is not active (physically present), DOS-III suspends the programaand outputs*
I/O ERR NR USER DISC
or
I/O ERR PE USER DISC
UD nnnnn *(nnnnn = address of EXEC call)*
xxxxx SUSP

PARAMETER PROCESSING

Certain user programs require parameters for their execution. DOS-III allows passing of parameters in the following environments:

(1) from a main program to a main program

(2) from a main program to a segment

(3) from a segment to a main program

(4) from a user to a suspended program

Parameter transferral from program to program (1-3) is handled programmatically by specifying parameters in an EXEC calling sequence. Parameter transferral from a user directly to a program (4) is handled by passing parameters back to the suspended program through the GO directive.

All the programs receiving parameters retrieve them in the same way. The parameters to be passed (if any) are located in the base page parameter buffer RONBF (see Appendix A). In the Assembly language environment, the B register contains the address of the parameter buffer. In the FORTRAN/ ALGOL environment, a library routine (RMPAR) is provided to transfer parameters to a user-defined buffer. (This call must be the first statement executed upon entry.)

ASSEMBLY LANGUAGE EXAMPLE

```
        EXT   EXEC
          .
          .
          .
        JSB   EXEC          (Call EXEC to suspend program)
        DEF   *+2
        DEF   RCODE
        LDA   B,I           (Get parameter from GO directive)
        SZA,RSS
        JMP   NOPAR
          .
          .
          .
RCODE   DEC 7
B       EQU 1
```

FORTRAN EXAMPLE

```
DIMENSION I(5)          (Define user parameter buffer)
CALL EXEC (7)           (Suspend program)
CALL RMPAR (I)          (Get parameters from :GO)
```

SECTION IV
Input/Output

In DOS-III, 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 main memory locations for interrupt processing. By means of several user-defined I/O tables, multiple-device drivers, and program EXEC calls, DOS-III relieves the programmer of most I/O problems.

> *Note: Refer to Section XIV, "Privileged Mode," for a discussion of privileged mode processing.*

USER PROGRAM I/O

The user program requests I/O by means of an EXEC call (see Section III) which specifies the logical unit, control information, type of operation, buffer location and buffer length.

> *Note: Within the DOS-III environment, it is possible to transfer up to 16K words in a single operation.*

All references to I/O devices are made through logical unit numbers. This relieves the programmer of the burden of knowing which physical device or which I/O channel is actually going to perform the I/O transfer.

DOS-III has the following standard function assignments for logical unit numbers:

	Logical Unit Number	Function
Restored after each :JOB.	1	System console
	2	System mass storage
	3	User mass storage
	4	Standard punch device
	5	Standard input device
	6	Standard list device
	7	Unassigned
	8	Recommended for magnetic tape
	9	Can be assigned to any device
	10	by user
	:	
	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 established at system generation (see Section X), whereas 10 through 63 are restored only on a start-up from the disc.

SYSTEM I/O PROCESSING

System I/O processing is controlled by three I/O tables:

1) Equipment Table (EQT) — which records all devices, I/O channels, driver entry points, DMA requirements, and disc location (if disc-resident).

2) Logical Unit Table (LUT) — which assigns an equipment table number to each of its entries, thus allowing the programmer to reference changeable logical units instead of fixed physical units.

3) Interrupt Table (INT) — which relates each I/O channel to its corresponding equipment table entry.

For a detailed description of these tables see Appendix A.

When the system recognizes an EXEC call that performs I/O, the request is sent to the I/O supervisor EXEC module ($EX18). $EX18 determines if the driver for the requested device is main-memory resident; if not, the driver is loaded into main memory from the disc. Once the driver is in main-memory, the addresses of its EQT entries are placed in the base page communication area and control is transferred to the driver's initiation section. After the driver initiates the I/O operation, it returns to $EX18. If the I/O was requested "without wait", DOS-III immediately returns control to the user program; if the I/O was requested "with wait", DOS-III waits until the I/O transfer is complete before returning to the user program.

Once a driver has been initiated, interrupts from the device are channeled through a central interrupt processing routine ($CIC). (All interrupt locations in main memory contain a JSB $CIC.) $CIC determines which device interrupted, resets the addresses of the EQT entries into the base page communication area (if necessary), and transfers control to the driver's continuation section. The driver either continues or completes the I/O operation, and control is then returned to the executing user program.

INPUT/OUTPUT DRIVERS

The I/O driver routines, either main-memory or disc-resident, handle the actual transfers of information between the computer and external devices. They are responsible for initiating and continuing operations on all devices of equivalent type. When a transfer is initiated, DOS-III places the EQT entry addressed into the base page communication area and executes a subroutine jump 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-III. When an interrupt occurs on the channel, indicating continuation or completion of the transfer, DOS-III again transfers control to the driver. DOS-III requires only three drivers: the Moving-Head Disc Driver (DVR31), the System Console Driver (DVR00, DVR05, or DVR26), and the Paper Tape Reader Driver (DVR01).

The following standard drivers are fully compatible with DOS-III:

Driver Number	Description	Part Number	DMA?
DVR00	System Console Driver (TTY)	20985-60001	No
DVR01	Paper Tape Reader Driver	20987-60001	No
DVR02	Paper Tape Punch Driver	20989-60001	No
DVR05	System Console Driver (TTY)	24157-60001	No
DVR10	Digital Plotter Complete Driver	07210-16001	No
DVR10	Digital Plotter Minimum Driver	07210-16002	No
DVR11	Card Reader Driver	24272-60001	Yes
DVR12	Line Printer Driver	24307-16011	No
DVR15	Optical Mark Reader Driver	24307-16017	No
DVR23	Magnetic Tape Unit Driver	13024-60001	Yes
DVR26	Terminal Printer Driver	24307-16018	No
DVR31	Disc Driver	24156-60001	Yes
DVR33	Writable Control Store Driver	24278-60001	Yes
DVR34	Card Reader Punch Driver	12989-16002	No
DVR67	Hardwired Serial Interface Driver	24341-16001	No

The driver name consists of the letters "DVR" prefixed to the equipment type code. In addition, the programmer can write drivers for special devices, following the guidelines in Section XIII, "Planning I/O Drivers." The driver is only responsible for updating the status field in the EQT entry; DOS-III handles the availability field.

SPECIAL DRIVER CONSIDERATIONS

Since the various peripheral devices are unique, the drivers designed for use with these devices are also unique. This diversification creates the need for special considerations when planning input/output operations. The *DOS-III Standard Drivers Reference Manual* (24307-90073) deals at length with such subjects as creating plotter drawings (Section II), line printer formatting (Section IV), magnetic tape error recovery (Section VI), and using the writable control store driver (Section VIII).

SECTION V
DOS-III Subsystems

This section describes conventions for using the following DOS-III subsystems:

- ALGOL Compiler
- Assembler
- FORTRAN and FORTRAN IV Compilers
- Relocating Loader
- Relocatable libraries, including the DEBUG subroutine

and concludes with a discussion of program segmentation.

SOURCE PROGRAM FILES

Using the DOS-III STORE,S and EDIT directives, the operator creates and edits files of source programs written in FORTRAN, ALGOL, or Assembly language. In load-and-go operations the FORTRAN Compiler, FORTRAN IV Compiler, ALGOL Compiler, and Assembler generate relocatable binary code onto temporary disc storage. The 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

DOS-III 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. Then separate directives initiate loading (PROG, LOADR) and execution (RUN,*program*).

DOS-III can store the object code of several programs and associated segments and subroutines on the disc. The Relocating Loader retrieves them from the disc, and relocates them into executable absolute program units.

ALGOL COMPILER

The ALGOL Compiler consists of a main program and a data segment which operate under the control of DOS-III. The compiler resides on the disc and is read into main memory when called for by a PROG directive.

Source programs written in ALGOL are accepted either from an input device or from a user disc file and are translated by the ALGOL Compiler into relocatable object programs optionally punched on paper tape (and optionally stored in the Job Binary Area of the disc). The object program can be loaded using the DOS-III Relocating Loader and executed using the RUN or PROG directive.

ALGOL I/O

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

Compiler Operation

The ALGOL Compiler is initiated with a PROG directive, and inputs the source program from an input device, or, if from a source file, from a file specified by 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_4,99]

where P_1 = logical unit number of input device (default is 5; set to 2 for source file input indicated by a JFILE directive)

P_2 = logical unit number of list device (default is 6)

P_3 = logical unit number of punch device (default is 4)

P_4 = lines/page on the source listing (default 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.)

All parameters are optional. If p_1 through p_4 are not present, the default operations are assumed. If 99 is not present, the binary output is not placed in the Job Binary Area.

Messages During Compilation

When the end of a source tape is encountered, the following is output on the system console:

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 the compilation, the following message is output to the system console:

$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 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

- Logical unit 2 has been given for input and no :JFILE has been declared. 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.

Language Considerations

The HP ALGOL control statement has this format:

HPAL [,L,A,B,P], "name" [,P_1] [,P_2]

where *HPAL* is mandatory

L,A,B,P are symbols (any combination is allowed) representing:

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

"name" is the program name (the quotes and a program name are mandatory)

P_1 is a decimal digit between 0 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 $n = 0$ if P_1 is not specified. ERR0 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).

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.

Sense switch control is not used with DOS-III.

EXAMPLE

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

ASSEMBLER

The Assembler, a segmented program that executes in the main-memory User Program Area, operates under control of DOS-III. The Assembler consists of a main program (ASMB) and six segments (ASMBD, ASMB1, ASMB2, ASMB3, ASMB4, ASMB5), and resides on the disc. The main program is read into main memory when called by a PROG directive.

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-III. ASMB can store relocatable code in the Job Binary Area of the disc for on-line execution, as well as punch it on paper tape.

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

Assembler I/O

The Assembly Language I/O EXEC calls should specify the proper logical unit numbers for the DOS-III configuration. (See Section IV.)

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

If the memory protect hardware option is present (and enabled), it protects the resident supervisor from alteration. It 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 register.
- The halt instruction.

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

Assembler Operation

The DOS-III Assembler is initiated with 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 $p_1 = 2$ in the PROG directive. The PROG directive for Assembler should take the following form:

PROG,ASMB

:PROG,ASMB[,P_1,P_2,P_3,P_4,99]

where P_1 = logical unit number of input device (default is 5; set to 2 for source file input indicated by a JFILE directive)

P_2 = logical unit number of list device (default is 6)

P_3 = logical unit number of punch device (default is 4)

P_4 = lines/page on the source listing (default 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.)

All parameters are optional. If p_1 through p_4 are not present, the default operations are assumed. If 99 is not present, the binary output is not placed in the Job Binary Area.

Messages During Assembly

When the end of a source tape is encountered, the following is output on the system console:

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.

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

$END ASMB

If another pass of the source program is required, this message is output at the end of pass one.

$END ASMB PASS

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

:GO

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

$END ASMB CS

and the current assembly stops.

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

$END ASMB XEND

and the current assembly stops.

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

$END ASMB NPRG

and the current assembly stops.

If the Job Binary Area, where binary code is stored by a 99 parameter, overflows, assembly continues but the following message is output on the system console:

JBIN OVF

However, no further binary code is stored in the Job Binary Area.

The next message is printed on a separate line just above 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 (printed on a separate line just above each diagnostic):

PG *ppp*

ppp is the page number (in the listing) of the *previous* error diagnostic. PG 000 is associated with the first error found in the program.

Language Considerations

ASSEMBLER CONTROL STATEMENT. Although only relocatable code can be run under DOS-III, the DOS-III 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" parameter. The "R" parameter, 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.*

NAM STATEMENT. The NAM statement allows up to eight optional parameters. Only the first two parameters are significant in DOS-III.

NAM name [,type] [,link mode]

where *name* is the program name (it should not equal any file name).

type is the program entry type code (octal):

0 — System main memory resident (default)
1 — Disc resident executive supervisor module
2 — Reserved for system
3 — User program, main
4 — Disc resident device driver
5 — User program segment
6 — Library routine
7 — Subroutine
10 — Relocatable binary
11 — ASCII source statements
12 — Binary data
13 — ASCII data
14 — Absolute binary

link mode is the mode of linkage to be performed:

0 — current page linking
non-zero — base page linking (default)

If *type* is 0, 1, 2, or greater than 7, the assembler and DSGEN will accept it, but the Relocating Loader will not.

The *link mode* parameter specifies the mode of linking that will occur at system generation time. If zero, current page linking occurs. If non-zero, base page linking occurs. If omitted, the default condition (non-zero) is assumed and base page linking occurs.

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

Note: DOS-III Assembly language does not contain the ORB statement because information cannot be directly 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.

FORTRAN COMPILERS

The FORTRAN Compilers operate under control of the DOS-III Supervisor. The compilers reside on the disc and are read into main memory only when needed.

FORTRAN and FORTRAN IV are problem-oriented programming languages. Source programs, accepted from either an input device or a user disc file, are translated into relocatable object programs, optionally punched on paper tape, and optionally stored in the Job Binary Area of the disc. The object program can be loaded using the DOS-III Relocating Loader and executed using the RUN or PROG directive.

FORTRAN I/O

FORTRAN I/O statements should specify the proper logical unit numbers for the DOS-III configuration. (See Section IV.)

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

Compiler Operation

The FORTRAN compilers are initiated with a PROG directive, and input the source program from an input device, or, if from a source file, from a file specified by a JFILE directive. The PROG directive for FORTRAN compilers should take the following form:

PROG,FTN[4]

:PROG,FTN[,P_1,P_2,P_3,P_4,99]

:PROG,FTN4[,P_1,P_2,P_3,P_4,99]

P_1 = logical unit number of input device (default is 5; set to 2 for source file input indicated by a JFILE directive)

P_2 = logical unit number of list device (default is 6)

P_3 = logical unit number of punch device (default is 4)

P_4 = lines/page on the source listing (default 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.)

All parameters are optional. If p_1 through p_4 are not present, the default operations are assumed. If 99 is not present, the binary output is not placed in the Job Binary Area.

Messages During Compilation

When the end of a source tape is encountered, the following is output on the system console:

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 output on the system console:

$END, FTN[4]

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 and compilation continues:

JBIN OVF

There is no further loading into the Job Binary Area.

The compiler terminates if

- logical unit 2 has been given for input and no JFILE has been declared. ($END,FTN[4] is not output.)
- There are not enough work tracks for the compiler. The following message is output:

  # TRACKS UNAVAILABLE

- A colon occurs 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 output.

  IE nnnnn

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

Language Considerations

FORTRAN CONTROL STATEMENT. Besides the standard options described in the FORTRAN manual, two 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 then supply an error routine, ERR*n*. If this option does not appear, the standard library error routine, ERR0, is used. The error routine is called when an error occurs in ALOG, SQRT, .RTOR, SIN, COS, .TROI, EXP, .ITOI, or TAN.

Extended and Auxiliary Statements

In addition to the standard FORTRAN statement, the FORTRAN compiler running under DOS-III supports the following extensions and additions:

1. extended PROGRAM statement
2. additional DATA statement
3. additional EXTERNAL statement

Execution of the following two FORTRAN statements results in special processing in the DOS-III environment:

1. PAUSE
2. STOP

PROGRAM STATEMENT

The program statement includes an optional type parameter.

PROGRAM name [,type] [,link mode]

name	is the five-character name of the program (and its main entry point. When the program is executed using a RUN or PROG directive, this *name* is used.
type	is a decimal digit specifying the program type. Only types 3 (main), 5 (segment), and 6 or 7 (library) are significant in DOS-III. The type is set to 3 if not given.
link mode	is the mode of linkage to be performed: 0 indicates current page linking and any non-zero digit indicates base page linking (default).

DATA STATEMENT

The DATA statement sets initial values for variables and array elements. The format of the DATA statement is

DATA $k_1/d_1/,k_2/d_2/, \ldots ,k_n/d_n/$

where k is a list of variables and array elements separated by commas, d is a list of (optionally signed) constants, separated by commas and optionally preceded by j* (j is an integer constant).

The elements of d_1 are serially assigned to the elements of k_1. The form j* means that the constant is assigned j times. The k_1 and d_1 must correspond one-to-one.

Elements of k_1 must 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)/1.0,2.0,3.0/,I(1),I(2)/2*1/
```

EXTERNAL STATEMENT

With the EXTERNAL statement, 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* is the entry point of a function, subroutine, or library program.

EXAMPLE

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

ERROR E-0018 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 already exist with the first character changed to "%".

PAUSE AND STOP

PAUSE causes the following message to be output to the system console:

PAUSE xxxx

where *xxxx* is an optional 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.

ERR0 LIBRARY ROUTINE

ERR0, the error print routine referred to under the FORTRAN or ALGOL control statement, outputs the following message to the system console whenever an error occurs in a library routine:

name: nn xx

where *name* is the name of the user's program,
nn is the routine identifier, and
xx is the error type.

The compiler generates calls to ERR0 automatically. If the FORTRAN (or ALGOL) control statement includes an *n* option, the call will be to ERR*n*, a routine which the user must supply.

DOS-III RELOCATING LOADER

The DOS-III Relocating Loader accepts relocatable object programs which have been translated by the Assembler, ALGOL Compiler, or FORTRAN Compilers. It generates an executable main-memory image of each such program. The relocatable programs may enter the loader as

- Job Binary Area programs translated during the current job
- User files
- Punched tapes, magnetic tapes
- 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 sub-routines. 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. With the STORE,P directive, the program may be stored as a permanent user file to be run during a later job. If the Loader is to be re-executed during a single job, the Job Binary Area must be cleared (using the CLEAR directive) to prevent duplicate program names.

PROG,LOADR

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

Format

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

P_1 = 0 for loading from JBIN and relocatable library (default)
= 2 for loading from JBIN, user files, and relocatable library
= n for loading from JBIN, user files, relocatable library, and paper tape or magnetic tape (logical unit *n*)

P_2 = list device logical unit number (default is 6)

P_3 = 0 for no DEBUG, ≠ 0 for DEBUG (default is 0)

P_4 = 0 for base page linking, ≠ 0 for current page linking (default is 0)

P_5 = 0 for system default program bounds (e.g., UBFWA-UBLWA and UMFWA-UMLWA); = 1 for user-specified program bounds (default is 0)

Comments

INPUT PARAMETER [P_1]. Note the hierachy here. If *n* is specified, the JBIN area is still scanned first, then user files are requested and, finally, the peripheral relocatable input is accepted.

If P_1 ≠ zero, the Loader first expects a list of relocatable file names. In keyboard mode, the Loader requests:

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 is entered as

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

following the PROG directive (or following the bounds parameters if P_5 = 1). If there are no user files, a /E record must be entered.

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 parentheses 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.)

DEBUG PARAMETER [P_3]. 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 system console.

CURRENT PAGE LINKING PARAMETER [P_4]. If requested to do so ($P_4 \neq$ zero), the Loader attempts to place necessary program links on the current page of memory as opposed to the base page, to provide more area on the base page for large programs.

Note: While using the Loader with the current page linking option, remember that:

a. Current page linking cannot be used on programs which use main memory following the program area for writing data (at execution time). For instance, the Assembler builds its symbol table immediately following the last word of the largest segment.

b. Programs should be broken into subroutines of less than 2K because links are generated only at the beginning and end of the program. Links cannot be inserted into the middle of a program since the boundary between program and links may fall in the middle of a skip or jump sequence. If the program spans more than two pages, the middle page(s) will have no area available for current links and will use base page links; thus, the potential for greater efficiency will be lost.

PROGRAM BOUNDS SPECIFICATION PARAMETER [P_5]. The user has the option of specifying the base page bounds and the main memory bounds for the relocatable modules being loaded. If parameter P_5 in the PROG,LOADR directive is zero, the program bounds are determined by the system pointers:

UBFWA	lower base page bound
UBLWA	upper base page bound
UMFWA	lower main memory bound
UMLWA	upper main memory bound

If P_5 is equal to one, the user can specify his own memory bounds. In batch mode, the Loader reads the bounds from the input device immediately following the :PROG, LOADR directive. The bounds are in the form of two records: the first record is interpreted as the lower and upper base page bounds, specified by two octal constants separated by a comma. If an error occurs in the first

record, the Loader outputs an L18 error message. The second record is interpreted as the lower and upper main memory bounds, specified by two octal constants separated by a comma. If an error occurs in the second record, the Loader outputs an L19 error message. If any of the bounds are omitted, the appropriate system default value is used. In keyboard mode, the two records are entered in response to the messages

BP BND [L,U]?
PROG BND [L,U]?

If an error occurs while entering the bounds in keyboard mode, the user can re-enter the bounds (after an L18 or L19 error message). If an L18 or L19 error message occurs in batch mode, the Loader aborts the job.

I/O Drivers

The Loader will accept Type 4 programs (Disc Resident Device Drivers) and store them as such in the user directory. Type 4 programs cannot be combined with any other program type during any given load operation.

Loader Operation

The DOS-III Relocating Loader is a two-pass Loader. The first pass consists of setting the bounds, inputting and scanning relocatable programs to build the necessary tables (program name table and a table of entry points and externals), and matching entry points with externals. The second pass involves the relocation of the programs into an absolute core image format on the disc.

INPUTTING AND SCANNING THE PROGRAMS. Programs are scanned (and input, if necessary) according to P_1 in the PROG,LOADR directive. (Only non-disc relocatable programs must be *input;* there are stored temporarily on the Work Area of the disc for processing during the second pass.) Since main programs are matched with segments during the scan, each main program must be loaded before any of its segments.

If paper tape input is requested, the following messages are output to the system console:

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 output to the system console:

I/O ERR ET EQT# nn *(paper tape only—not magnetic tape)*
LOAD TAPE
LOADR SUSP
@

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

If a checksum error occurs when loading relocatable programs from paper tape, the Loader prints an L01 error message and returns to the paper tape load point with the messages

LOAD TAPE
LOADR SUSP
@

The operator can attempt to reload the program by placing the tape in the reader at the beginning of the program and typing :GO.

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 output and the external references are listed, one per line.

To load additional programs from a peripheral device, the operator types

:GO,0[,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, enter

:GO,2

and the appropriate prompt is output:

ENTER FILE NAME(S) OR /E

RELOCATING PROGRAMS. The main and segment names (from the PROGRAM, HPAL, or NAM records) 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 existing 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 main memory image, stores it on the disc, places the name in the user directory where it remains during the current job, and lists (on the logical unit specified by the P_2 parameter) the program address map and entry points. After each main program, any associated segments are loaded in the same way. When the Loader is completely finished, the following message is output:

LOADR COMPLETE

During the current job, the absolute main memory 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, P directive.

If no programs are entered, the Loader outputs the following messages and terminates:

NO PROGRAMS LOADED
LOADR COMPLETE

Loader error messages are given in Section XV.

EXAMPLE

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

```
:CLEAR                          Eliminate any programs from the job binary area
@
:PROG,LOADR,5                   Paper tape input is specified
ENTER FILE NAME(S) OR /E
ALGLM,/E                        One disc file is specified
LOAD TAPE
LOADR SUSP                      Place paper tape in input device

@:GO                            Return to Loader
I/O ERR ET EQT# 02              End of paper tape
LOAD TAPE
LOADR SUSP

@:UP,2                          Declare input device ready

@:GO,1                          Specify no more paper tapes
```

The following is then output on the standard list device (logical unit 6):

```
RELOCATING LOADER

NAME        PROG BOUNDS    BP BOUNDS

ALGOL
   *HPAL    26601                          Main program
   *%HPST   27005                               Main's entry points
  .EAU.     30370          01402           Subroutine
   *.MPY    30370                                Subroutine's entry points
   *.DIV    30375
   *.DLD    30402
   *.DST    30407
  %WRIT     30440          01407
   *%WRIT   30626
   *%WRIF   30522
   *%WBUF   30725
  SREAD     31141          01411
   *%READ   31141
   *%JFIL   31612
   *%RDSC   31563
  DUMRX     31677          01412
   *$LIBR   31677
   *$LIBX   31724
  .OPSY     31757          01412
   *.OPSY   31757

(BOUNDS)    16000 32017    00716 01415     Main programs bounds


ALGL1                                      Segment
   *ALGL1   32461                               Segment's entry points
   *%LNAL   32020
   *%ABAL   32017

(BOUNDS)    32017 32463    01415 01416          Segment's bounds


LOADR COMPLETE          Console message to indicate normal Loader completion

@:ST,P                  Make newly created programs permanent disc files
```

THE RELOCATABLE LIBRARIES

There are two System libraries, or collections of relocatable subroutines that can be used by DOS-III: 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. 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 external function references in FORTRAN, and by CODE procedures in ALGOL.

When the system is generated, several combinations of libraries are possible. Every system should 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 manual *Relocatable Subroutines (02116-91780).*

DEBUG LIBRARY SUBROUTINE

RTE/DOS DEBUG, a subroutine of the Relocatable Library, allows the programmer to check for logical errors during program execution. If the third parameter specified in the PROG, LOADR directive is non-zero, the DEBUG subroutine is appended to the user program being loaded. The primary entry point (where execution begins) is set to DEBUG. When the program is executed with a RUN directive, the DEBUG subroutine has control and displays the message:

BEGIN 'DEBUG' OPERATION

The programmer may enter any DEBUG operation directive. Illegal entries are ignored but result in the message:

ENTRY ERROR

Re-enter the DEBUG operation directive correctly.

DEBUG OPERATIONS

B,*n*	Instruction breakpoint at octal address *n* (Note: if *n* = JSB EXEC, a memory protect violation occurs)
D,A,n_1[,n_2]	ASCII dump of octal main memory address n_1 or from n_1 through n_2
D,B,n_1[,n_2]	Binary dump of octal main memory address n_1 or from n_1 through n_2
M,*n*	Sets absolute base of relocatable program unit at octal address *n*
R[,*n*]	Execute user program starting at octal address *n* or 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,*n*,*d*	Set octal value *d* in octal address *n*
S,*n*,d_1,d_2, . . . ,d_n	Set octal values d_1 through d_n in successive memory locations beginning at octal address *n*
W,A,*d*	Set A-register to octal value *d*
W,B,*d*	Set B-register to octal value *d*
W,E,*d*	Set E-register to octal value *d* (0=off; non-zero = on)
W,O,*d*	Set Overflow to octal value *d* (0 = off; non-zero = on)
X,*n*	Clear breakpoint at octal address *n*
A	Abort DEBUG operation.

SPECIAL CONSIDERATIONS

Because of the extended instruction group coding available to the programmer using an HP 21MX Computer Series system, the current RTE/DOS DEBUG subroutine should not be used within these systems.

For systems based on an HP 21MX series processor, a modified version of the subroutine called HP 21MX RTE/DOS DEBUG is available and should be used in place of the current subroutine. HP 21MX RTE/DOS DEBUG can be used only on HP 21MX Computer Series systems, it cannot be executed successfully on systems based on an HP 2100A or HP 2100S processor.

During the Program Input Phase of RTE or DOS-III system generation, load the HP 21MX RTE/DOS DEBUG subroutine from paper tape (relocatable binary code) immediately after loading the RTE/DOS Relocatable Library. An error message indicating the existence of a duplicate program name will be displayed but the system generator will proceed to replace the current RTE/DOS DEBUG subroutine with the HP 21MX version.

Externally, with a few differences, the HP 21MX RTE/DOS DEBUG subroutine appears the same as the current version in the RTE/DOS Relocatable Library. The differences are as follows:

1. HP 21MX RTE/DOS DEBUG allows the programmer to set breakpoints on instructions which are extensions to the base set microcode. Breakpoints set on standard HP 21MX instructions—specifically, base set, base set extension (extended instruction group), single precision floating point arithmetic, or extended arithmetic unit (EAU) instructions — are processed normally; that is, the break occurs before execution of the instruction and is not cleared if the programmer resumes execution of his program. Breakpoints set on instructions which are extensions to the standard instruction set — FFP, user written instructions, and so forth — result in the breakpoint being cleared after execution of the break.

   Note: In the current RTE/DOS DEBUG subroutine, setting a breakpoint on a non-EAU multiple-word instruction results in the incorrect execution of the instruction at the breakpoint.

2. HP 21MX RTE/DOS DEBUG displays the contents of the X-register and Y-register as part of the standard breakpoint message.

3. HP 21MX RTE/DOS DEBUG provides two additional operation directives which allow the programmer to set the X-register or Y-register to specific values. These directives are:

   W,X,*d* Set X-register to octal value *d*

   W,Y,*d* Set Y-register to octal value *d*

SEGMENTED PROGRAMS

User programs may be structured into a main program and several segments, as shown in Figure 5-1. 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 main memory by EXEC calls, when needed. Only one segment may be in main memory at a time. When a segment is read into main memory, it overlays the segment previously in main memory.

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 the Loader, the main program must be entered prior to its segments. One external reference from each segment to the main routine is required for DSGEN or the Loader to link the segments and main programs. Also, each segmented program should use unique external reference symbols. Otherwise, DSGEN or the Loader may link segments and main programs incorrectly.

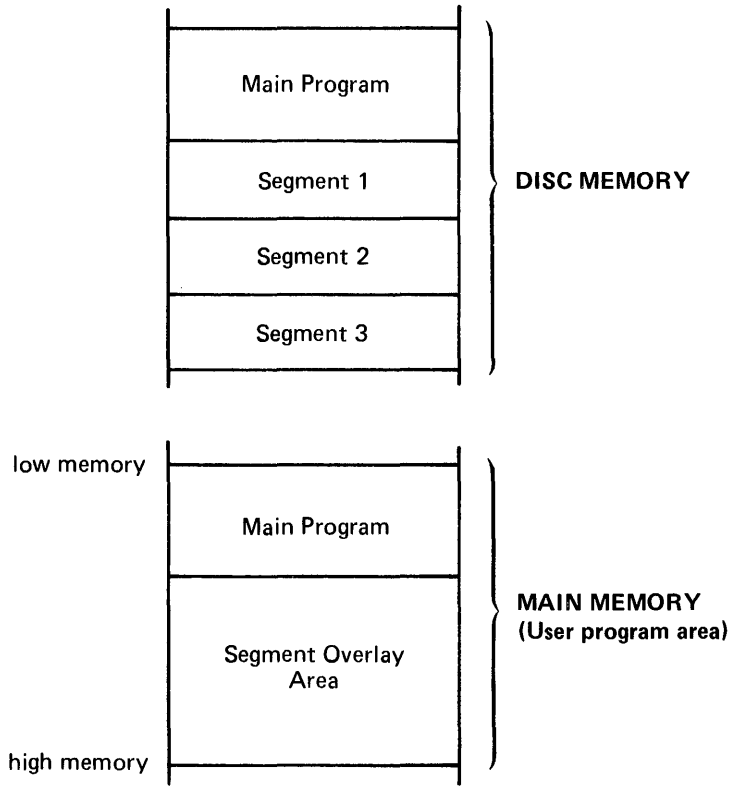


Figure 5-1. Segmented Programs

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

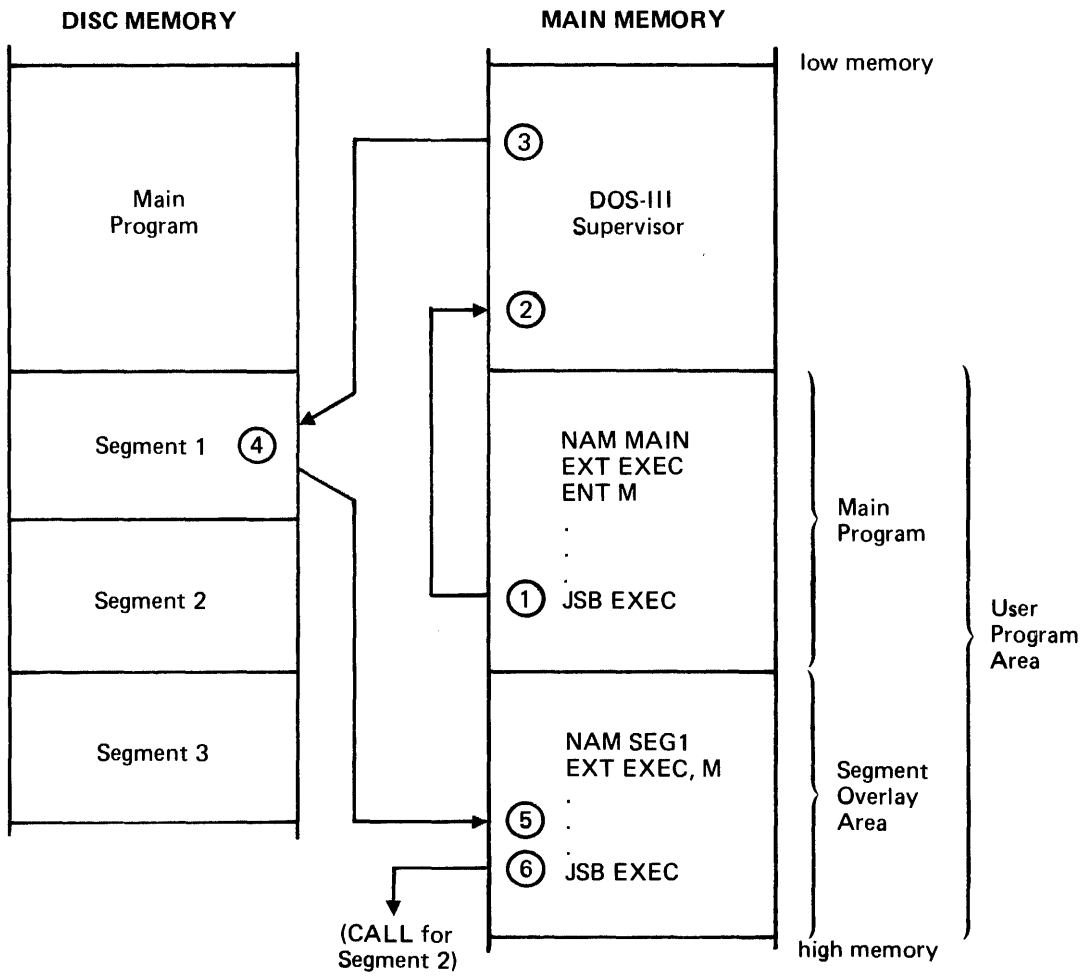


Figure 5-2. Main Calling Segment

Figure 5-3 shows how DOS-III processes the call from the segment (7) by locating the segment on the disc (8-9), loading it into main memory (10), and beginning execution of it.

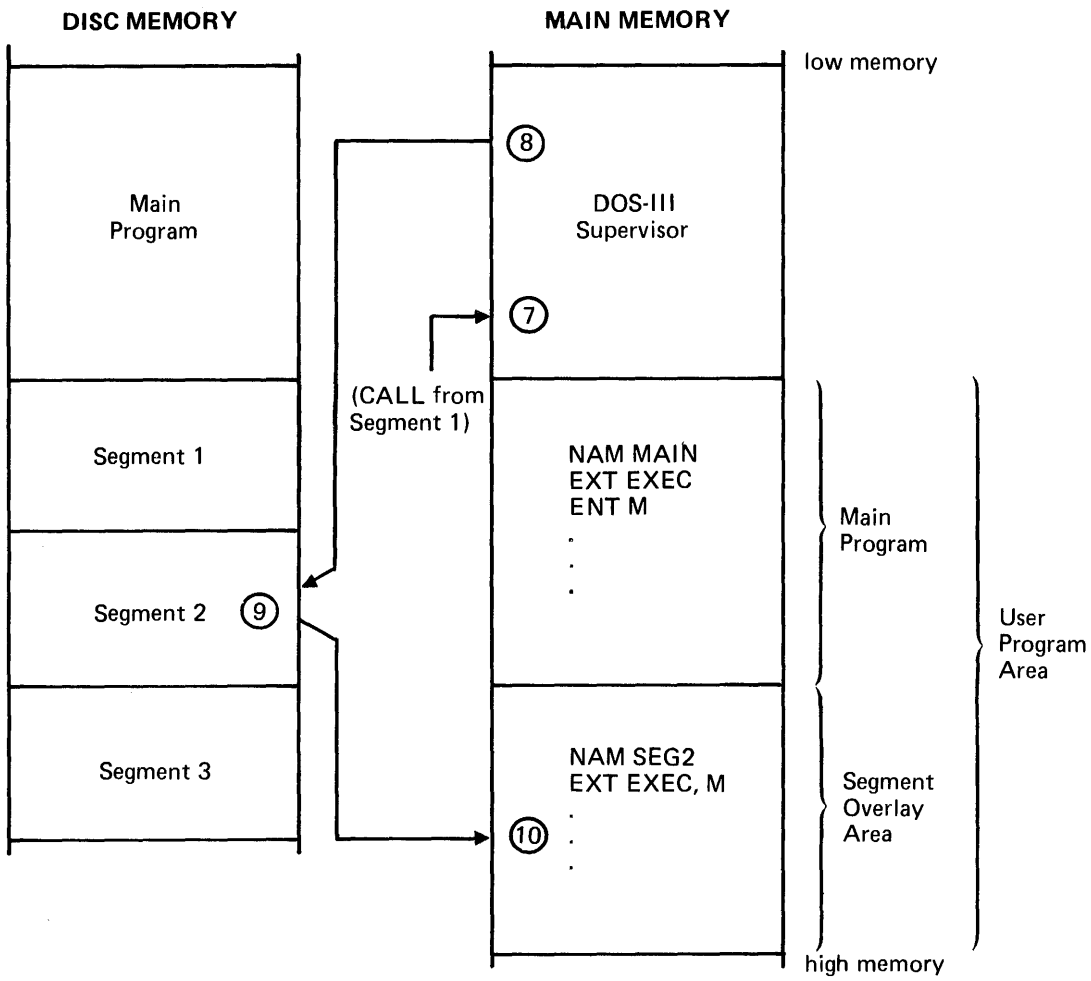


Figure 5-3. Segment Calling Segment

When a main program and segment are currently residing in main memory, they operate as a 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 or JSB instruction. (See Figure 5-4.) A matching entry symbol must be defined as the destination in the other program. DSGEN or the Loader associates the main programs and segments, replacing the symbolic linkage with actual absolute a 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 main memory before any JMP instructions are executed.

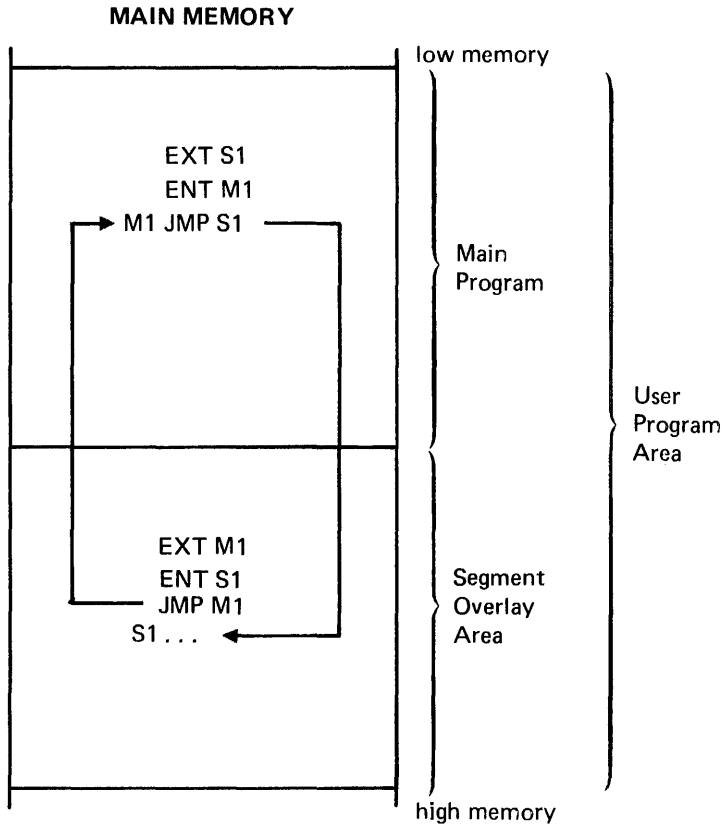


Figure 5-4. Main-to-Segment Jumps

FORTRAN 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 by using the SEGMENT LOAD EXEC call (RC ᴞDE = 8) from the main or another segment. A dummy CALL to the main must appear in each segment to ensure that proper linkage will be established between the main and its segments.

Once a segment is loaded, control is passed to it and execution begins at its primary entry point (or at the address specified in base page location 135_8). The segment, in turn, may call another segment using another SEGMENT LOAD EXEC call. Communication between the main program and segments may be through COMMON or via parameters passed in the SEGMENT LOAD or SEGMENT RETURN EXEC calls.

Any segment may return to the main program at the statement immediately following the initial SEGMENT LOAD EXEC call (RCODE = 8) by executing a SEGMENT RETURN EXEC call (RCODE = 29). (See Section III for a description of these EXEC calls.) However, segments may not return directly to other segments.

ALGOL Segments

ALGOL programs can be segmented if certain conventions are followed. A segment must be defined as type 5 in the HPAL control statement. The segment must be initiated by using the SEGMENT LOAD EXEC call (RCODE = 8) 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, each segment must declare the following:

PROCEDURE MAIN;CODE;

Once a segment is loaded, control is passed to it and execution begins at its primary entry point (or at the address specified in base page location 135_8). The segment, in turn, may call another segment using another SEGMENT LOAD EXEC call. Communication between the main program and its segments may be through parameters passed in the EXEC call.

Any segment may return to the main program at the statement immediately following the initial SEGMENT LOAD EXEC call by executing a SEGMENT RETURN EXEC call (RCODE = 29). (See Section III for a description of these EXEC calls.) However, segments may not return directly to other segments.

SECTION VI
Typical DOS-III Job Decks

1. 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,NEXTJ
```

Source Program

2. LOAD AND EXECUTE A RELOCATABLE FILE

```
:JOB,LOADE
:PROG,LOADR,2
AFILE,/E
:STORE,P,TEST
:RUN,TEST
10
23
  .
  .
  .
  .
51
:JOB,NEXTJ
```

Data

3. STORE, EDIT, COMPILE, LOAD AND RUN A PROGRAM

```
:JOB, EVERY
:STORE,S,SOURC,5
FTN,B,L
      PROGRAM ZOOM
      DIM   I(10)
        .
        .
        .
      END$

::
:LIST,S,6,SOURC
:EDIT,SOURC,5
/I,2
  .
  .
  .
/E
:JFILE,SOURC
:PROG,FTN,2,6,4,56,99
:PROG,LOADR
:RUN,ZOOM
123.62
  .
  .
  .
00001
:RUN,ZOOM
321.5
  .
  .
  .
0.56
:JOB,NEXTJ
```

Source Program

Edit List

Data for first run

Data for second run

4. LIST ONLY ERROR STATEMENTS ON SYSTEM CONSOLE IN A COMPILE

```
:PR,FTN4,,1
FTN4
      PROGRAM EX1
            .
            .
            .
      END
$
```

5. COPY A SYSTEM FROM SUBCHANNEL 1 TO SUBCHANNEL 0

```
:JO
JOB          TODAY          TIME=0831 MIN. 43.3 SECS.
@
:UD,,0                      Interrogates the system as to what label
LBL=SYSTEM                  is on subchannel 0.
@

:UD,SYSTEM,0                Change current user disc to subchannel 0
@
:UD                         Verify
SUBCHAN=0
LBL=SYSTEM
@
:IN,*                       Purge system and user files on subchannel 0
DOS LABEL SYSTEM
OK TO PURGE?
YES
@
:UD                         Verify purge
SUBCHAN=0
UNLBL
@

:UD,*,1                     Change current user disc to subchannel 1
@
:UD                         Verify
SUBCHAN=1
UNLBL
@
:DD,X                   }
@                       }   Copy system to subchannel 0
:UD,*,0                 }
@
:UD
SUBCHAN=0
LBL=SYSTEM
@
:EJ
END JOB      RUN=0008 MIN. 01.7 SEC.  EXEC=0000 MIN. 00.0 SEC.
@
```

PART 2
DOS-III Extended File Management Package (EFMP)

SECTION VII
EFMP Organization

The DOS-III Extended File Management Package (EFMP) extends the file handling capabilities of DOS-III by allowing the user to create and use files with different record lengths, security codes, and other conveniences. EFMP consists of a series of additional EXEC modules and a utility program; it maintains a file structure that operates within, and in addition to, the standard DOS-III file structure.

ENVIRONMENT

EFMP functions in the DOS-III environment. It is implemented through a set of EXEC modules which are incorporated into DOS-III at system generation time: the EXEC modules are invoked using the standard EXEC call mechanism.

FUNCTIONS AND STRUCTURE

The EFMP modules themselves allow any program executing in the user area to Initialize EFMP areas, Create/Destroy, Open/Close, Read/Write, Reset, Repack, Copy, Change Name, and Post files on the moving-head disc. Also, EFMP makes available detailed status information on all files and packs known to it. EFMP may be accessed conversationally from the keyboard by using UTIL, a utility program that executes in the User Area.

DOS-III Files vs. EFMP Files

DOS-III maintains files that are referenced by five-character names and relative sector numbers. The user can access these files in either a keyboard mode (via directives) or in a programming mode (via EXEC calls). In keyboard mode, the user creates a file with the STORE directive and operates on that file with directives such as :EDIT and :DUMP. In programming mode, the DOS-III files are accessed by EXEC calls such as FILE READ/WRITE and FILE NAME SEARCH.

In addition to the file structure, DOS-III maintains a subchannel (or user disc) identification scheme. User discs are first formatted either during system generation or by a special function of the system generator. These functions format the hardware tracks and set up information such as the Label Presence Code and System Proprietary Code. After a disc pack is formatted, the INITIALIZE directive is used to set up labels (six-character codes), change labels, and purge old discs.

EFMP operates within this file structure of DOS-III to set up and maintain additional—but distinctly different— files. Areas of discs within DOS-III (hereafter referred to as EFMP areas) are turned over to EFMP exclusively. The user must identify them with a pack number of the form PN*xxx*, where *xxx* is a decimal integer. The procedure for doing this is described under "Set Up."

Within an EFMP area, EFMP creates files of its own that are not known to DOS-III. They are identified by a fixed-length name, contain a grouping of specified length records, and have a security code. Since only the DOS-III files can be created and accessed by directives, all EFMP files must be used through the EFMP EXEC calls or the UTIL program. EFMP files are limited in size only by the requirement that they fit within one subchannel or pack.

Note: All references to files within this Part will mean EFMP files, not DOS-III files, unless specifically stated otherwise.

Duplicate Pack Numbers

EFMP pack numbers are always unique on any given platter, but not necessarily unique across platters. To minimize the possibility of accessing a duplicate pack number, the user should (if possible):

1. Create unique pack numbers.
2. Have platters containing EFMP areas mounted on the subchannel designated as the current user subchannel.

EFMP Buffers and Tables

To provide maximum flexibility in main memory size and speed of file accessing, EFMP allows the user to define (at execution time) the size and location of the tables and buffers required in main memory by EFMP. Two areas are defined by the user and provided in his program space:

1. Opened File Table
2. Temporary Record Buffers

The Opened File Table contains all information necessary for EFMP to identify and access files belonging to the user. The minimum size of the Opened File Table is one sector (128 words) and allows up to seven files to be opened concurrently.

EFMP uses the Temporary Record Buffers as an intermediate storage area between the disc and the user's record buffer. The user defines the number of Temporary Record Buffers and the size of each. There must be at least one buffer and it must be at least two sectors (256 words) long. Particular files and buffers can be linked to increase the access speed of files. The effect of varying the number and size of these buffers cannot be predicted exactly and must be determined empirically by trial and error.

CAUTION: SINCE THESE TABLES AND BUFFERS EXIST IN THE USER AREA AND ARE NOT PROTECTED, EXTREME CAUTION MUST BE TAKEN NOT TO MODIFY THEM IN ANY WAY.

Logical Read vs. Physical Read

A logical read occurs each time the user requests a record from a file. At that time EFMP checks the appropriate Temporary Record Buffer to determine if the requested record is already in main memory. If in main memory, the record is transferred to the user's record buffer without actually physically reading the disc. If the record is not present in main memory, the necessary disc transfers are performed (physical reads—and writes, if necessary) to bring the record into main memory. If the Temporary Record Buffer is larger than the record size, several records are brought into main memory at once.

Logical Write vs. Physical Write

A logical write occurs each time a user requests that a record be written to a file. At that time, EFMP determines if that record is present in the Temporary Record Buffer; if it is, EFMP simply transfers the data in the user's record buffer to the Temporary Record Buffer and flags it as "must be written." Each succeeding read or write is treated in the same manner until a logical record transfer occurs for which the record is not in main memory, or until the last record in the Temporary Record Buffer is logically written. In these cases, the EFMP must physically write the records in the Temporary Record Buffer (i.e., post them) on the disc.

If the record is not present in main memory on a write request, EFMP locates the record on the disc and transfers it physically into the Temporary Record Buffer. The data to be written is then transferred from the user buffer to the Temporary Record buffer and flagged as "must be written." The read before write is necessary because records do not necessarily fall on sector boundaries in the disc. If a CLOSE or POST request occurs, all buffers flagged are written to the disc.

Update-Writes vs. Append-Writes

The purpose of an update-write is to change the contents of an existing record; the purpose of append-write is to add new records onto the end of a file. EFMP writes a record as an update-write whenever the record specified exists in a previously accessed section of a file.

EFMP writes a record as an append-write whenever the record specified is beyond the previously accessed section of a file. In this case, EFMP automatically inserts zeros into all records (if any) between the highest record previously written and the new record.

SET UP

There are two prerequisites for EFMP. First, the EFMP EXEC modules must be included in DOS-III when the system is generated. Second, when DOS-III is running, the user must create EFMP areas on formatted DOS-III packs or cartridges.

An EFMP area is created by issuing a STORE, B directive in this format:

:STORE,B,PNxxx,sectors

where *xxx* is a unique decimal number,

PNxxx is the unique pack number, and

sectors is the number of sectors of the EFMP area.

Note: EFMP changes the file from Type-B to Type-A during initialization (see "Initialize").

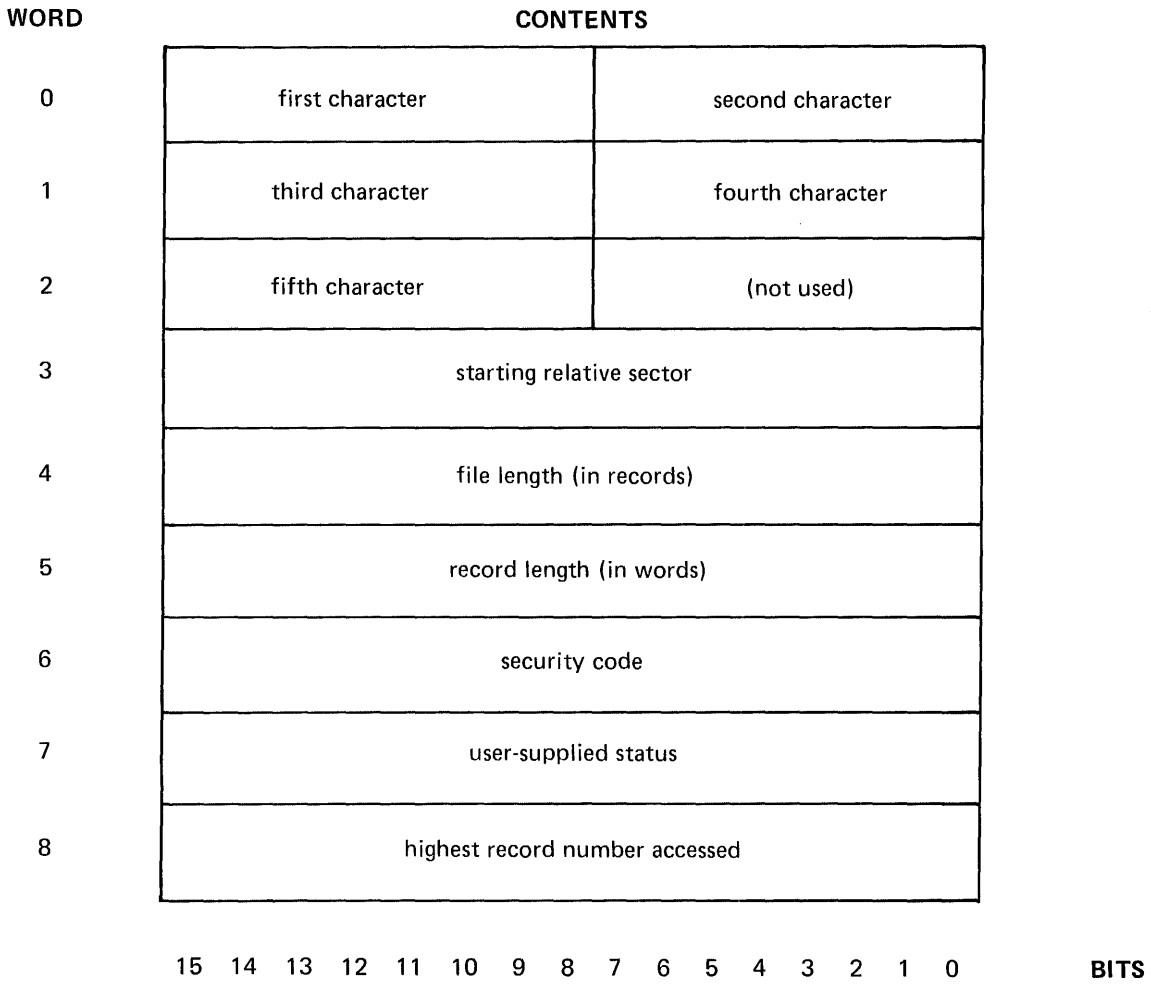


Figure 7-1. EFMP File Disc Directory Format

SECTION VIII
EFMP EXEC Calls

The method of communication between a user program and EFMP is through the standard DOS-III EXEC call format (discussed in Section III of this manual).

One standard DOS-III request code (RCODE = 24) is reserved for EFMP requests. The DOS-III operating system combines this request code with an EFMP function number to determine which action the user EXEC call is requesting. The EFMP function numbers are one element in each of the EFMP EXEC calling sequences.

FORMAT FOR EFMP EXEC CALLS

In this section, only the Assembly language calling sequences are given for the EFMP EXEC calls. The methods for converting these calling sequences to FORTRAN or ALGOL are described in Section III.

The EFMP EXEC calls described in this section are presented in ascending order, by EFMP function number. The STATUS EXEC call (EFMPF = 10) has several status function numbers: these are presented in ascending order, by status function number.

Note: A complete list of EFMP error codes can be found in PART 5 of this manual, "Error Codes and Messages."

DEFINE

Purpose

To define, before any other EFMP calls are made, the number of 16-bit words within the user program to be used by EFMP for its internal buffers and tables.

Assembly Language

```
                JSB     EXEC
                DEF     *+9         Return address
                DEF     RCODE       Request code
                DEF     EFMPF       EFMP function number
                DEF     OPNTB       Opened-file table address
                DEF     OPNSZ       Opened-file table size
                DEF     TRBUF       Temp. record buffer address
                DEF     NOTRB       Number of temp. record buffers and number of
                                    active pack numbers
                DEF     TRBSZ       Temp. record buffer size
                DEF     ERRNO       Error number
                return              Continue execution
                  .
                  .
                  .
RCODE           DEC     24
EFMPF           DEC     1
OPNTB           BSS     n           Opened-file table (n is the size)
OPNSZ           DEC     n           Size of opened-file table (in 16-bit words,
                                    see Comment 1)
TRBUF           BSS     m           Beginning of temp. record buffers, see Comment 2
NOTRB           DEC     p           No. of temp. record buffers, see Comment 2
(NOTRB+1)       DEC     n           n = the maximum number of unique EFMP pack
                                    numbers active (MAXPK), see Comment 4
TRBSZ           DEC     q           Size of each temp. record buffer (in sectors)
ERRNO           BSS     1           Return point for error codes
```

Comments

1. The size of the Opened-file table (n) can be calculated by this formula:

   n = 4(MAXPK)+ 3*(NOTRB)+16*(Max. no. of files to be opened)*

   The minimum size of this table is 128 words. This allows approximately seven files to be opened concurrently.

2. There must be at least one temporary record buffer and it must be at least two sectors long (256 words). There may, however, be more buffers and they may be more than two sectors in size. All of the space for these buffers must be allocated starting at the location TRBUF. Increasing the number of buffers allows disc efficiency to be increased by assigning a buffer exclusively to one file. Increasing the size of each buffer increases the speed of disc accessing by allowing more than one sector to be transferred per disc access.

   The total size of the Temp. Record Buffers (m) can be calculated by the following formula:

   $m = NOTRB * TRBSZ * 128$

   (The minimum value for TRBSZ is 2.)

3. All the tables and buffers are fixed by DEFINE until the end of a program, or until another DEFINE. Each time a DEFINE occurs, all information contained in tables and buffers is lost, all pointers are reset, and EFMP assumes a fresh start. At the end of each program, DOS-III calls EFMP to perform a POST on any records flagged as "must be written."

4. MAXPK indicates the maximum number of unique EFMP pack numbers a user will have active at any one time. A pack number is active when one or more of its files are opened by a user through an OPEN call (or for PN000 through a CREATE call).

CREATE

Purpose

To set up a directory on disc with all of the information necessary to create a file that can be accessed at a later time.

Assembly Language

```
                JSB     EXEC
                DEF     *+9         Return address
                DEF     RCODE       Request code
                DEF     EFMPF       EFMP function number
                DEF     FNAME       File name
                DEF     PAKNO       Pack number
                DEF     FLGTH       File length (in records)
                DEF     RLGTH       Record length (in words)
                DEF     SCODE       Security code and user status
                DEF     ERRNO       Error number
                return              Continue execution
                  .
                  .
                  .
RCODE           DEC     24
EFMPF           DEC     2
FNAME           ASC     3,xxxxx     xxxxx is the name to be applied to the file
                                    (first two characters cannot be zero or 177400(8))
PAKNO           DEC     p           p is the pack number, see Comments
FLGTH           DEC     q           q is the number of records in the file;
                                    (1 ≤ q ≤ 32,767)
RLGTH           DEC     r           r is the number of words in a record;
                                    r must be less than or equal to 1/2 the size
                                    of the temp. record buffer
SCODE           OCT     s           s is any 16-bit combination to be checked by EFMP
                                    during OPEN and DESTROY
(SCODE+1)       OCT     t           t is any 16-bit combination of status information
                                    desired by the user (referred to as USTAT elsewhere)
ERRNO           BSS     1           Return point for error codes
```

Comments

1. If PAKNO is a number between 1 and 999 it indicates the EFMP area in which the file is to be created. When EFMP creates a file, it reserves the necessary area on the disc after the last previous file generated. No attempt is made to search for an area between files. If PAKNO is equal to –1, the file is to be created in any EFMP area that is available.

2. If PAKNO equals zero, the file is placed on the Work Area of the disc and no area will be reserved in the EFMP areas. When such a temporary file is created, the only directory information that is maintained is in the Opened-File Table. A disc-based directory is not maintained. Also, since the directory information is established in main memory during the CREATE function, the OPEN function is not required. The only reason for using an OPEN call for a temporary file is to assign it to a specific Temporary Record Buffer or to change the starting record number to a value other than 1. If no OPEN call is given, the first Temporary Record Buffer is used.

3. When the Work Area is used for temporary files, EFMP reserves this whole area and identifies it as PN000. In order to keep PN000 from using the entire Work Area, the user must enter a STORE,B,PN000 directive for the system disc with the desired number of sectors. When EFMP has terminated, the user should PURGE the file PN000 from the Work Area.

DESTROY

Purpose

To eliminate the directory information for a particular file from main memory and the disc. The user must specify the correct security code for the file. The disc area is repacked only for temporary files. To repack the EFMP areas use the REPACK EFMP call.

Assembly Language

```
                JSB     EXEC
                DEF     *+7         Return address
                DEF     RCODE       Request code
                DEF     EFMPF       EFMP function code
                DEF     FNAME       File name
                DEF     PAKNO       Pack number
                DEF     SCODE       Security code
                DEF     ERRNO       Error number
                return              Continue execution
                  .
                  .
                  .
RCODE           DEC     24
EFMPF           DEC     3
FNAME           ASC     3,xxxxx
PAKNO           DEC     n           If n = 0, then FNAME refers to a temporary file
                                    (if n ≥ 1 and n ≤ 999, FNAME is to be located in
                                    this EFMP area; if n = -1, EFMP searches all of its
                                    areas until it finds a file that matches FNAME)
SCODE           OCT     s           s is the security code for the file established by the
                                    CREATE EFMP call; security code ignored on
                                    temporary files
ERRNO           BSS     1           Return point for error codes
```

OPEN

Purpose

To make a previously created file accessible by extracting the necessary file information from the disc directories and placing it in main memory. The number of files that can be opened at any one time is limited by the size of the Opened File Table (see DEFINE).

Assembly Language

```
            JSB     EXEC
            DEF     *+9         Return address
            DEF     RCODE       Request code
            DEF     EFMPF       EFMP function code
            DEF     FNAME       File name
            DEF     PAKNO       Pack number
            DEF     RCDNO       Record number
            DEF     SCODE       Security code
            DEF     BUFNO       Buffer number
            DEF     ERRNO       Error number
            return              Continue execution
              .
              .
              .
RCODE       DEC     24
EFMPF       DEC     4
FNAME       ASC     3,xxxxx
PAKNO       DEC     n           If n = 0, the file is a temporary file on the work area;
                                if n is between 1 and 999, EFMP looks for FNAME in
                                the appropriate area; if n = -1, EFMP searches all
                                available areas for the requested file
RCDNO       DEC     r           If r = 0, EFMP sets the next record to be accessed
                                (for sequential READS or WRITES) to the highest
                                record previously accessed + 1. Otherwise, r can be
                                any number between 1 and the maximum record
                                number contained in the file. This allows sequential
                                access to be initialized at any record.
SCODE       OCT     s           s is the security code established by the CREATE
                                call. It must match.
BUFNO       DEC     b           b must be a number between 1 and the maximum
                                number of Temp. Record Buffers available. For
                                any other number, EFMP uses 1
ERRNO       BSS     1           Return point for error codes
```

CLOSE

Purpose

To remove information about a particular file from the Opened-File Table. This allows an additional file to be opened. Also, CLOSE updates the user status information (USTAT) and the highest record accessed on the disc.

Assembly Language

```
                JSB     EXEC
                DEF     *+6         Return address
                DEF     RCODE       Request code
                DEF     EFMPF       EFMP function number
                DEF     FNAME       File name
                DEF     USTAT       User status
                DEF     ERRNO       Error number
                return              Continue execution
                  .
                  .
                  .
RCODE           DEC     24
EFMPF           DEC     5
FNAME           ASC     3,xxxxx     See Comment 2
USTAT           OCT     u           User status information (any 16-bit combination) to
                                    be written into the disc directory for the file
ERRNO           BSS     1           Return point for error codes
```

Comments

1. If a CLOSE is requested for a temporary file, the directory information in the Opened-File Table is deleted and the Work Area is automatically repacked. If a file has been copied to the Work Area, the user status (USTAT) and highest record assessed are *not* updated on the original copy of the file.

2. To CLOSE all files in the Opened-File Table set the first word of FNAME equal to a binary zero.

READ

Purpose

To retrieve a specified record (random access) or the next record (sequential access) from a file that has previously been opened and written.

Assembly Language

```
                JSB     EXEC
                DEF     *+7         Return address
                DEF     RCODE       Request code
                DEF     EFMPF       EFMP function code
                DEF     FNAME       File name
                DEF     RCDNO       Record number
                DEF     BUFFR       Buffer for data
                DEF     ERRNO       Error number
                return              Continue execution
                  .
                  .
                  .
RCODE           DEC     24
EFMPF           DEC     6
FNAME           ASC     3,xxxx
RCDNO           DEC     n           n is a record number between 1 and 32,767. For
                                    sequential access and backspacing, see Comments.
BUFFR           BSS     m           m is the length of the buffer in words. It must be
                                    at least the record length.
ERRNO           BSS     1           Return point for error codes
```

Comments

If RCDNO = 0, a sequential read or write is implied. This feature provides the program with the next record available relative to the last read or write performed (or OPEN operation). If RCDNO is a negative number, it specifies a backspace, relative to the current record (last record accessed plus 1), before the read or write. If an attempt is made to backspace the record number indicator to a value less than one, the EFMP issues an error and terminates the read or write. Unless needed, care should be taken so as not to backspace the record number indicator beyond the range of records held in the Temporary Record Buffer at that time, since this will initiate a posting operation and a physical disc access.

INITIALIZE

Purpose

To initialize an EFMP area previously created by means of a DOS-III STORE directive.

Assembly Language

```
            JSB     EXEC
            DEF     *+6         Return address
            DEF     RCODE       Request code
            DEF     EFMPF       EFMP function number
            DEF     PAKNO       Pack number
            DEF     DIRSZ       Directory size
            DEF     ERRNO       Error number
            return              Continue execution
              .
              .
              .
RCODE       DEF     24
EFMPF       DEC     7
PAKNO       DEC     p           (1 ≤ p ≤ 999)
DIRSZ       DEC     n           (n = number of entries, one entry/file; see Comment 2)
ERRNO       BSS     1           Return point for error codes
```

Comments

1. Pack number PN000 cannot be initialized.

2. The directory occupies the first sector(s) of the EFMP area.

The number of sectors allocated to a directory is determined as follows:

The variable *n* is used to calculate the number of sectors to be reserved for the directory. It does not indicate the maximum number of file entries allowed in the directory. If the *n*th file entry does not completely fill the last sector of the directory, the space remaining may be used to contain additional file entries.

$$\#\text{Sectors} = \frac{(1 + n) * 9}{128}$$

(add 1 to #Sectors if remainder is > zero)

WRITE

Purpose

To write into a specified record (random access) or into the next record (sequential access) of a file that has previously been opened.

Assembly Language

```
                JSB     EXEC
                DEF     *+7         Return address
                DEF     RCODE       Request code
                DEF     EFMPF       EFMP function number
                DEF     FNAME       File name
                DEF     RCDNO       Record number
                DEF     BUFFR       Buffer for data
                DEF     ERRNO       Error number
                return              Continue execution
                  .
                  .
                  .
RCODE           DEC     24
EFMPF           DEC     8
FNAME           ASC     3,xxxxx
RCDNO           DEC     n           Same as for the READ EXEC CALL
BUFFR           BSS     m           Same as for READ
ERRNO           BSS     1           Return point for error codes
```

RESET

Purpose

To reset the highest record accessed pointer for a file to a lower value. The information beyond the pointer is lost. The file must be open before it can be reset. (PAKNO below provides an additional check.)

Assembly Language

```
                JSB     EXEC
                DEF     *+7
                DEF     RCODE     Request code
                DEF     EFMPF     EFMP function code
                DEF     FNAME     File name
                DEF     PAKNO     Pack number
                DEF     RCDNO     Record number
                DEF     ERRNO     Error number
                return            Continue execution
                  .
                  .
                  .
RCODE           DEC     24
EFMPF           DEC     9
FNAME           ASC     3,xxxxx
PAKNO           DEC     n         If n = 0, EFMP searches the work area to find the
                                  desired file name; if n is a number between 1 and 999,
                                  EFMP searches EFMP area PNn to find the desired
                                  file name; if n = -1, EFMP searches all EFMP areas
RCDNO           DEC     m         m is a number between 0 and 32,767 to which the
                                  highest record accessed pointer will be set ( m must
                                  be lower than the current value)
ERRNO           BSS     1         Return point for error codes
```

STATUS

Purpose

To allow the user program access to various types of status information relative to EFMP. Several separate status functions (identified by unique Status Function Numbers) are provided; all have basically the same form of calling sequence, but they vary in the parameters used.

Assembly Language

```
                JSB     EXEC
                DEF     *+9         Return address
                DEF     RCODE       Request code
                DEF     EFMPF       EFMP function code
                DEF     FSTAT       Status function number
                DEF     FNAME       File name
                DEF     PAKNO       Pack number
                DEF     DUMMY       Not used
                DEF     STATB       Status buffer
                DEF     ERRNO       Error number
                return              Continue execution
                  .
                  .
                  .
                Note:   Above is the general format for Status EFMP calls. The use
                        and meaning of each parameter in the calling sequence varies
                        from status call to status call. The parameters for each call
                        are given separately. below. Common to all status functions
                        are
RCODE           DEC     24
EFMPF           DEC     10
DUMMY           BSS     1
```

STATUS

Purpose

To provide the user with all information, except the security code, contained in the directory for a file.

Parameters

FSTAT	*DEC*	*1*	
FNAME	*ASC*	*3,xxxxx*	
PAKNO	*DEC*	*m*	*If m = 0, EFMP searches the Work Area for the requested file. If m is between 1 and 999, EFMP searches the EFMP area of that pack number. For m = –1, EFMP searches all available EFMP areas for the requested file.*
STATB	*BSS*	*10*	*The pack number is returned in the first word if PAKNO = –1. The remaining nine words will receive the directory status information in the same format as the directory itself (see Figure 7-1).*
ERRNO	*BSS*	*1*	*Return point for error code.*

STATUS

Purpose

To determine if a file is open.

Parameters

FSTAT	*DEC*	*2*	
FNAME	*ASC*	*3,xxxxx*	
PAKNO	*OCT*	*0*	*Not used*
STATB	*BSS*	*2*	*The first word returns the pack number if the file is open. The second word returns a value of 0 if the file is open or 1 if the file is not open.*
ERRNO	*BSS*	*1*	*Return point for error codes.*

STATUS

Purpose

To check the security code of a file.

Parameters

FSTAT	*DEC*	*3*	
FNAME	*ASC*	*3,xxxxx*	
PAKNO	*DEC*	*m*	*Same as function number 1*
STATB	*BSS*	*3*	*The first word returns the pack number if appropriate. The second word is used by the user program to give the security code to be checked. The third word returns 0 if the code checks or 1 if it does not check.*
ERRNO	*BSS*	*1*	*Return point for error codes.*

STATUS

Purpose

To determine the number of available full sectors left between the highest record accessed in a file and the end of the file.

Parameters

FSTAT	*DEC*	*4*	
FNAME	*ASC*	*3,xxxxx*	
PAKNO	*DEC*	*m*	*Same as function number 1*
STATB	*BSS*	*2*	*The first word returns the pack number if appropriate. The second word returns the number of sectors available.*
ERRNO	*BSS*	*1*	*Return point for error codes.*

STATUS

Purpose

To determine the number of available sectors left between the last file in an EFMP area and the end of the EFMP area.

Parameters

FSTAT	*DEC*	*5*	
FNAME	*OCT*	*0*	*Not used*
PAKNO	*DEC*	*m*	*Same as function number 1, but cannot equal –1*
STATB	*BSS*	*2*	*The first word must be present, but is not used. The second word returns the number of sectors available.*
ERRNO	*BSS*	*1*	*Return point for error codes.*

STATUS

Purpose

To obtain the name of the *n*th file in an EFMP area where *n* is an integer between 1 and the maximum number of files in an EFMP area.

Parameters

FSTAT	*DEC*	*6*	
FNAME	*BSS*	*3*	*Return point for file name or all zeroes if no file is present*
PAKNO	*DEC*	*m*	*m is a number between 1 and 999*
STATB	*DEC*	*n*	*n indicates the nth file*
ERRNO	*BSS*	*1*	*Return point for error codes*

STATUS

Purpose

To obtain the name of the nth pack number on a specific subchannel where n is an integer (specifying the ordinal position of the pack number) between 1 and the maximum number of pack numbers on a subchannel.

Parameters

FSTAT	*DEC*	*7*	
FNAME	*DEC*	*m*	*m = the desired subchannel*
.			*On return, FNAME is zero if the EFMP area of*
.			*the pack number is initialized and 1 if the EFMP*
.			*area of the pack number is not initialized.*
.			
.			
PAKNO	*BSS*	*1*	*Return point for the pack number*
STATB	*DEC*	*n*	*n indicates the nth pack number.*
ERRNO	*BSS*	*1*	*Return point for error codes.*

REPACK (PURGE)

Purpose

To repack the existing files on an EFMP area(s), removing empty spaces left when files have been destroyed.

Assembly Language

```
            JSB     EXEC
            DEF     *+5
            DEF     RCODE   Request code
            DEF     EFMPF   EFMP function code
            DEF     PAKNO   Pack number
            DEF     ERRNO   Error number
            return          Continue execution
              .
              .
              .
RCODE       DEC     24
EFMPF       DEC     11
PAKNO       DEC     n       For n between 1 and 999, only the specified EFMP
                            area is repacked; for n = -1, all the EFMP areas
                            available to EFMP are repacked
ERRNO       BSS     1       Return point for error codes
```

CAUTION: IF THE EFMP DISC DIRECTORY CONTAINS A LARGE NUMBER OF FILES AND THE SIZES OF THE TEMPORARY RECORD BUFFERS ARE SMALL, REPACKING MAY REQUIRE CONSIDERABLE TIME. THEREFORE, REPACK SHOULD BE PERFORMED WHEN SUFFICIENT TIME IS AVAILABLE. UNDER NO CIRCUMSTANCES SHOULD AN ABORT BE PERFORMED DURING A REPACK.

COPY

Purpose

To transfer a copy of an opened file and its directory from an EFMP area to the Work Area of DOS-III, from one EFMP area to another EFMP area or from the Work Area to an EFMP area.

Assembly Language

```
            JSB     EXEC
            DEF     *+6
            DEF     RCODE   Request code
            DEF     EFMPF   EFMP function code
            DEF     FNAME   File name
            DEF     PAKNO   Pack number
            DEF     ERRNO   Error number
            return          Continue execution
              .
              .
              .
RCODE       DEF     24
EFMPF       DEC     12
FNAME       ASC     3,xxxxx See Comment 1
PAKNO       DEC     n       If n = 0, EFMP copies the file onto the Work Area;
                            if n is between 1 and 999, EFMP copies the file
                            into the specified EFMP area; if n is between -1
                            and -999, EFMP copies the file from the Work
                            Area to an EFMP area specified by the 10's
                            complement of n (see Comment 2)
ERRNO       BSS     1       Return point for error codes
```

Comments

1. Remember that a file must be opened before it can be copied. This is necessary to determine from which pack to copy the file. When a file has been copied to the Work Area, all reads and writes referencing that file use the Work Area version until the file is closed. (Files copied from the Work Area to an EFMP area continue to use the Work Area version for reads and writes.) Temporary copies of files do not have security codes. Therefore, files copied from the Work Area to a pack have a security code of 0. When a file is copied from pack to pack, the original security code is retained. See "CLOSE" for further notes on Work Area files.

2. If there is already a file with the same name in the destination EFMP area directory, an error code is returned and the copy is aborted. In this case, the user can first destroy the name in the destination EFMP area, and then perform the copy again.

3. When copying from one EFMP area to another EFMP area not on the drive (and only a single removable pack is available), EFMP automatically requests that the user continually swap packs until the entire file has been copied. EFMP outputs:

   INSERT DESTINATION [SOURCE] PACK AND PRESS RUN.

   and halts the computer with 102076 in the DISPLAY register.

   After the user inserts the appropriate pack and presses RUN, a check is made to determine if the proper pack has been entered. If EFMP cannot find the correct pack, the message is repeated. To allow the user an orderly exit in case the correct pack is not available, the following question is asked after each question:

   ENTER C OR T

   where C means to continue copying, and

   T means to terminate the copy and return to the program.

4. Care *must* be taken to insert the original pack (if it has been removed during the copy function) into its original subchannel.

CHANGE FILE NAME

Purpose

To change a file name (file need not be opened).

Assembly Language

```
                JSB     EXEC
                DEF     *+7
                DEF     RCODE   Request code
                DEF     EFMPF   EFMP function code
                DEF     FNAME   File name
                DEF     PAKNO   Pack number
                DEF     SCODE   Security code
                DEF     ERRNO   Error number
                return          Continue execution
                  .
                  .
                  .
RCODE           DEC     24
EFMPF           DEC     13
FNAME           ASC     3,xxxxx Current file name
                ASC     3,zzzzz New file name
PAKNO           DEC     n       n = 0, indicates that the file is on the Work Area;
                                if n is between 1 and 999, n indicates the EFMP
                                area containing the file; if n = -1, EFMP searches
                                all available EFMP areas for the current file name
SCODE           OCT     m       Security code, see CREATE
ERRNO           BSS     1       Return point for error codes
```

POST

Purpose

To physically write on the disc all buffers that have been flagged as "must be written" in the Temporary Record Buffer. (That is, convert all outstanding logical writes into physical writes.)

Assembly Language

```
                JSB     EXEC
                DEF     *+4
                DEF     RCODE   Request code
                DEF     EFMPF   EFMP function code
                DEF     ERRNO   Error number
                return          Continue execution
                  .
                  .
                  .
RCODE           DEC     24
EFMPF           DEC     14
ERRNO           BSS     1       Return point for error codes
```

Comments

The POST operation updates the highest record accessed pointer in the disc directories, but not the user status word (USTAT).

SECTION IX
EFMP Utility Program

The EFMP Utility Program (UTIL) allows the user to access most of the EFMP functions through the keyboard. UTIL accepts commands or directives from the operator and converts these into EFMP calling sequences. After EFMP has processed the call, UTIL reports back (to the operator) a successful operation or an EFMP error.

This section describes how to initiate the UTIL program using the DOS-III PROG directive and then describes the following UTIL commands (presented in alphabetic order):

BRIEF
CHANGE
CLOSE
COPY
CREATE
DESTROY
END
INITIALIZE
OPEN
POST
REPACK
RESET
STATUS-1
STATUS-2
STATUS-3
STATUS-4
STATUS-5
STATUS-6
STATUS-7

All are EFMP functions, except BRIEF and END, which are UTIL program functions.

Note: UTIL requires the FORTRAN IV version of the Formatter program to operate properly.

:PROG,UTIL

Purpose

To initiate execution of the UTIL program.

Format

:PROG, UTIL,n

where $n = 0$ to print a list of commands or
$n \neq 0$ to skip printing the list.

List of commands message (all parameters are decimal):

/INI,PAKNO,DIRSZ
/CRE,FNAME,PAKNO,FLGTH,RLGTH,SCODE,USTAT
/DES,FNAME,PAKNO,SCODE
/OPE,FNAME,PAKNO,RCDNO,SCODE
/CLO,FNAME,USTAT
/RES,FNAME,PAKNO,RCDNO
/STA,DF,FNAME,PAKNO
/STA,FO,FNAME
/STA,SC,FNAME,PAKNO,SCODE
/STA,LR,FNAME,PAKNO
/STA,LF,PAKNO
/STA,NF,PAKNO,STATB
/STA,AP
/REP,PAKNO
/COP,FNAME,PAKNO
/CHA,FNAM1,FNAM2,PAKNO,SCODE
/POS
/BRI,FNAME,SCODE
/END

UTIL begins by outputting a message to indicate that it is ready for a directive:

UTIL READY

After it processes the directive, UTIL outputs the results of the operation (where appropriate) or any error codes that may have been returned by EFMP. When it is ready for another directive, UTIL outputs

UTIL READY

If an incorrect directive is entered, UTIL outputs

ILLEGAL OPERATION
UTIL READY

UTIL is terminated when the operator inputs the command /END.

UTIL outputs any error messages on the system console; normal output is output on the list device.

BRIEF

Purpose

To increase or decrease the amount of disc storage reserved for a file. BRIEF is a UTIL program function, not an EFMP function.

Format

/BRI, fname,scode

fname is the name of the file, and
scode is the security code of the file.

BRIEF first outputs the status of the file:

AVAILABLE RECS. = m *RECORDS USED = r*
NEW RECORD COUNT?

The operator inputs either:

/E to terminate the command and prepare UTIL for more commands,
or
n to change the available record count to *n*

BRIEF stores the contents of *fname* on the Work Area, destroys the current file, repacks the EFMP area, and creates and opens a new file. The contents of *fname* are transferred from the Work Area to the new file and BRIEF prints out a message:

AVAILABLE RECS. = n *RECORDS USED = r*

BRIEF then terminates.

Comment

BRIEF creates and uses a temporary file named "ΔΔΔΔΔΔ" (all blanks).

CHANGE

Purpose

To change the name of a file (i.e., to invoke the CHANGE FILE NAME function of EFMP).

Format

/CHA,fnam1,fnam2,pakno,scode

fnam1 is the current file name
fnam2 is the new file name.

See CHANGE FILE NAME EFMP CALL for explanation of other parameters.

EXAMPLE

/CHA,LOB70,XXXXX,120,0

Example print-out:

```
FILE LOB70 OLD FILE
FILE XXXXX NEW FILE
THE FILE IS ON PACK#   120
THE SECURITY CODE IS   0
```

CLOSE

Purpose

To close a previously opened file (i.e., to invoke the CLOSE function of EFMP).

Format

/CLO,fname,ustat

See CLOSE EFMP CALL for explanation of parameters. Note, however, that all the files in the Opened-File Table *cannot* be closed by setting the first word of FNAME (in the CLOSE calling sequence) to a binary zero.

EXAMPLE

/CLO,LOB70,0

Example print-out:

```
FILE LOB70        CLOSED
THE USER STATUS WORD IS       0
```

COPY

Purpose

To copy a file (i.e., to invoke the COPY function of EFMP).

Format

/COP,fname,pakno

See COPY EFMP CALL for explanation of parameters and messages.

EXAMPLE

/COP,LOB70,120

Example print-out:

```
FILE     LOB70     COPIED
THE FILE IS TEMPORARY IN WORK AREA
FILE     LOB70     COPIED
THE FILE IS ON PACK#     120
```

CREATE

Purpose

To create a new file (i.e., to invoke the CREATE function of EFMP).

Format

/CRE,fname,pakno,flgth,rlgth,scode,ustat

See CREATE EFMP CALL for explanation of parameters.

EXAMPLE

/CRE,C0,120,8,8,0,0

Example print-out:

```
FILE     C0     CREATED
THE FILE IS ON PACK#     120
THE FILE LENGTH IS     8 RECORDS
THE RECORD LENGTH IS     8 WORDS
THE SECURITY CODE IS     0
THE USER STATUS WORD IS     0
```

DESTROY

Purpose

To destroy a file by eliminating its directory entry (i.e., to invoke the DESTROY EFMP function).

Format

/DES,fname,pakno,scode

See DESTROY EFMP CALL for explanation of parameters.

EXAMPLE

/DES,C0,120,0

Example print-out:

FILE C0 DESTROYED

END

Purpose

To terminate the operation of the UTIL program. END is an UTIL program function, not an EFMP function.

Format

/END

INITIALIZE

Purpose

To initialize an EFMP area previously allocated space by means of a DOS-III STORE directive.

Format

/INI,pakno,dirsz

See INITIALIZE EFMP CALL for explanation of parameters.

EXAMPLE

/INI,100,20

Example print-out:

PACK #100 INITIALIZED

OPEN

Purpose

To OPEN a previously CREATED file (i.e., to invoke the OPEN function of EFMP).

Format

/OPE,fname,pakno,rcdno,scode

See OPEN EFMP CALL for explanation of parameters.

EXAMPLE

/OPE,LOB70,120,1,0

Example print-out:

```
FILE     LOB70     OPENED
THE FILE IS ON PACK#     120
THE RECORD # IS     1
THE SECURITY CODE IS     0
```

POST

Purpose

To post files (i.e., to invoke the POST function of EFMP).

Format

/POS

Example print-out:

ALL FILES POSTED

RESET

Purpose

To reset the highest record number accessed for a file (i.e., to invoke the RESET function of EFMP).

Format

/RES,fname,pakno,rcdno

See RESET EFMP CALL for explanation of the parameters.

EXAMPLE

/RES,LOB70,120,0

Example print-out:

```
FILE     LOB70     RESET
THE FILE IS ON PACK#     120
THE RECORD # IS     0
```

REPACK

Purpose

To repack existing EFMP areas (i.e., to invoke the REPACK EXEC CALL function of EFMP).

Format

/REP,pakno

See REPACK EFMP CALL for explanation of parameters.

EXAMPLES

/REP,42 (repacks EFMP area in pack 42)
/REP, –1 (repacks all EFMP areas)

Example print-out:

PACK # 42 REPACKED

or

ALL PACKS AVAILABLE REPACKED

STATUS-1

Purpose

To print out directory information about a file (i.e., to invoke STATUS function number 1 of EFMP).

Format

/STA,DF,fname,pakno

See STATUS EFMP CALL (FSTAT = 1) for explanation of the parameters and results.

EXAMPLE

/STA,DF,LOB70,120

Example print-out:

```
FILE     LOB70     STATUS
THE FILE IS ON PACK#      120
STARTING TRACK # IS      6
STARTING SECTOR # IS      9
THE FILE LENGTH IS      12 RECORDS
THE RECORD LENGTH IS      128 WORDS
THE USER STATUS WORD IS      0
HIGHEST RECORD # ACCESSED IS      0
```

STATUS-2

Purpose

To determine if a file is OPEN (i.e., to invoke STATUS function number 2 of EFMP).

Format

/STA,FO,fname

See FSTAT = 2 for explanation of the parameters and results.

EXAMPLE

/STA,FO,LOB70

Example print-out:

```
FILE LOB70     STATUS
FILE IS OPEN
```

STATUS-3

Purpose

To check the security code of a file (i.e., to invoke STATUS function number 3 of EFMP).

Format

/STA,SC,fname,pakno,scode

See FSTAT=3 for explanation of parameters and results.

EXAMPLE

/STA,SC,LOB70,120,0

Example print-out:

```
FILE     LOB70     STATUS
THE FILE IS ON PACK#      120
THE SECURITY CODE IS      0
CODE CHECKS
```

Note: The security code returned is a restatement of the security code entered; it is not necessarily the correct security code.

STATUS-4

Purpose

To determine the number of available full sectors left between the highest record accessed in a file and the end of the file (i.e., to invoke STATUS function number 4 of EFMP).

Format

/STA,LR,fname,pakno

See FSTAT=4 for explanation of parameters and results

EXAMPLE

/STA,LR,LOB70,120

Example print-out:

```
FILE     LOB70     STATUS
THE FILE IS ON PACK#     120
# OF AVAILABLE SECTORS IS     12
```

STATUS-5

Purpose

To determine the number of available sectors left between the last file in an EFMP area and the end of the EFMP area (i.e., to invoke STATUS function number 5 of EFMP).

Format

/STA,LF,pakno

See FSTAT=5 for explanation of parameters and results.

EXAMPLE

/STA,LF,120

Example print-out:

```
FOR PACK#     120
# OF AVAILABLE SECTORS IS     4610
```

STATUS-6

Purpose

To obtain the name of the *n*th file in an EFMP area where *n* is an integer between 1 and the maximum number of files in an EFMP area (i.e., to invoke STATUS function number 6 of EFMP).

Format

/STA,NF,pakno,statb

See FSTAT=6 for explanation of parameters and results.

EXAMPLE

/STA,NF,120,1

Example print-out:

```
FILE    LOB70    STATUS
THE FILE IS ON PACK#    120
FILE #    1 IN THE DIRECTORY
```

STATUS-7

Purpose

To obtain the name of the *n*th pack number on a specific subchannel where *n* is an integer (specifying the ordinal position of the pack number) between 1 and the maximum number of pack numbers on a subchannel.

Format

/STA,AP,subch,statb

See FSTAT = 7 for explanation of parameters and results.

EXAMPLE

/STA,AP,1,1

Example print-out:

PACK #120 IS AVAILABLE AND INITIALIZED

PART 3
Generating and Loading DOS-III

SECTION X
Generating DOS-III

HP 24307B DOS-III Disc Operating System software must be generated and then loaded into the computer's memory before DOS-III system operation is possible. Generating a DOS-III system consists of two operations:

1. Configuring the system to the available hardware.
2. Storing the configured system on disc memory.

In addition, the discs included in the system must be formatted before they can be used by DOS-III.

This section describes the procedures required to format a disc and to generate DOS-III system software. Both disc formatting and system generation are performed using a stand-alone program, DSGEN.

Depending on the type of moving-head disc device selected for the DOS-III system, generation can be performed either from relocatable modules and drivers punched on paper tape or contained on a master disc cartridge. Systems including an HP 7901, HP 2883, or HP 2884 disc device initially must be generated from paper tape. Systems including an HP 7900 disc device are delivered with a master disc cartridge (HP part number 24307-13001) labeled DSGEN. The DSGEN disc cartridge contains a DOS-III software system together with a set of relocatable modules and drivers. The cartridge may be used to generate DOS-III software. A procedure for preparing to generate DOS-III software from the DSGEN disc cartridge is described later in this section (see "DSGEN Disc Cartridge System Generation").

DSGEN

DSGEN (the DOS-III System Generator) is an absolute program which is loaded into main memory: 1) by the paper tape portion of the main-memory loader, BMDL, when using an HP 2100A/S computer or, 2) by the Bootstrap loaders contained in either the paper tape loader ROM or in the disc loader ROM when using an HP 21MX computer. Since DSGEN input/output is independent of the DOS-III system it generates, the I/O operation of DSGEN requires SIO drivers which are distributed with the DOS-III software. The SIO drivers must be configured to the user's hardware configuration. A copy of the configured DSGEN program can be punched on paper tape using SIO System Dump, if desired. SIO drivers and SIO System Dump are absolute programs — not part of DOS-III — needed only for DSGEN operation. An optional utility program which uses SIO drivers is the Prepare Tape System (PTS). PTS can be used to transfer relocatable modules from paper tape to magnetic tape to expedite the DSGEN program input phase. DSGEN has two independent functions:

1. To format new disc cartridges (or packs).

2. To generate a DOS-III software system that fits the user's main-memory size, I/O equipment, and programming needs.

DSGEN CONFIGURATION FROM PAPER TAPE

DSGEN is executed in a Software Input/Output environment to generate DOS-III. First, ensure that equipment power is on and disc storage is unprotected (Disc Protect Override or Format enabled). At this point in DSGEN configuration, the procedure for loading paper tape depends on the computer being used.

HP 2100A/S

The main-memory loader, BMDL, is used to load programs from paper tape into memory. BMDL is described in detail in Section XI. A simplified procedure follows:

A. Place the paper tape into the paper tape reader and press READ to ready the reader.

B. On the computer front panel, set the P-register to the BMDL starting address 37700_8 for 16K words of memory; 57700_8 for 24K words; or 77700_8 for 32K words.

C. Press PRESET (INTERNAL and EXTERNAL); then press RUN. After a successful load, the computer will halt with 102077_8 in the display register.

HP 21MX

The HP 21MX processor is equipped with a paper tape loader ROM, the contents of which are equivalent to the Basic Binary Loader portion of the BMDL used on HP 2100A/S computers. The contents of the ROM must be loaded into memory before the drivers or DSGEN (or any program on paper tape) can be placed into main memory. Use the following procedure to accomplish paper tape ROM loading.

A. Press PRESET.

B. Select the S-register for display.

C. Press the CLEAR DISPLAY.

D. Bits 15 and 14 of the Display Register must be 00 to select the paper tape loader ROM.

E. Change bits 11 through 6 of the Display Register to the octal select code of the tape reader.

   Since bits 13, 12, and 5 through 0 are not used in conjunction with the paper tape loader, they are ignored.

F. Press STORE to store the contents of the Display Register in the S-register.

G. Press IBL to load the contents of the paper tape loader ROM into the uppermost 64 locations in memory. The computer halts with 102077_8 in the T-register.

H. Place the DSGEN paper tape in the paper tape reader, press READ to ready the reader, and press RUN at the main processor. After a successful load, the computer halts with 102077_8 in the T-register.

To configure DSGEN (using either an HP 2100A/S or HP 21MX computer), proceed as follows:

1. Specific SIO drivers must be configured before DSGEN can be executed. To configure a driver:

   a. Load the driver program into memory via the paper tape reader using the proper set of procedures from those described above (*HP 2100A/S* Steps A through C or *HP 21MX* Steps A through H).

   b. Set the I/O channel select code of the device (lower numbered select code if there are two I/O channels) in bits 5-0 of the switch register.

   c. Start the driver program by setting the P-register to address 2_8 ; then press RUN. Upon successful completion of the driver configuration, the computer will halt with 102077_8 in the display register.

2. Configure the SIO console driver (HP part no. 24127-60001) using Steps 1-a through 1-c. (If the console device is an HP 2754B teleprinter, switch register bit 15 must be set to one at Step 1-b.)

3. If program input is to be from the paper tape reader, configure the SIO paper tape reader driver (HP part no. 20319-60001) using Steps 1-a through 1-c.

4. If a high-speed paper tape punch is included in the system, configure the SIO punch driver (HP part no. 20320-60001) using Steps 1-a through 1-c.

5. Load DSGEN via the paper tape reader using the appropriate procedure described above (*HP 2100A/S* Steps A through C or *HP 21MX* Steps A through H).

6. If program input is to be from magnetic tape, configure the SIO magnetic tape driver (HP part no. 13022-60001) using Steps 1-a through 1-c.

7. If the system includes a high-speed or console punch, a configured DSGEN can be punched on paper tape using the following procedure:

   a. Load the SIO System Dump program (HP part no. 20335-60001) via the paper tape reader using the procedure described in *HP 2100A/S* Steps A through C or the procedure described in *HP 21MX* Steps A through H.

   b. Set switch register bit 15 to one.

   c. Start the SIO System Dump program by setting the P-register to address 2_8 ; then press RUN. After tape punching is successfully completed, the computer will halt with 102077_8 in the display register. For an additional copy of the configured DSGEN, press RUN.

8. If the disc or discs to be used by DOS-III have been formatted, DOS-III system generation can begin immediately. Proceed as follows:

   a. Set switch register bit 15 to zero.

   b. Set the P-register to DSGEN starting address 100_8.

   c. Press RUN. DOS-III system generation dialog begins (see "Using DSGEN to generate DOS-III").

9. To format discs before executing system generation:

   a. Set switch register bit 15 to one.

   b. Set the P-register to DSGEN starting address 100_8.

   c. Press RUN. The disc formatting dialog begins (see "Using DSGEN to Format Discs").

DSGEN Start-up

To start either disc formatting or DOS-III system generation from a configured DSGEN program (on paper tape) perform a standard paper-tape load. These procedures are described in *HP 2100A/S* Steps A through C or in *HP 21MX* Steps A through H. Then proceed as follows:

- For disc formatting:

  a. Set switch register bit 15 to one.

  b. Set the P-register to the DSGEN starting address 100_8.

  c. Press RUN. The disc formatting dialog begins (see "Using DSGEN to Format Discs").

- For DOS-III system generation:

  a. Set switch register bit 15 to zero.

  b. Set the P-register to the DSGEN starting address 100_8.

  c. Press RUN. DOS-III system generation begins at the initialization phase (see "Using DSGEN to generate DOS-III").

USING DSGEN TO FORMAT DISCS

Before a fresh disc can be used in DOS-III, it must be formatted by DSGEN. System discs (including a possible User Area) are formatted during system generation, but dedicated user discs must be formatted by running DSGEN again in a special mode. Formatting a disc involves assigning it a system generation code, reading every sector, clearing any existing user or system directory, and so forth. The result is an unlabeled user disc ready for use in DOS-III. The following operator responses are only examples, actual responses should be appropriate to the particular system being generated.

Operating Instructions

1. Turn on all equipment.
2. Unprotect the disc (enable Disc Protect Override).
3. Load a configured DSGEN using the main-memory resident BMDL or the paper tape loader ROM. (See "DSGEN Configuration and Start-up" in this section.)
4. Set up a starting address at location 100_8.
5. Set switch register bit 15 equal to 1.
6. Start the computer executing (press RUN).
7. 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 1- to 4-digit decimal number . . . . . ~~79~~ 4DIGIT
8. DSGEN requests the octal channel number (select code) of the disc controller . . . . . *SYS DISC CHANNEL?*

   Operator responds with the appropriate octal number *10* 15

   CSW DISC CHANNEL 15
9. DSGEN requests the type of disc storage . . . . . *DISC TYPE?*

   Operator responds with 7900, 7901, 2883, or 2883B (A response of 2883 implies four subchannels per disc drive; 2883B implies two subchannels per disc drive.) *7900* 7900
10. 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 0

11. DSGEN requests that the disc be unprotected (if it is still protected) . . . . . *TURN ON DISC PROTECT OVERRIDE — PRESS RUN*

    Operator unprotects the disc and starts the computer executing.

12. DSGEN carries out formatting on the specified subchannel and halts with a code of 102007_8.

13. This procedure should be repeated for each proposed user disc.

    Operator can start the computer (press RUN) to format a new disc of the same type (switch bit 15 must still be equal to 1).
    DSGEN repeats from . . . . . *USER DISC SUBCHANNEL?*

    Operator can set switch bit 15 equal to 0 and start the computer (press RUN) to proceed to system generation.

USING DSGEN TO GENERATE DOS-III

The operation of DSGEN involves four phases:

1. INITIALIZATION PHASE. DSGEN requests specifications for DOS-III, including description of available disc space, memory, Time-base Generator channel, system generation code, system and user disc subchannels, and program input devices.

2. PROGRAM INPUT PHASE. DSGEN reads the relocatable programs to be included in the system. The relocatable program modules can be input via paper tape, disc, or magnetic tape (the magnetic tape must be prepared off-line using the Prepare Tape System).

3. PARAMETER INPUT PHASE. Parameters to change EXEC modules or drivers from disc- to main-memory resident may be entered. The programs' NAM records are already set for a minimum main-memory system except that two console drivers have been included. DISCM, $EX30 (if EFMP is used), moving-head driver DVR31, and *one* console driver must be main-memory resident.

4. DISC LOADING PHASE. DSGEN requests a specification of the base page linkage, and begins loading programs onto the disc in absolute format. Systems programs (i.e., the modules of DOS-III) 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.

Restart

If an error occurs during execution of any phase, the operator can restart that phase by restarting DSGEN at location 100_8.

Initialization Phase

During the initialization phase, DSGEN requests information necessary to begin generating the DOS-III. After each output on the system console, the operator responds by entering 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 with a 1- to 4-digit decimal integer . . . . . 79 4 DIGIT

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

   Operator responds with the high priority (low number) channel 14 15

   Note: BMDL requires that the SYS DISC CHNL? response must be the same value as the EQT entry for the system. GEN DISC 15

3. DSGEN requests the type of disc storage . . . . . *DISC TYPE?*

   Operator responds with 7900, 7901, 2883, or 2883B. A response of 2883 implies four subchannels per disc drive; 2883B implies two subchannels per disc drive. *7900*

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* 200

5. DSGEN requests the number of drives on the system . . . . . *# DRIVES?*

   If response to Step 3 was 2883, the operator responds with 1 or 2; if response to Step 3 was 2883B, 7900, or 7901, the operator responds with a number between 1 and 4 inclusive. 1

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

   Operator responds *0* 0

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

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

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

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

Note: On a 7901 disc, only odd numbered subchannels are available.

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.) *0*

10. DSGEN requests the octal channel number (select code) of the Time-base Generator . . . . . *TIME BASE GEN CHNL?*

    Operator responds with the proper select code or 0 if the Time-base Generator is not present *0*

    DSGEN now requests the select code of the privileged-interrupt card . . . . . *PRIV INT CARD CHNL?*

    Operator responds with the channel (octal) of the privileged interrupt fence if privileged interrupt is desired; otherwise, type 0. . . . . *0*

11. DSGEN requests the number of DMA channels in the system . . . . . *# DMA CHANNELS?*

    Operator responds with the number of DMA channels available *2*

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

    Operator responds . . . . . ~~*27077*~~

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 primary input unit for relocatable program modules . . . . . *PRGM INPT?*

    Operator responds with PT (for paper tape), TY (for teleprinter), DF (for disc file), or MT (for magnetic tape; see *PREPARE TAPE SYSTEM (02116-91751)*) . . . . . *DF*

15. If the previous answer is DF, DSGEN requests the subchannel number of the disc containing the relocatable program modules . . . . . *INPUT DISC SUBCHNL?*

1 -(INFO FROM DISC 2)

Operator responds with the appropriate subchannel number. The subchannel must contain a disc (prepared by a pre-existing DOS-III) whose user area contains only relocatable modules of DOS-III. By specifying PT to the next question (LIBR INPT?) the operator can include programs from the paper tape reader in addition to those on the disc file 4

16. DSGEN requests the type of optional input unit for relocatable program modules . . . . . *LIBR INPT?*

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

Note: Any type of relocatable program can be entered through the Program Input Unit or the Library Input Unit.

17. DSGEN requests the type of input unit for the parameter input phase . . . . . *PRAM INPT?*

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

(ALWAYS FROM TY)

When DSGEN finishes the initialization phase, the computer halts.

NOW LOADING OTHER RELOCATIBLES FROM DISC 1

PROGAM INPUT S Reg to 00 HIT RUN

LIBRARY INPUT S Reg to 10 HIT RUN

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 0-1 (00_2 for the Program Input Unit, or 10_2 for the Library Input Unit), and places the programs in the input device. Main programs must be entered prior to their segments. DISCM should be the first module loaded.

DISC MOUNTED TAPES TO DISC SO HAS TO BE FIRST - THEN DRIVERS

The suggested order of module input is

- DOS-III MAIN-MEMORY RESIDENT SYSTEM (DISCM)
- DOS-III I/O DRIVERS (DVR05, DVR01, DVR71, . . . ETC)
- DOS-III EXEC MODULES ($EX01 . . .)
- EFMP EXEC MODULES (IF DESIRED-$EX30 . . .)
- DOS-III JOB PROCESSOR/FILE MANAGER (JOBPR)
- DOS-III RELOCATING LOADER (LOADR)
- DOS-III ASSEMBLER (MAIN CONTROL, SEGMENTD, SEGMENT1, . . .)
- DOS-III FORTRAN (MAIN CONTROL, PASS1, . . .)
- DOS-III EFMP UTIL (IF $EX30 . . . AND FORTRAN IV LIBRARY ARE INCLUDED)
- RTE/DOS ALGOL
- RTE/DOS FORTRAN IV LIBRARY OR RTE/DOS BASIC FORMATTER
- RTE/DOS RELOCATABLE PROGRAM LIBRARY (EAU OR FLOATING POINT) — See Note 1
- RTE/DOS FAST FORTRAN PROCESSOR (FFP) SUBROUTINE LIBRARY AND $SETP SYSTEM SUBROUTINE — See Note 2
- Any relocatable user programs to be made a permanent part of DOS-III

Notes: 1) For systems based on an HP 21MX series processor only, the HP 21MX RTE/DOS DEBUG subroutine should be loaded immediately following the Relocatable Program Library (see "Special Considerations" under "DEBUG Library Subroutine" in Section V).

2) When the FFP and $SETP tapes are loaded, ERR08 and ERR05 will occur and messages will be printed on the console because the entry point names used by these subroutines replace the FORTRAN and library subroutine entry point names.

Load the first input module and start the computer executing. When entering paper tape, the message "*EOT" is output whenever an end-of-tape occurs. The computer halts. Program input can be switched back and forth between the input units by varying the switch register bits between 00_2 and 10_2 before starting the computer.

To terminate the program input phase, the operator must set switch register bits to 01_2, and start the computer. If there are no undefined externals, this message is printed on the system console:

NO UNDEF EXTS

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

UNDEF EXTS

The externals are listed one per line and the computer halts. External references are satisfied by loading more programs. The operator must set switch register bits to 00_2 (for Program Input Unit) or 10_2 (for the Library Input Unit) and start the computer executing. If the externals are to be left unsatisfied, set the switch register bits to 01_2 and start the computer executing.

Note: $EX30 through $EX33 (the EFMP EXEC modules) and $EX36 and $EX37 (user EXEC modules) are not listed when missing.

Parameter Input Phase

During the parameter input phase, the operator can change selected I/O drivers and EXEC modules from disc-memory to main-memory resident or vice versa. In addition, an optional parameter allows the operator to change the linking mode for each module. Either current page or base page linking can be selected.

The console driver must always be main-memory resident. Console drivers DVR00 and DVR05 are distributed as main-memory resident while terminal printer driver DVR26 is distributed as disc-resident. The console model to be used in the configured system determines which driver must be main-memory resident. Any unnecessary I/O drivers *must* be eliminated at this time. If the memory management capability is not desired, delete modules $EX22 and $$MGT from the system by specifying them as type 8 (see below).

DVR31, DISCM, and $EX30 are distributed as main-memory resident modules; they must not be changed to disc-resident.

Each parameter record has the form:

name,type[,link mode]

where *name* is the name of the program to be changed.

type is the program type code:
- 0 — System main-memory 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 is deleted from the system

link mode is the mode of linking to be performed:
- 0 — current page linking (default)
- non-zero — base page linking

When changing the linking mode, the program *type* must be specified. An error in either the *type* or *link mode* parameter results in an error message (ERR10).

The following modules are designed to execute with base page linking and must not be changed to current page linking mode:

Program	Module Name
HP ALGOL	ALGOL
HP Assembler	ASMB
HP FORTRAN	FTN
HP FORTRAN IV	.FTN4 (4K area)
	FTN4 (10K area)
HP DOS-III Job Processor	JOBPR

For programs changed to current page linking mode, the programs should be structured into subroutines of less than 2048 words (two pages of memory) in length. Current page links are generated only at the beginning and end of a program. They cannot be inserted into the program area because the boundary between program code and current page links might occur within a skip or jump sequence. If a program spans more than two pages, there is no area available for current page links in the middle pages, so base page links will be used; thus, the potential for greater efficiency is lost.

Parameter input is terminated by entering the slash character followed by the letter E (/E). This ends the parameter input phase.

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

Module Name	Request Codes	Function
$EX01	16	Disc work tracks status
$EX02	17	Disc work tracks limits
$EX03	6	Program completion
$EX04	7	Program suspension and associated messages
$EX05	8,10	Program main or segment search
		(Note: $EX05 calls $EX10)
$EX06	18	User file name search
$EX07	11	Current time processor
$EX08	4 (RT)	Real-time disc allocation
$EX09		:EQ processor
$EX10	8,10	Load and execute main program or segment
		(Note: see also $EX05)
$EX11	14,15	System file name search
		(Note: used for file read/write)
$EX12		System startup

Module Name	Request Codes	Function
$EX13		Error message processor
$EX14		:UP, :DN, :LU processor
$EX15		Abort and post-mortem dump
$EX16		:GO parameter processor
$EX17	23	:UD processor
$EX18	1,2,3, 14,15	I/O initiation processor *(Note: See also $EX11)*
$EX19		:IN processor
$EX20		Disc parity processor
$EX21	32,33,34	Programmatic file control
$EX22	35,36,38,41	Memory management
$EX36	27	User written module
$EX37	28	User written module

Functions of EFMP EXEC Modules

$EX30	—	Always main-memory resident (common routines and values).
$EX31	—	DEFINE, CREATE, DESTROY, OPEN, CLOSE
$EX32	—	READ, WRITE, RESET, STATUS, CHANGE
$EX33	—	COPY, REPACK

When changing program types, it is not necessary to explicitly specify all subroutines called by an EXEC module which is made main-memory resident. The generator automatically makes the proper linkages. In addition to making the subroutine main memory resident, the generator places it in the system library, thus making it available to user programs.

Disc Loading Phase

1. DSGEN asks for the number of base page links . . . . . *# LINKS?*

   The operator responds with the decimal number of links. If the operator responds with a blank character, DSGEN allocates the maximum number of links (800) *540*

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 and ending locations of both main memory and base page portions of the program. In addition, if bit 15 is set (ON), the entry points for main programs and subroutines are printed. (Subroutines are indented two spaces, and entry point addresses are preceded by an asterisk.)

2. DSGEN requests memory management subsystem names . . *ENTER SUBSYSTEM NAMES*

   The operator responds with a series of one line entries which specify the subsystem name (1-4 characters) of each subsystem that utilizes memory management (see :MMGT directive). Terminate the input list with the characters "/E" *SUB3*
   SUB7
   /E

   Note: 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 ENTRY*

   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 . . . . . *nn,DVRnn[,D] [,u]*

   where *nn* is the octal channel number (lower number if multi-board, maximum is 37_8)

   DVRnn is the driver name (*nn* is the equipment type code)

   D, if present, means DMA channel required

   u is the physical subchannel (unit) number (valid responses; 0-31)

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

   Here is a sample Equipment Table:

```
* EQUIPMENT TABLE ENTRY
10,DVR31,D      (EQT entry #1 = disc)
12,DVR23,D      (EQT entry #2 = magnetic tape)
14,DVR05        (EQT entry #3 = system console)
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 console on channel 14, EQT #3)
3
 2   = EQT #?      (Disc on channel 10, EQT #1)
1
 3   = EQT #?      (Disc on channel 10, EQT #1—reserved for system use)
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 0 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 channel which may interrupt, in ascending order and in the format . . . . . n_1,*option*

where n_1 is the octal channel number (high number if multi-board) between 10_8 and 37_8 inclusive (must be entered in ascending order)

option directs the system in handling the interrupt:

EQT,n_2 relates the channel to EQT entry number n_2,

ABS, *value* places an absolute octal value in the interrupt location. *value* is an octal integer.

ENT,*entry* transfers control to the entry point of a user-written system program upon interrupt (typically the P.*xx* entry of a privileged I/O driver).

If 2883B was specified as the disc type (see "Initialization Phase," step 3) and a second controller is added, the octal channel number of both controllers must be specified.

The operator terminates entry by typing . . . . . */E*

Here is a sample Interrupt Table:

```
*INTERRUPT TABLE
10,ENT,P.73      (Channel 10 linked to privileged interrupt routine P.73)
12,EQT,1         (Channel 12 linked to EQT #1)
13,ABS,102077    (Channel 13 interrupt location filled with an octal halt instruction)
14,EQT,4         (Channel 14 linked to EQT #4)
15,EQT,5         (Channel 15 linked to EQT #5)
16,ABS,0         (Channel 16 interrupt location filled with a NOP; all zeros)
/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 octal address plus 1 of the system base page link area . . . . . *LWA LINKS yyyyy* 0066

7. DSGEN requests the first word base page octal address of the user link area . . . . . *FWA USER LINKS?* 0067

   Operator responds with an octal address greater than or equal to *yyyyy* and less than 2000_8 . . . . . *mmmmm*

8. DSGEN reports the last octal address plus 1 of the main-memory resident system . . . . . *LWA PROG xxxxx* 23707

9. DSGEN requests the octal address of the first word of the user program area . . . . . *FWA USER?* 24000

   Operator responds with an octal address greater than or equal to *xxxxx*. (This option is provided so that user programs can start on a page boundary, if desired) . . . . . *nnnnn*

   Note: Some system programs must be base page linked, i.e., the FORTRAN compiler. For this reason it is recommended that the User Area always be started on a page boundary. Since pages contain 2000_8 words, the octal number specified should be some multiple of 2000_8.

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

10. When system generation is complete, DSGEN reports . . . ** SYSTEM STORED ON DISC*

11. Protect the disc (enable Disc Protect) to prevent access to the system portion of the disc.

12. The DOS-III system which has just been generated (in this case, on Subchannel 0) must be loaded into main memory. This is accomplished by using BMDL or the disc loader ROM. (See Section XI).

    Note: If a configured DOS-III system resides on a disc starting at head 0, track 0, simply press RUN. The system will execute and halt with 102077_8 in the Memory Data register. Then, set the switch register to the subchannel of the newly generated system (in this case, Subchannel 0), press PRESET (INTERNAL and EXTERNAL) and press RUN. The newly generated DOS-III system will be automatically loaded into memory.

```
SYS GEN CODE?
0103

SYS DISC CHNL?
15

DISC TYPE?
7900

SYS DISC SIZE?
200

# DRIVES?
2

FIRST SYSTEM TRACK?
0
FIRST SYSTEM SECTOR?
3

SYS DISC SUBCHNL?
3

USER DISC SUBCHNL?
3

TIME BASE GEN CHNL?
14

PRIV INT CARD CHNL?
11

# DMA CHANNELS?
2

LWA MEM?
77677

ALLOW :SS?
YES

PRGM INPT?
DF
INPUT DISC SUBCHNL?
1

LIBR INPT?
PT
```

```
PRAM INPT?
TY

*EOT

 NO UNDEF EXTS

ENTER PROG PARAMETERS

$EX18,Ø
/E

# LINKS?
8ØØ

SYSTEM

NAME        PROG BOUNDS    BP BOUNDS

DISCM
  $TIME     Ø5231          ØØ574
  $SETP     Ø5362          ØØ574
  $PFAL     Ø5403          ØØ574

(BOUNDS)    Ø2ØØØ Ø54Ø6    ØØ337 ØØ574

DVR31

(BOUNDS)    Ø54Ø6 Ø6127    ØØ574 ØØ635

F4D.C

(BOUNDS)    Ø6127 Ø6127    ØØ635 ØØ635

F2F.B

(BOUNDS)    Ø6127 Ø6127    ØØ635 ØØ635
```

```
DVR00

(BOUNDS)    06127 06600    00635 00637

DVR70

(BOUNDS)    06600 07022    00637 00642

$EX18

(BOUNDS)    07022 07712    00642 00642

$$MGT

(BOUNDS)    07712 10523    00642 00667


ENTER SUBSYSTEM NAMES

/E

* EQUIPMENT TABLE ENTRY

10,DVR70
12,DVR00


13,DVR01
15,DVR31,D
17,DVR12
20,DVR11,D
21,DVR23,D
23,DVR02
/E
```

```
* DEVICE REFERENCE TABLE

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


* INTERRUPT TABLE

10,ENT,P.70
12,EQT,2
13,EQT,3
16,EQT,4
17,EQT,5
20,EQT,6
22,EQT,7
23,EQT,8
/E


EXEC SUPERVISOR MODULES

NAME        PROG BOUNDS    BP BOUNDS

$EX21
  $SRCH     11734          00670

(BOUNDS)    11034 12375    00667 00713
```

```
$EX01
  $ADDR     11121          00670

(BOUNDS)    11034 11136    00667 00670


$EX02
  $ADDR     11124          00670

(BOUNDS)    11034 11141    00667 00670


$EX03

(BOUNDS)    11034 11105    00667 00667


$EX04
  ASCII     11426          00671

(BOUNDS)    11034 11550    00667 00671


$EX05
  $SRCH     11117          00670

(BOUNDS)    11034 11560    00667 00670


$EX06
  $SRCH     11136          00671
  $ADDR     11577          00671

(BOUNDS)    11034 11614    00667 00671


$EX07
  $ADDR     11221          00670

(BOUNDS)    11034 11236    00667 00670
      .

$EX08
  $ADDR     11207          00670

(BOUNDS)    11034 11224    00667 00670
```

```
$EX09
  ASCII     11433          00671

(BOUNDS)    11034 11555    00667 00671

$EX10

(BOUNDS)    11034 11372    00667 00667

$EX11
  $SRCH     11057          00670

(BOUNDS)    11034 11520    00667 00670

$EX12

(BOUNDS)    11034 11320    00667 00667

$EX13
  ASCII     11411          00671

(BOUNDS)    11034 11533    00667 00671

$EX14
  ASCII     11557          00670

(BOUNDS)    11034 11701    00667 00671

$EX15
  ASCII     11403          00670

(BOUNDS)    11034 11525    00667 00671

$EX16

(BOUNDS)    11034 11165    00667 00667
```

```
$EX17
  $LBL      11424          00672

(BOUNDS)    11034 11532    00667 00674

$EX19
  $LBL      11427          00674

(BOUNDS)    11034 11535    00667 00674

$EX20

(BOUNDS)    11034 11520    00667 00667

$EX22

(BOUNDS)    11034 13134    00667 00705


I/O DRIVER MODULES

NAME        PROG BOUNDS    BP BOUNDS

DVR01

(BOUNDS)    13134 13525    00713 00715

DVR02

(BOUNDS)    13134 13361    00713 00715

DVR11

(BOUNDS)    13134 14053    00713 00724

DVR12

(BOUNDS)    13134 13521    00713 00715
```

```
DVR23

(BOUNDS)    13134 13752    00713 00715


LWA LINKS   00724

FWA USER LINKS?
724


LWA PROG    14053

FWA USER?
16000


USER SYSTEM PROGRAMS

NAME        PROG BOUNDS    BP BOUNDS

LOADR
  .EAU.     27501          01422
  DUMRX     27551          01426

(BOUNDS)    16000 27631    00724 01426


ASMB

(BOUNDS)    16000 23131    00724 01303


ASMBD

(BOUNDS)    23131 23741    01303 01304


ASMB1

(BOUNDS)    23131 24553    01303 01347


ASMB2

(BOUNDS)    23131 24570    01303 01331
```

```
ASMB3

(BOUNDS)    23131 24002    01303 01307

ASMB4

(BOUNDS)    23131 24040    01303 01311

ASMB5

(BOUNDS)    23131 24445    01303 01326

XREF
  .OPSY     21230          01013
  DUMRX     21270          01015

(BOUNDS)    16000 21350    00724 01015

FTN4

(BOUNDS)    16000 31170    00724 01272

F4.0

(BOUNDS)    31170 37041    01272 01354

F4.1

(BOUNDS)    31170 34732    01272 01406

F4.2

(BOUNDS)    31170 36260    01272 01370
```

```
XDISC
  .SWCH     20620          01041
  FMTIO     20637          01041
  INDEX     22070          01101
  .PRAM     22246          01101
  EXECX     22356          01101
  INITX     22402          01101
  FLIB      22441          01101
  .FLUN     22544          01107
  .XFER     22565          01107
  DBLE      22631          01110
  SNGL      22666          01112
  FRMTR     22734          01113
  .OPSY     25474          01360
  .EAU.     25534          01360
  DUMRX     25604          01361
  .ZRLB     25664          01361
  .XPAK     25725          01361
  .ENTR     26122          01373
  .PACK     26212          01374
  .XCOM     26326          01374

(BOUNDS)    16000 26377    00724 01374


JOBPR

(BOUNDS)    16000 30422    00724 01401


*SYSTEM STORED ON DISC
```

DSGEN DISC CARTRIDGE SYSTEM GENERATION

Each HP 24307B DOS-III Disc Operating System with an HP 7900 Disc device included in the system hardware is delivered with a disc cartridge labeled DSGEN (HP part number 24307-13001). The DSGEN cartridge contains a DOS-III software system together with a set of modules with which to generate a DOS-III software system in the computer's memory.

Care must be taken to protect the contents of this disc from modification or destruction. The DSGEN cartridge can be copied to another disc and set aside. Modification can now be made to the copy without affecting the original disc.

If modules not included on the DSGEN cartridge are required, they must be loaded into the system from another type of input unit during the system generation procedure.

The I/O PCA boards must be arranged according to the select codes specified by the label on the DSGEN cartridge. For example:

Select Codes

11 7900 DISC
10 SYSTEM CONSOLE

The example indicates that the HP 7900 disc device resides in select codes 11 and 12, and the system console device resides in select code 10.

Initial generation steps differ for an HP 21MX computer (with an optional disc loader ROM installed) and an HP 2100A/S computer. For an HP 21MX not equipped with a disc loader ROM, use HP 2100A/S procedures.

HP 21MX

1. Insert the DSGEN cartridge in the HP 7900 disc device. Press PRESET.
2. Select the S-register for display. Press CLEAR DISPLAY.
3. Set the select code of the disc in bits 11 through 6. Set bit 14 to select the disc loader ROM to be loaded.
4. Press STORE to store the contents of the Display Register in the S-register. Press IBL to load the contents of the disc loader ROM into the uppermost 64 locations of memory.

The computer halts with octal 102077 in the T-register. Press RUN and dialogue with the system begins. Go to step 5.

HP 2100A/S

1. Load and configure the Stand-alone Paper Tape Bootstrap Loader to the system hardware.
2. Insert the DSGEN cartridge in the HP 7900 Disc device.
3. Load DOS-III from Subchannel 1 and initiate it using the Stand-alone Paper Tape Bootstrap Loader. Go to step 5.

Once DOS-III is initiated, a dialogue between the system and the operator begins on the system console. In the following example, information typed by the operator is underlined, and the information printed by the system is not underlined. These underlines will not appear on the terminal under actual operating conditions.

5. The DOS-III system begins the dialog by requesting the DATE directive:

```
INPUT :DATE,XXXXXXXXXX
```

@:DATE,,	DATE directive entered
SUBCHAN=1	
LBL=DSGEN	
@	
:JOB	JOB directive entered
JOB	
@	
:UD,*,0	Change user disc to Subchannel 0, no label
@	
:IN,*	Initialize user disc, no label
@	
:UD,DSGEN,1	Change user disc to Subchannel 1, label is DSGEN
@	
:UD	Verify correct subchannel and label
SUBCHAN=1	
LBL=DSGEN	
@	
:DD	Disc-to-disc dump of disc on Subchannel 1
@	
:UD,*,0	Destination disc for dump operation.

6. Wait for the system to respond with @ to indicate that the entire contents of Subchannel 1 have been copied to Subchannel 0.

7. Press HALT.

8. Remove the DSGEN cartridge from the HP 7900 Disc device.

9. Insert a disc cartridge to be used for subsequent DOS-III system generation.

10. Load DOS-III from Subchannel 0 and initiate it using the Stand-alone Paper Tape Bootstrap Loader.

11. System dialog begins:

```
INPUT :DATE,XXXXXXXXXX
```

@:DATE,,	DATE directive entered
SUBCHAN=0	
LBL=DSGEN	
@	
:JOB	JOB directive entered
JOB ,	
@	
:LIST,S,1,INDEX	List user source file, INDEX on the console (the following list is an example)

```
ØØØ1      DOS III B  (24307B)  REV 1419
ØØØ2      THIS INDEX RELATES THE NAMES OF THE RELOCATABLE MODULES
ØØØ3      TO THE PART NUMBERS OF THE EQUIVALENT PAPER TAPES AND
ØØØ4      INDICATES THE PURPOSE OF THE MODULES IN THE SYSTEM.
ØØØ5      NAME    PART NUMBER    REV    DESCRIPTION
ØØØ6      DISCM   243Ø7-16ØØ2    1419   DISC MONITOR
ØØØ7      $EXMD   243Ø7-16ØØ3    1419   EXEC MODULES
ØØØ8      DVRØØ   2Ø985-6ØØØ1    1419   TTY-LIKE CONSOLE/TERMINAL
ØØØ9      DVRØ1   2Ø987-6ØØØ1    1419   PAPER TAPE READER
ØØ1Ø      DVRØ2   2Ø989-6ØØØ1    1419   PAPER TAPE PUNCH
ØØ11      DVRØ5   24157-6ØØØ1    1419   TTY-LIKE CONSOLE
ØØ12      D2892   24272-6ØØØ1    1419   2892B CARD READER (DVR11)
ØØ13      D2767   24168-6ØØØ1    1419   2767A LINE PRINTER (DVR12)
ØØ14      D261Ø   24271-6ØØØ1    1419   261ØA/2614A LINE PRINTER (DVR12)
ØØ15      D26Ø7   24349-6ØØØ1    1419   26Ø7A LINE PRINTER (DVR12)
ØØ16      DVR23   13Ø24-6ØØØ1    1419   797ØB/E MAG TAPE
ØØ17      DVR26   24333-6ØØØ1    1419   2762A CONSOLE PRINTER
ØØ18      DVR31   24156-6ØØØ1    1419   79ØØ/79Ø1/287Ø DISC
ØØ19      DVR67   24341-16ØØ1    1419   12889A HS SERIAL IF
ØØ2Ø      DVR72   2435Ø-16ØØ1    1419   12587B ASYNC DATA SET IF
ØØ21      DVR73   24377-16ØØ1    1419   1292ØA/B MUX
ØØ22      EFMP    243Ø9-6ØØØ2    1419   EXT FILE MGR EXEC MODULES
ØØ23              243Ø9-6ØØØ3    1419   EXT FILE MGR UTILITIES
ØØ24      JOBPR   243Ø7-16ØØ4    1419   JOB PROCESSOR
ØØ25      RLODR   243Ø8-6ØØØ1    1419   RELOCATING-LINKING LOADER
ØØ26      ASMB    24158-6ØØØ1    B      ASSEMBLER
ØØ27              24158-6ØØØ2    B
ØØ28              24158-6ØØØ3    B
ØØ29              24158-6ØØØ4    B
ØØ3Ø              24158-6ØØØ5    B
ØØ31              24158-6ØØØ6    B
ØØ32              24158-6ØØØ7    B
ØØ33      .FTN4   2417Ø-6ØØØ1    C      FORTRAN IV COMPILER
ØØ34              2417Ø-6ØØØ2    C
ØØ35              2417Ø-6ØØØ3    C
ØØ36      FTN4    24177-6ØØØ1    B      FORTRAN IV COMPILER (1ØK AREA)
ØØ37              24177-6ØØØ2    B
ØØ38      ALGOL   24129-6ØØØ1    C      ALGOL COMPILER
ØØ39              24129-6ØØØ2    C
ØØ4Ø      XREF    24223-6ØØØ1    B      CROSS REF TABLE GENERATOR
ØØ41      F2E.N   24151-6ØØØ1    D      RELO SUBR (EAU) LIBR
ØØ42      F2F.N   24248-6ØØØ1    B      RELO SUBR (FP ) LIBR
ØØ43      F4D.N   24152-6ØØØ1    C      RELO SUBR (FTN4) LIBR
ØØ44      FFP.N   129Ø7-16ØØ1    A      RELO SUBR (FFP) LIBR
ØØ45      ATDØ1   24381-16ØØ1    1419   ASYNC TERMINAL DRIVER #1
ØØ46      DVR33   24278-6ØØØ1    1419   129Ø8 WCS DRIVER
ØØ47      MASMB   24332-6ØØØ1    1419   129Ø8 WCS MICRO ASSEMBLER
ØØ48      WCSUT   24333-6ØØØ1    A      129Ø8 WCS I/O UTILITIES
ØØ49      MDBUG   24334-6ØØØ1    1419   129Ø8 WCS DEBUG EDITOR
**** LIST END ****
@
```

At this point, use the list printed to select those modules which are to be included in the system to be generated. The PURGE directive is used to flag modules and drivers for deletion. Some guidelines for building a DOS-III system follow.

a. These modules must be included in every system:

DISCM		Disc Monitor
$EXMD		EXEC Modules
DVR01		Paper Tape Reader Driver
DVR00 DVR05 DVR26	Choose One,	System Console Driver
DVR31		Disc Device Driver
JOBPR		Job Processor
EFMP		Include if EFMP or IMAGE is desired
F2E.N F2F.N	Choose One,	EAU Floating-point Arithmetic

b. These driver modules are required if the associated peripheral device is included in the system to be generated:

D2767 D2610 D2607	Choose One,	DVR12 — HP 2767 Line Printer Driver DVR12 — HP 2610/2614 Line Printer Driver DVR12 — HP 2607 Line Printer Driver
DVR23		HP 7970A/B/E Magnetic Tape Driver
DVR02		HP 2895/2753 Paper Tape Punch Driver
D2892		DVR11 — HP 2892 Card Reader Driver
DVR67		HP 12889A Interface Driver
DVR72		HP 12587B Interface Driver
DVR73		HP 12920A/B MUX Driver
DVR33		HP 12908A WCS Driver

c. These modules are normally included during system generation, but may be run from the user area instead:

RLODR	Relocating/Linking Loader
ASMB	Assembler
ALGOL	ALGOL Compiler
FTN4	FORTRAN IV Compiler
F4D.N	FORTRAN IV Library — Required in addition to the library selected under point a above if FORTRAN IV or EFMP is included in the system to be generated.
FFP.N	FFP Library — Required if the FFP option is present (this module must appear in the directory *after* F4D.N).
XREF	Cross Reference Table Generator

d. These modules should be included if WCS is present:

MASMB	HP 12908 WCS Micro-assembler
WCSUT	HP 12908 WCS I/O Utilities

e. These modules must be deleted from the cartridge on Subchannel 0 under specific conditions:

FFP.N	If FFP hardware is not present
F2F.N	If Floating Point hardware is not present
RDBUG	If HP 21MX is present

Any drivers not required by the system to be generated.

f. This module must be deleted from the cartridge on Subchannel 0:

INDEX
MDBUG

In the dialog following, assume that a DOS-III system is to be generated which includes these modules:

DISCM
$EXMD
DVR01
DVR02
DVR03
D2767
DVR23
DVR31
JOBPR
RLODR
ASMB
FTN4
XREF
F2F.N
F4D.N

The dialog continues from the @ symbol at the end of Step 11:

:PURGE,EFMP,DVR00,DVR26,F2E.N,D2892,D2610,D2607
EFMP
DVR00
DVR26
F2E.N
D2892
D2610
D2607
@

```
:PURGE,DVR67,DVR72,DVR73,ALGOL,FFP.N,ATD01,DVR33
DVR67
DVR72
DVR73
ALGOL
FFP.N
ATD01
DVR33
@
:PURGE,.FTN4,MASMB,WCSUT,MDBUG
.FTN4
MASMB
WCSUT
MDBUG
@
:PURGE,INDEX
INDEX
@
:LIST,U,1                    List the user directory on the console
NAME TYPE  SCTRS  DISC ORG   PROG LIMITS   B.P. LIMITS   ENTRY FWAM   PB
SUBCHAN=Ø
DISCM RB   ØØØ26  TØØ7 ØØØ
$EXMD RB   ØØ1Ø8  TØØ7 Ø26
DVRØ1 RB   ØØØØ4  TØØ9 Ø43
DVRØ2 RB   ØØØØ3  TØØ9 Ø47
DVRØ5 RB   ØØØØ3  TØ1Ø ØØ2
D2767 RB   ØØØØ4  TØ1Ø Ø11
DVR23 RB   ØØØØ6  TØ1Ø Ø27
DVR31 RB   ØØØØ5  TØ1Ø Ø37
JOBPR RB   ØØØ81  TØ15 ØØØ
RLODR RB   ØØØ59  TØ16 Ø33
ASMB  RB   ØØØ88  TØ17 Ø44
FTN4  RB   ØØ177  TØ24 Ø28
XREF  RB   ØØØ23  TØ3Ø ØØ6
F2F.N RB   ØØ113  TØ33 ØØ4
F4D.N RB   ØØ148  TØ35 Ø21
@
:EJOB                        Terminate current job
END JOB
@
```

The modules residing on Subchannel 0 are ready to be used for DOS-III system generation. Proceed as follows:

1. Load the DSGEN program from paper tape using the Stand-alone Paper Tape Bootstrap Loader or the paper tape loader ROM.

2. If the DSGEN program loaded is not configured, perform the procedure under "DSGEN Configuration" presented earlier in this section.

3. Use DSGEN to format the disc cartridge on Subchannel 1. When this step is completed, the computer will halt with 102077_8 in the Memory Data register.

4. Use DSGEN to generate a DOS-III system on Subchannel 1. Proceed as directed under "Using DSGEN to Generate DOS-III" in this section.

After DOS-III system generation is completed, modules to be run from the user area of disc memory can be retrieved from the master DSGEN cartridge. For example, if WCS is present in the system, the modules WDBUG and ATD01 may be loaded into the user area as follows:

1. Insert the master DSGEN disc cartridge in the HP 7900 Disc device.

2. Load DOS-III from Subchannel 0 and initiate it using the Stand-alone Paper Tape Bootstrap Loader or the disc loader ROM.

3. System dialog begins:

```
INPUT :DATE,XXXXXXXXXX

@:DATE,,                    DATE directive entered
SUBCHAN=0
LBL=DSGEN
@
:UD,DSGEN,1                 Change user disc to Subchannel 1, label is DSGEN
@
:DD,U,MDBUG,ATD01           Disc-to-disc dump of specified files from user area
@
:UD,*,0                     Destination disc for dump operation
@
:LIST,U,1                   List user directory to verify that modules were copied
```

4. The system will print a list of the user directory on the console.

```
INPUT :DATE,XXXXXXXXXX

@:DA,,
SUBCHAN=1
LBL=DSGEN
@



:JOB
JOB       ,
@
:UD,*,Ø
LBL=SYSTEM
DISC GEN CODE 65ØØ NOT SYS GEN CODE Ø529 ERR POSS
RE-ENTER STATEMENT ON TTY.
@
:UD,SYSTEM,Ø
DISC GEN CODE 65ØØ NOT SYS GEN CODE Ø529 ERR POSS
@
:IN,*
DOS LABEL SYSTEM
OK TO PURGE?
YES
@
:UD,DSGEN,1
@
:UD
SUBCHAN=1
LBL=DSGEN
@
:DD
@
:UD,*,Ø
@
INPUT :DATE,XXXXXXXXXX

@:DATE,,
SUBCHAN=Ø
LBL=DSGEN
@
:JOB
JOB       ,
@
```

```
:LIST,S,1,INDEX
0001      DOS III B  (24307B)  REV 1419
0002      THIS INDEX RELATES THE NAMES OF THE RELOCATABLE MODULES
0003      TO THE PART NUMBERS OF THE EQUIVALENT PAPER TAPES AND
0004      INDICATES THE PURPOSE OF THE MODULES IN THE SYSTEM.
0005      NAME    PART NUMBER    REV    DESCRIPTION
0006      DISCM   24307-16002    1419   DISC MONITOR
0007      $EXMD   24307-16003    1419   EXEC MODULES
0008      DVR00   20985-60001    1419   TTY-LIKE CONSOLE/TERMINAL
0009      DVR01   20987-60001    1419   PAPER TAPE READER
0010      DVR02   20989-60001    1419   PAPER TAPE PUNCH
0011      DVR05   24157-60001    1419   TTY-LIKE CONSOLE
0012      D2892   24272-60001    1419   2892B CARD READER (DVR11)
0013      D2767   24168-60001    1419   2767A LINE PRINTER (DVR12)
0014      D2610   24271-60001    1419   2610A/2614A LINE PRINTER (DVR12)
0015      D2607   24349-60001    1419   2607A LINE PRINTER (DVR12)
0016      DVR23   13024-60001    1419   7970B/E MAG TAPE
0017      DVR26   24333-60001    1419   2762A CONSOLE PRINTER
0018      DVR31   24156-60001    1419   7900/7901/2870 DISC
0019      DVR67   24341-16001    1419   12889A HS SERIAL IF
0020      DVR72   24350-16001    1419   12587B ASYNC DATA SET IF
0021      DVR73   24377-16001    1419   12920A/B MUX
0022      EFMP    24309-60002    1419   EXT FILE MGR EXEC MODULES
0023              24309-60003    1419   EXT FILE MGR UTILITIES
0024      JOBPR   24307-16004    1419   JOB PROCESSOR
0025      RLODR   24308-60001    1419   RELOCATING-LINKING LOADER
0026      ASMB    24158-60001    B      ASSEMBLER
0027              24158-60002    B
0028              24158-60003    B
0029              24158-60004    B
0030              24158-60005    B
0031              24158-60006    B
0032              24158-60007    B
0033      .FTN4   24170-60001    C      FORTRAN IV COMPILER
0034              24170-60002    C
0035              24170-60003    C
0036      FTN4    24177-60001    B      FORTRAN IV COMPILER (10K AREA)
0037              24177-60002    B
0038      ALGOL   24129-60001    C      ALGOL COMPILER
0039              24129-60002    C
0040      XREF    24223-60001    B      CROSS REF TABLE GENERATOR
0041      F2E.N   24151-60001    D      RELO SUBR (EAU) LIBR
0042      F2F.N   24248-60001    B      RELO SUBR (FP ) LIBR
0043      F4D.N   24152-60001    C      RELO SUBR (FTN4) LIBR
0044      FFP.N   12907-16001    A      RELO SUBR (FFP) LIBR
0045      ATD01   24381-16001    1419   ASYNC TERMINAL DRIVER #1
0046      DVR33   24278-60001    1419   12908 WCS DRIVER
0047      MASMB   24332-60001    1419   12908 WCS MICRO ASSEMBLER
0048      WCSUT   24333-60001    A      12908 WCS I/O UTILITIES
0049      MDBUG   24334-60001    1419   12908 WCS DEBUG EDITOR
**** LIST END ****
```

```
:PU,DVRØØ,DVR26,F2F.N,EFMP,D2767,D2610,D2892,DVR67,DVR73,DVR72
DVRØØ
DVR26
F2F.N
EFMP
D2767
D2610
D2892
DVR67
DVR73
DVR72
@
:PU,DVR33,.FTN4,ALGOL,FFP.N,ATDØ1,MASMB,WCSUT,MDBUG
DVR33
.FTN4
ALGOL
FFP.N
ATDØ1
MASMB
WCSUT
MDBUG
@
:PU,INDEX
INDEX
@
:LIST,U,1

NAME TYPE  SCTRS  DISC ORG   PROG LIMITS   B.P. LIMITS   ENTRY FWAM   PB
SUBCHAN=Ø
DISCM RB   ØØØ26  TØØ7 ØØØ
$EXMD RB   ØØ1Ø8  TØØ7 Ø26
DVRØ1 RB   ØØØØ4  TØØ9 Ø43
DVRØ2 RB   ØØØØ3  TØØ9 Ø47
DVRØ5 RB   ØØØØ3  TØ1Ø ØØ2
D26Ø7 RB   ØØØØ6  TØ1Ø Ø21
DVR23 RB   ØØØØ6  TØ1Ø Ø27
DVR31 RB   ØØØØ5  TØ1Ø Ø37
JOBPR RB   ØØØ81  TØ15 ØØØ
RLODR RB   ØØØ59  TØ16 Ø33
ASMB  RB   ØØØ88  TØ17 Ø44
FTN4  RB   ØØ177  TØ24 Ø28
XREF  RB   ØØØ23  TØ3Ø ØØ6
F2E.N RB   ØØ119  TØ3Ø Ø29
F4D.N RB   ØØ148  TØ35 Ø21
@
:EJOB
END JOB
@
```

Preparation of the DSGEN cartridge is completed. Proceed as directed under "Using DSGEN to Generate DOS-III" in this section. Sample generation dialog follows:

```
SYS GEN CODE?
4000

SYS DISC CHNL?
11

DISC TYPE?
7900

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?
17

PRIV INT CARD CHNL?
0

# DMA CHANNELS?
2
```

```
LWA MEM?
37677

ALLOW :SS?
YES

PRGM INPT?
DF
INPUT DISC SUBCHNL?
Ø

LIBR INPT?
PT

PRAM INPT?
TY


*EOT


 NO UNDEF EXTS


ENTER PROG PARAMETERS

$EX18,Ø
/E


# LINKS?

8ØØ


SYSTEM

NAME        PROG BOUNDS    BP BOUNDS

DISCM
  $TIME     Ø5274          ØØ6Ø3
  $SETP     Ø5421          ØØ6Ø3
  $PFAL     Ø5442          ØØ6Ø3

(BOUNDS)    Ø2ØØØ Ø5445    ØØ337 ØØ6Ø3


$EX18

(BOUNDS)    Ø5445 Ø6341    ØØ6Ø3 ØØ631
```

```
$$MGT

(BOUNDS)    06341 07152    00631 00647

DVR05

(BOUNDS)    07152 07421    00647 00651

DVR31

(BOUNDS)    07421 10145    00651 00713

F2E.D

(BOUNDS)    10145 10145    00713 00713

F4D.C

(BOUNDS)    10145 10145    00713 00713


ENTER SUBSYSTEM NAMES

/E

* EQUIPMENT TABLE ENTRY

10,DVR00
ERR 25
10,DVR05
11,DVR31,D
13,DVR01
14,DVR02
15,DVR23,D,0
15,DVR23,D,1
20,DVR12
/E
```

```
* DEVICE REFERENCE TABLE

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


* INTERRUPT TABLE

1Ø,EQT,1
12,EQT,2
13,EQT,3
14,EQT,4
16,EQT,5
2Ø,EQT,7
/E


EXEC SUPERVISOR MODULES

NAME        PROG BOUNDS    BP BOUNDS

$EXØ1
  $ADDR     10520          ØØ714

(BOUNDS)    10433 10535    ØØ713 ØØ714


$EXØ2
  $ADDR     10523          ØØ714

(BOUNDS)    10433 10540    ØØ713 ØØ714
```

```
$EX03

(BOUNDS)    10433 10504    00713 00713


$EX04
  ASCII     11025          00715

(BOUNDS)    10433 11147    00713 00715


$EX05
  $SRCH     10516          00714

(BOUNDS)    10433 11157    00713 00714


$EX06
  $SRCH     10535          00715
  $ADDR     11176          00715

(BOUNDS)    10433 11213    00713 00715


$EX07
  $ADDR     10620          00714

(BOUNDS)    10433 10635    00713 00714


$EX08
  $ADDR     10606          00714

(BOUNDS)    10433 10623    00713 00714


$EX09
  ASCII     11032          00715

(BOUNDS)    10433 11154    00713 00715


$EX10

(BOUNDS)    10433 10771    00713 00713
```

```
$EX11
  $SRCH     10456          00714

(BOUNDS)    10433 11117    00713 00714


$EX12

(BOUNDS)    10433 10717    00713 00713


$EX13
  ASCII     11010          00715

(BOUNDS)    10433 11132    00713 00715


$EX14
  ASCII     11156          00714

(BOUNDS)    10433 11300    00713 00715


$EX15
  ASCII     11002          00714

(BOUNDS)    10433 11124    00713 00715


$EX16

(BOUNDS)    10433 10564    00713 00713


$EX17
  $LBL      11023          00716

(BOUNDS)    10433 11131    00713 00720


$EX19
  $LBL      11026          00720

(BOUNDS)    10433 11134    00713 00720


$EX20

(BOUNDS)    10433 11117    00713 00713
```

```
$EX21
  $SRCH     11472          00714

(BOUNDS)    10433 12133    00713 00742


$EX22

(BOUNDS)    10433 12603    00713 00727


I/O DRIVER MODULES

NAME        PROG BOUNDS    BP BOUNDS

DVR01

(BOUNDS)    12603 13174    00742 00744


DVR02

(BOUNDS)    12603 13030    00742 00744


DVR12

(BOUNDS)    12603 13374    00742 00744


DVR23

(BOUNDS)    12603 13422    00742 00744


LWA LINKS   00744

FWA USER LINKS?
744


LWA PROG    13422

FWA USER?
14000
```

```
USER SYSTEM PROGRAMS

NAME        PROG BOUNDS    BP BOUNDS

JOBPR

(BOUNDS)    14000 26613    00744 01412


LOADR
  .EAU.     25511          01446
  DUMRX     25561          01452

(BOUNDS)    14000 25641    00744 01452


ASMB

(BOUNDS)    14000 21131    00744 01323


ASMBD

(BOUNDS)    21131 21741    01323 01324


ASMB1

(BOUNDS)    21131 22553    01323 01367


ASMB2

(BOUNDS)    21131 22570    01323 01351


ASMB3

(BOUNDS)    21131 22002    01323 01327


ASMB4

(BOUNDS)    21131 22040    01323 01331


ASMB5

(BOUNDS)    21131 22445    01323 01346
```

```
FTN4

(BOUNDS)    14000 27170    00744 01312

F4.0

(BOUNDS)    27170 35041    01312 01374

F4.1

(BOUNDS)    27170 32732    01312 01426

F4.2

(BOUNDS)    27170 34260    01312 01410

XREF
  .OPSY     17241          01033
  DUMRX     17301          01035

(BOUNDS)    14000 17361    00744 01035


*SYSTEM STORED ON DISC
```

SECTION XI
Loading DOS-III

This section describes the loaders used to load a generated DOS-III system into main memory.

Loaders for an HP 21MX computer and an HP 2100A/S computer are essentially the same — the only apparent difference being how the loaders are supplied.

HP 21MX LOADERS

The HP 21MX processor is equipped with a paper tape loader ROM. The contents of this ROM are equivalent to the Basic Binary Loader portion of the BMDL used with HP 2100A/S computers. The ROM contents must be placed in memory before programs can be loaded from paper tape.

The HP 21MX processor can also be equipped with an optional disc loader ROM. This ROM performs the same functions as the BMDL used with HP 2100A/S computers. (If your HP 21MX does not have the disc loader ROM installed, follow steps for loading programs from disc using HP 2100A/S loaders.) The ROM contents must be placed in memory before programs can be loaded from a disc.

To load either of the ROM's into memory, perform the following at the operator panel:

a. Press PRESET.

b. Select the S-register for display in the Display Register.

c. Press CLEAR DISPLAY to clear the contents of the Display Register.

d. Bits 15 and 14 of the Display Register are used to select the loader ROM to be loaded. The paper tape loader ROM is automatically selected by the clear display operation. Set bit 14 to select the disc loader ROM.

e. Change bits 11 through 6 of the Display Register to the octal select code of the disc or paper tape reader.

f. Change bits 5 through 0 of the Display Register to contain the system disc subchannel number.

g. Press STORE to store the contents of the Display Register in the S-register.

h. Press IBL to load the contents of the selected loader ROM into the uppermost 64 locations in the first 32K of directly addressable memory.

   The computer halts with octal 102077 in the T-register (indicating a successful load of the ROM).

i. Place the tape or disc (containing the program to be loaded) into the input device and ready that device.

HP 2100A/S LOADERS

To load a generated DOS-III system from the disc into main memory of an HP 2100A/S computer, execute either the BMDL or the Stand-alone Bootstrap Loader. The former resides in the uppermost 64 words of main memory and is hardware protected. The BMDL exists in two versions depending on the type of disc drive included in the system (HP 7900/7901, HP 2883/2884). Operation of these loaders is essentially the same. They consist of two parts: a Basic Binary Loader which loads absolute binary programs into main memory (from paper tape devices), and a disc loader which loads the configured DOS-III system from the disc into main memory.

The BMDL loads the system from any active subchannel, with one major requirement: whether that particular system is loaded or not, a configured DOS-III system must exist on the disc starting at head 0, drive 0 of the disc device. Head 0, drive 0 corresponds to Subchannel 0 on the HP 2883/2884 disc, or to Subchannel 1 on the HP 7900/7901 disc. The BMDL will read that system or any other configured DOS-III system on the disc as long as a configured system resides on head 0, drive 0.

To load a configured DOS-III system when no system exists on head 0, drive 0, the user must load the Stand-alone Bootstrap Loader into main memory (using the paper tape portion of the BMDL) and execute the Stand-alone Bootstrap Loader. This program loads the configured DOS-III system from the specified disc subchannel without the existence of a configured system on head 0, drive 0 of the disc.

USING THE BMDL TO LOAD ABSOLUTE BINARY PROGRAMS

The BMDL loads absolute binary program tapes into main memory of an HP 2100A/S computer. The Loader resides in the last 64_{10} words of main memory.

Note: When using an HP 21MX computer, the paper tape loader ROM is used to load absolute binary program tapes into memory. Replace steps 2 through 6 (below) with the procedure (described earlier) for loading a paper tape ROM.

Operating Instructions

1. Halt the computer.
2. Place the tape to be loaded into the paper tape input device and ready that device.
3. Set the Loader starting address according to the memory size of the computer:

Memory Size	Starting Address (octal)
16K	037700
24K	057700
32K	077700

4. Clear the switch register.
5. Enable the Loader
6. Press both PRESET buttons.
7. Press RUN.
8. After all or part of the tape is read, the computer halts with $1020xx_8$ displayed.

If xx = 11, a checksum error was detected. Check for torn tape or dust in the reader, check the tape for ragged edges or torn holes, then return to step 2.

If xx = 55, an address error was detected. A program being loaded attempted to enter a location reserved for the main-memory resident Loader, or a location not available in the computer. Check that an absolute binary tape was used, and that it was placed properly in the reader.

If xx = 77, the tape was loaded correctly.

INITIATING DOS-III WITH THE BMDL

When DOS-III has been generated on the disc (by DSGEN), it can be loaded into main memory and initiated by a main-memory resident program called the BMDL. This program resides permanently in the last 64_{10} words of main memory and is hardware protected. Once DOS-III has been loaded and initiated, it is ready to process user tasks.

Note: When using an HP 21MX computer with a disc loader ROM installed, steps 1 through 6 (below) are replaced by the ROM loading procedures described earlier.

Operating Instructions

1. Verify that a configured DOS-III system resides on head 0, drive 0 of the disc. (Head 0, drive 0 corresponds to subchannel 1 for the HP 7900/7901, or to subchannel 0 for the HP 2883/2884 disc.) If a configured system does not reside there, then use the Stand-alone Bootstrap Loader program (see *Initiating DOS-III with the Stand-alone Bootstrap Loader*, in this Section).

2. Set a starting address of 0x7750, where x = 3 for 16K; x = 5 for 24K; x = 7 for 32K.

3. Enable (unprotect) the main-memory resident Loader.

4. Press PRESET button(s) and start the computer executing.

5. The computer halts with 102077_8 displayed in the Display register. Protect the main-memory resident Loader (if necessary).

6. Set the disc subchannel number of the system to be loaded into the switch register (bits 5 through 0).

7. Start computer execution. The system is loaded into main memory and prints the following message:

   INPUT :DATE, XXXXXXXXXX (No Time-base Generator)
   or
   INPUT :DATE, XXXXXXXXXX,H,M (Time-base Generator)

8. All other directives are ignored until a valid DATE directive is entered. Immediately following the DATE directive, the only valid directives are :TRACKS, :BATCH, :TYPE, and :JOB. All other directives are ignored until a JOB directive is entered.

CONFIGURING THE DOS-III STAND-ALONE BOOTSTRAP LOADER

Once DOS-III has been generated onto a disc, it may be initiated into operating status using the DOS-III Stand-alone Bootstrap. The Bootstrap, however, must be configured before being used.

Operating Instructions

1. Turn on all equipment.

2. Load (using the BMDL or the paper tape loader ROM) and configure the SIO Punch or Teleprinter Driver.

3. Load the Bootstrap using the BMDL or the paper tape loader ROM.

4. Set up the Bootstrap configuration starting address at location 2_8.

5. Set switch register bits 5 through 0 equal to the octal channel number (select code) of the disc controller (low number, high priority channel).

6. Set switch register bit 15 on to punch a configured Bootstrap tape; off to configure the Bootstrap in main memory only.

7. Start the computer executing.

8. If bit 15 of the switch register is set, the Bootstrap punches out a configured copy of itself and halts. For another copy, simply start the computer executing again.

INITIATING DOS-III WITH THE STAND-ALONE BOOTSTRAP LOADER

When DOS-III has been generated onto the disc, it can be loaded into main memory and initiated by using a small stand-alone program called the Stand-alone Bootstrap Loader. Once DOS-III has been loaded and initiated, it is ready to process user tasks.

Note: The Stand-alone Bootstrap Loader need be used only if a configured DOS-III system does not reside on head 0, drive 0 of the disc. If a system resides on the disc in the above mentioned area, the BMDL can be used.

Operating Instructions

1. Turn on all equipment.

2. Configure a Stand-alone Bootstrap Loader (as previously described).

3. Load the configured Bootstrap into main memory using the BMDL or the paper tape loader ROM.

4. Set up the starting address of the Bootstrap at location 100_8.

5. Set switch register bits 5 through 0 equal to the octal subchannel of the system disc. (If this subchannel differs from that established at system generation time, the new subchannel overrides the old.)

6. Set switch register bit 14 equal to one if the disc type is 2883 with two subchannels per drive; to zero if the disc type is 7900, 7901, or 2883 with four subchannels per drive.

7. Start the computer executing.

8. When DOS-III has been loaded into main memory, it prints the following message:

   INPUT :DATE,XXXXXXXXXX (No Time-base Generator)
   or
   INPUT :DATE, XXXXXXXXXX,H,M (Time-base Generator present)

9. All other directives are ignored until a valid DATE directive is entered. Immediately following the DATE directive, the only valid directives are :TRACK, :BATCH, :TYPE, and :JOB. All others are ignored until one of these directives is entered.

BMDL

The BMDL resides in the last 64_{10} words of main memory (hardware protected by a button/switch on the computer front panel) and is responsible for loading main-memory resident modules from configured DOS-III systems residing on the disc into main memory. The BMDL also loads absolute binary programs into main memory through the paper tape input device. A separate version of the BMDL exists for each of two classes of disc, depending upon which disc type is used with the system (HP 7900/7901, or HP 2883/2884). Only one version can exist in main memory at any one time. The following two tables show the last 64_{10} word addresses and their octal contents for each version of the BMDL.

Note: When using the HP 7900/7901 BMDL with a newly-inserted 7900 or 7901 disc cartridge, it is necessary to execute the bootstrap twice. After executing the bootstrap the first time, the system loops; it must be halted and the bootstrap executed a second time. This procedure does not apply to the Stand-alone Bootstrap.

Table 11-1. HP 7900/7901 BMDL

Address	Contents	Address	Contents	
x7700	002701	x7740	1023kk	Paper tape loader starting address = $x7700_8$; Moving-head disc loader starting address = $x7750_8$ (PRESET must be pressed).
x7701	063722	x7741	027740	
x7702	002307	x7742	1064kk	
x7703	102077	x7743	002041	
x7704	017735	x7744	127735	
x7705	007307	x7745	005767	
x7706	027702	x7746	027737	
x7707	077733	x7747	030000*	x = 3 for 16k, 4 for 20k, 5 for 24k, 6 for 28k, 7 for 32k
x7710	017735	x7750	002400	
x7711	017735	x7751	1026cc	
x7712	074000	x7752	1037cc	
x7713	077734	x7753	067747	
x7714	067734	x7754	1066dd	kk = tape input device select code
x7715	047777	x7755	1037dd	
x7716	002040	x7756	1066cc	
x7717	102055	x7757	063776	dd = low priority (higher numbered) disc select code
x7720	017735	x7760	102606	
x7721	040001	x7761	067732	
x7722	177734	x7762	106602	
x7723	037734	x7763	1037cc	cc = high priority (lower numbered) disc select code
x7724	000040	x7764	102702	
x7725	037733	x7765	106602	
x7726	027714	x7766	013741	
x7727	017735	x7767	1026dd	n = 4 for 16k, 3 for 20k, 2 for 24k, 1 for 28k, 0 for 32k
x7730	054000	x7770	1037cc	
x7731	027701	x7771	103706	
x7732	102011	x7772	1037dd	
x7733	000000	x7773	1023dd	*The HP 7900/7901 BMDL can be altered to boot a DOS-III system from subchannel 0 instead of subchannel 1 by changing the contents of address x7747 from 30000_8 to 31000_8.
x7734	000000	x7774	027773	
x7735	000000	x7775	127717	
x7736	006600	x7776	1200cc	
x7737	1037kk	x7777	1n0100	

Table 11-2. HP 2883/2884 BMDL

Address	Contents	Address	Contents	
x7700	002701	x7740	1023kk	Paper tape loader starting address = $x7700_8$; Moving-head disc loader starting address = $x7750_8$ (PRESET must be pressed).
x7701	063722	x7741	027740	
x7702	002307	x7742	1064kk	
x7703	102077	x7743	002041	
x7704	017735	x7744	127735	
x7705	007307	x7745	005767	
x7706	027702	x7746	027737	
x7707	077733	x7747	177600	x = 3 for 16k, 4 for 20k, 5 for 24k, 6 for 28k, 7 for 32k
x7710	017735	x7750	063775	
x7711	017735	x7751	1026dd	
x7712	074000	x7752	1037dd	
x7713	077734	x7753	1023dd	
x7714	067734	x7754	027753	kk = tape input device select code
x7715	047777	x7755	067776	
x7716	002040	x7756	106606	
x7717	102055	x7757	067732	dd = low priority (higher numbered) disc select code
x7720	017735	x7760	106602	
x7721	040001	x7761	102702	
x7722	177734	x7762	067747	
x7723	037734	x7763	106602	cc = high priority (lower numbered) disc select code
x7724	000040	x7764	001000	
x7725	037733	x7765	1067dd	
x7726	027714	x7766	1026dd	
x7727	017735	x7767	1037cc	n = 4 for 16k, 3 for 20k, 2 for 24k, 1 for 28k, 0 for 32k
x7730	054000	x7770	103706	
x7731	027701	x7771	1037dd	
x7732	102011	x7772	1023dd	
x7733	000000	x7773	027772	
x7734	000000	x7774	127717	
x7735	000000	x7775	020000	
x7736	006600	x7776	1200cc	
x7737	1037kk	x7777	1n0100	

PART 4
DOS-III Systems Programming

SECTION XII
User-written EXEC Modules

DOS-III is capable of accepting user-written EXEC modules. Up to two EXEC modules may be written; these must be loaded with all the DOS-III EXEC modules during DOS-III Generation. (See Section X, "Generating DOS-III" for details.)

This section presents the user-written EXEC call directives and calling sequences, along with a brief description of internal design and a sample EXEC module.

For example, DOS-III halts on power failure. The user may write a power fail recovery routine. Because of system requirements, the routine must be called $PFAL.

USER EXEC MODULES: DIRECTIVES

Purpose

To execute user EXEC modules.

Format

:EA[,p1,. . . ,p5]	*(Calls EXEC module $EX36)*
:EB[,p1,. . . ,p5]	*(Calls EXEC module $EX37)*

where all parameters are non-negative decimal integers.

Comments

Number and meaning of the parameters varies depending upon user definition of the EXEC module.

USER EXEC MODULES: EXEC CALLS

Purpose

To execute either user-created EXEC module $EX36 or $EX37. The number of parameters in the EXEC call are defined by the user. The general format of the call is

Assembly Language

```
                EXT   EXEC
                 .
                 .
                 .
                JSB   EXEC        (Transfer control to DOS-III)
                DEF   *+2 (to 7)  (Determine number of parameters—from 1 to 5)
                DEF   RCODE       (Define request code)
                DEF   PRAM1       (Define the first optional parameter)
                 .
                 .
                 .
                DEF   PRAM5       (Define the fifth optional parameter)
                 .
                 .
                 .
RCODE           DEF   27 (or 28)  (RCODE for $EX36 = 27; RCODE for $EX37 = 28)
PRAM1           ---               (Up to five words of parameter information)
                 .
                 .
                 .
PRAM5           ---
```

FORTRAN

```
IRCDE = 27 (or 28)
CALL EXEC (IRCDE[,P1, . . . . . .P5] )
```

USER EXEC MODULES: INTERNAL DESIGN

EXEC modules are typically type-1 Assembly-language routines which are incorporated at generation time as part of the operating system. As "system" modules, they execute with the interrupt system and memory protect off. They may directly access entry points and subroutines within the system, but must not issue any EXEC calls (EXEC processing is not re-entrant). Also, user-written EXEC modules should be defined as disc-resident supervisory modules; the NAM pseudo-instruction for these modules should indicate that the routine is a type-1 program.

Special programming considerations are required upon initiation and completion.

Initiation

Upon entry, information used in processing the EXEC function can be found in the following base page locations.

Location	Name	Definition
224_8	RQCNT	# of parameters in the calling sequence
225_8	RQRTN	return address upon completion
226_8	RQP1	address of request code
$227\text{-}233_8$	RQP2-RQP6	address(es) of specified parameters

Completion

Prior to returning to the system, the EXEC module must

1. release itself from the EXEC module overlay area if it is disc-resident. This code handles EXEC module release:

```
                LDA   EXMOD    (Get current module in overlay area)
                CPA   NUMB     (Is it this one?)
                CMA,INA        (Yes—set value positive)
                STA   EXMOD    (No—leave value alone)
                  .
                  .
                  .
EXMOD           EQU   245B
NUMB            DEC   -36 (or -37)
```

2. place the desired transfer address in XIRT (location 137_8) and jump to the label $IRT (defined as an EXTernal), for example,

```
            EXT   $IRT
             .
             .
             .
            LDA   RQRTN    (Set the return address)
            STA   XIRT
            JMP   $IRT     (Transfer to system)
RQRTN       EQU   225B
XIRT        EQU   137B
```

SAMPLE EXEC MODULE

```
 PAGE 0001

0001                ASMB,L,C,X,N,R,B   DISC WORK LIMITS MODULE (SEX02)
**  NO ERRORS*
```

```
0001                    ASMB,L,C,X,N,R,B     DISC WORK LIMITS MODULE ($EX02)
0002  00000                     NAM $EX02,1
0003                            ENT $EX02
0004                            EXT $RQER,$ADDR
0005                            EXT $IRT
0006*     $EX02 ROUTINE PROVIDES THE USER WITH DISC WORK AREA TRACK
0007*   ADDRESS LIMITS AND THE # OF SECTORS PER DISC TRACK.
0008*
0009*     CALLING SEQUENCE:
0010*
0011*         JSB EXEC
0012*         DEF *+5(OR 6)
0013*         DEF RCODE       RCODE = 17
0014*         DEF FTRAK       FTRAK = ADDR OF WORD TO STORE 1ST WORK TRK
0015*         DEF LTRAK       LTRAK = ADDR OF WORD TO STORE LAST WORK TRK
0016*         DEF SIZE        SIZE = # SECTOR/TRACK WORD ADDR.
0017*         DEF DISC(OPTIONAL)  DISC = 0 FOR SYSTEM DISC, NON-0 FOR US
0018*                             DEFAULT IS SYSTEM.
0019*
0020  00000 060224  $EX02 LDA RQCNT     CHECK PARAMETER COUNT
0021  00001 050057        CPA ..+4      4 PARAMETERS?
0022  00002 026010R       JMP CHK       YES. OK.
0023  00003 050060        CPA ..+5     5 PARAMETERS?
0024  00004 002001        RSS          YES. OK
0025  00005 026065R       JMP RQER       TOO FEW OR TOO MANY PARAMETERS.
0026  00006 060232        LDA RQP5      CHECK ADDR OF 5TH PARAM
0027  00007 016002X       JSB $ADDR
0028  00010 060227  CHK   LDA RQP2
0029  00011 016002X       JSB $ADDR          PARAMETERS
0030  00012 060230        LDA RQP3
0031  00013 016002X       JSB $ADDR
0032  00014 060231        LDA RQP4
0033  00015 016002X       JSB $ADDR
0034  00016 064224        LDB RQCNT
0035  00017 054057        CPB ..+4      DEFAULT AS SYSTEM DISC?
0036  00020 026024R       JMP SYS       YES.
0037  00021 160232        LDA RQP5,I     NO. CHECK 5TH PARAM
0038  00022 002002        SZA          0 MEANS SYSTEM DISC
0039  00023 026057R       JMP USER
0040  00024 060160  SYS   LDA SYNTS     GET START OF WORK AREA TRACK
0041  00025 040074        ADA .377
0042  00026 001727        ALF,ALF
0043  00027 010074        AND .377
0044  00030 170227        STA RQP2,I
0045  00031 060102        LDA JBINC
0046  00032 002003        SZA,RSS
0047  00033 026043R       JMP EX010
0048  00034 010074        AND .377
0049  00035 070001        STA B
0050  00036 020102        XOR JBINC
0051  00037 001727        ALF,ALF
0052  00040 010074        AND .377
```

```
0053   00041 040052          ADA N1
0054   00042 026045R         JMP EX020
0055   00043 060154  EX010 LDA DISCO      STORE END OF WORK AREA TRACK #
0056   00044 010074          AND .377
0057   00045 170230  EX020 STA RQP3,I
0058   00046 060116          LDA SECTR     STORE # OF SECTORS PER TRACK
0059   00047 170231          STA RQP4,I
0060   00050 060245          LDA EXMOD
0061   00051 050051          CPA ..-2
0062   00052 003004          CMA,INA
0063   00053 070245          STA EXMOD
0064   00054 060225          LDA RQRTN     SET UP TRANSFER ADDR FOR $IRT
0065   00055 070137          STA XIRT
0066   00056 026003X         JMP $IRT
0067   00057 060157  USER  LDA UDNTS     GET USER DISC NEXT TR/SECTR
0068   00060 040074          ADA .377
0069   00061 001727          ALF,ALF
0070   00062 010074          AND .377
0071   00063 170227          STA RQP2,I
0072   00064 026043R         JMP EX010
0073   00065 002400  RQER  CLA             FREE MODULE AREA
0074   00066 070245          STA EXMOD
0075   00067 026001X         JMP $RQER
0076*
0077   00000           A     EQU 0
0078   00001           B     EQU 1
0079   00053           ..    EQU 53B
0080   00052           N1    EQU ..-1
0081   00074           .377  EQU ..+17
0082   00100           .     EQU 100B
0083   00102           JBINC EQU .+2
0084   00116           SECTR EQU .+14
0085   00126           RONBF EQU .+26B
0086   00137           XIRT  EQU RONBF+9
0087   00154           DISCO EQU .+44
0088   00157           UDNTS EQU .+47
0089   00160           SYNTS EQU .+48
0090   00224           RQCNT EQU .+84
0091   00225           RQRTN EQU .+85
0092   00227           RQP2  EQU .+87
0093   00230           RQP3  EQU .+88
0094   00231           RQP4  EQU .+89
0095   00232           RQP5  EQU .+90
0096   00245           EXMOD EQU .+101
0097                        END
** NO ERRORS*
```

$ADDR	00004	00027	00029	00031	00033		
$EX02	00020	00003					
$IRT	00005	00066					
$RQER	00004	00075					
.	00082	00083	00084	00085	00087	00088	00089
	00090	00091	00092	00093	00094	00095	00096
..	00079	00021	00023	00035	00061	00080	00081
.377	00081	00041	00043	00048	00052	00056	00068
	00070						
.A	00077						
B	00078	00049					
CHK	00028	00022					
DISCO	00087	00055					
EX010	00055	00047	00072				
EX020	00057	00054					
EXMOD	00096	00060	00063	00074			
JBINC	00083	00045	00050				
N1	00080	00053					
RONBF	00085	00086					
RQCNT	00090	00020	00034				
RQER	00073	00025					
RQP2	00092	00028	00044	00071			
RQP3	00093	00030	00057				
RQP4	00094	00032	00059				
RQP5	00095	00026	00037				
RQRTN	00091	00064					
SECTR	00084	00058					
SYNTS	00089	00040					
SYS	00040	00036					

```
  $EX02        CROSS-REFERENCE SYMBOL TABLE        PAGE 0002

UDNTS   00088    00067
USER    00067    00039
XIRT    00086    00065
```

SECTION XIII
Planning I/O Drivers

STANDARD I/O DRIVERS

Note: 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-III, is responsible for all data transfer between an I/O device and the computer. During its execution, the driver may refer to the base page communication area for information from the system: the device equipment table (EQT) entry, which contains the parameters of the transfer, and the current DMA value (CHAN), which 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 the base page contains the number of the DMA channel assigned to the device, if needed. This section is entered by a jump subroutine (JSB) to the entry point I.*nn*. On entry, the A register contains the select code (channel number) of the device (bits 0 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-III 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 0 through 5 in place of the logical unit. (See Figure A-4 and Section III, I/O READ/WRITE EXEC Call (RCODE = 1 or 2), for details of the parameters.)

Once initiated, the drive can use words 5, 6, and 11 through 14 of the EQT entry in any way, but words 1, 2, 3, 7, 8, 9, 10, 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 is responsible for these functions (as flow-charted in Figure 13-1):

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.

   Note: All drivers must accept a clear request. (Request code = 3, function code = 0.)

2. Configures all I/O instructions in the driver to include the select code of the device (or DMA channel). (Does not apply to DVR05 and 7900/7901 DVR31.)

3. Initializes DMA, if appropriate.

   Note: The initiation section must save the DMA channel number (found in CHAN) in the EQT entry, since it is not set on entry to the continuation section.

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 = 0, then the operation was initiated.

   If A ≠ 0, 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)

   6 = driver cannot handle a control request; the system is instructed to wait

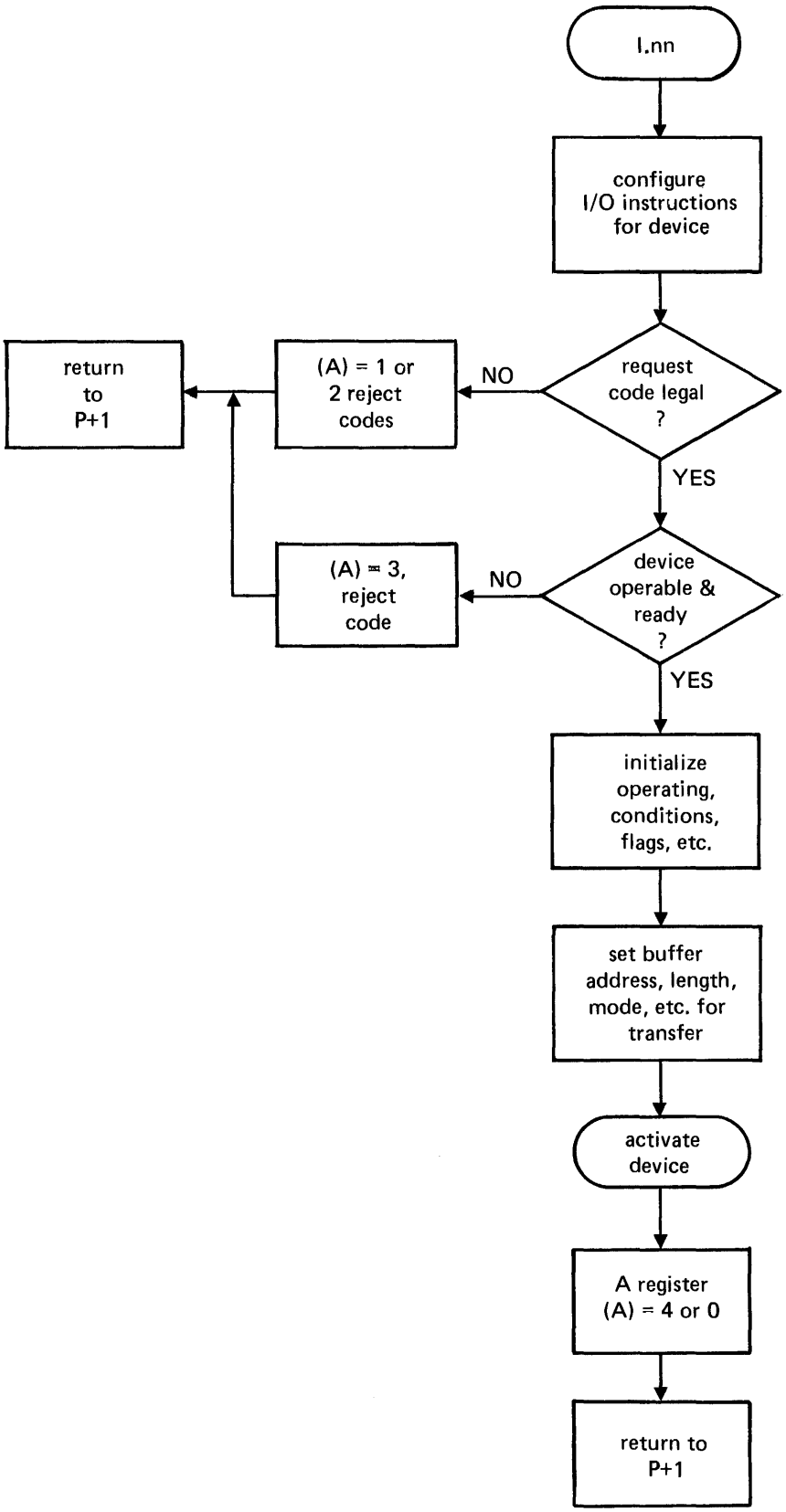


Figure 13-1. I/O Driver Initiation Section

Completion Section

DOS-III 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 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).

FUNCTIONS OF THE COMPLETION SECTION: The completion section of the driver is responsible for the functions below (as flow-charted in Figure 13-2):

1. The driver configures all I/O instructions in the completion section to reference the interrupting device.

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 0 through 7 of EQT word 4.

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

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

      0 = successful completion

      1 = device malfunction or not ready

      2 = end-of-tape (information)

      3 = transmission parity error

6. Clear the device and DMA control on end-of-operation, or set the device and DMA for the next transfer or retry. Return 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.

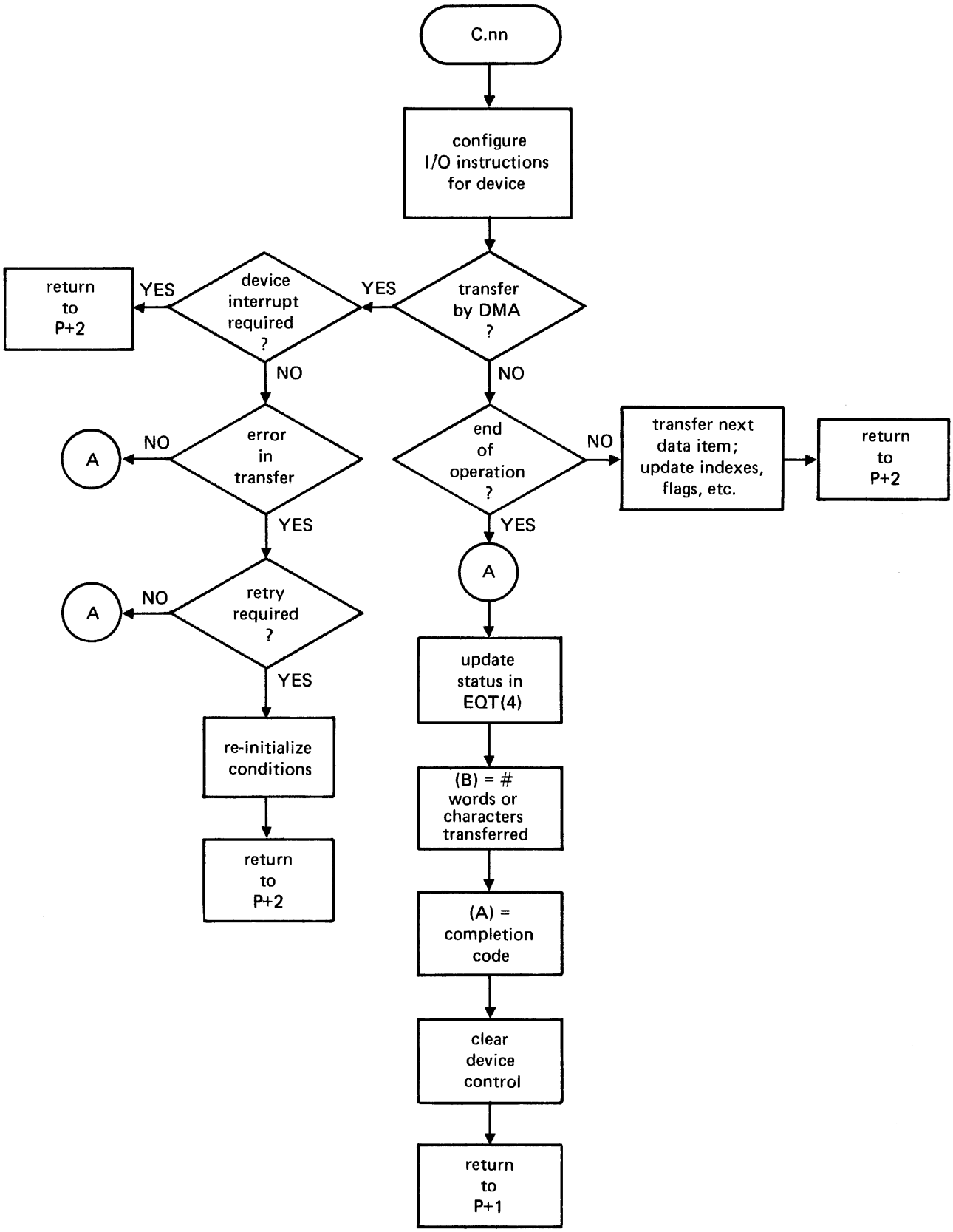


Figure 13-2. I/O Driver Completion Section

SAMPLE I/O DRIVER

The following pages provide an assembly listing and cross-reference symbol table for a sample I/O driver.

```
 PAGE 0001

0001                    ASMB,R,B,L,C
**  NO ERRORS*
```

```
0001                    ASMB,R,B,L,C
0003  00000                   NAM DVR02,4
0004*
0005**********   VERSION 8/24/72
0006*
0007*
0008                         ENT I.02,C.02
0009*
0010****** PROGRAM  DESCRIPTION *******
0011*
0012*    DRIVER 02 OPERATES UNDER THE CONTROL OF THE
0013* I/O CONTROL MODULE OF THE D.O.S. EXECUTIVE
0014* THIS DRIVER IS RESPONSIBLE FOR CONTROLLING
0015* OUTPUT DATA TRANSMISSION WITH A 2753A TAPE PUNCH.
0016* <02> IS THE EQUIPMENT TYPE CODE ASSIGNED TO THIS
0017* TYPE OF DEVICE. I.02 IS THE ENTRY POINT FOR THE
0018* *INITIATION* SECTION AND C.02 FOR THE *COMPLETION*
0019* SECTION.
0020*
0021*  -  THE INITIATION SECTION IS CALLED FROM I/O
0022*     CONTROL TO INITIALIZE A DEVICE AND INITIATE
0023*     AN OUTPUT OR CONTROL OPERATION.
0024*
0025*      CALLING SEQUENCE:
0026*
0027*           - ADDRESSES OF DEVICE EQT ENTRY
0028*             SET IN "EQT1-EQT17" -
0029*
0030*           (A) = I/O ADDRESS OF DEVICE
0031*
0032*        (P)   JSB  I.02
0033*        (P+1) - RETURN -
0034*
0035*              (A) = 0, OPERATION INITIATED, OR
0036*              (A) = REJECT CODE:
0037*
0038*                   1, ILLEGAL READ REQUEST
0039*                   2, ILLEGAL CONTROL FUNCTION
0040*
0041*
0042*  -  THE COMPLETION SECTION IS CALLED BY CENTRAL
0043*     INTERRUPT CONTROL TO CONTINUE OR COMPLETE
0044*     AN OPERATION.
0045*
0046*      CALLING SEQUENCE:
0047*
0048*           - ADDRESSES OF DEVICE EQT ENTRY
0049*             SET IN "EQT1-EQT17" -
0050*
0051*           (A) = I/O ADDRESS OF DEVICE
0052*
0053*        (P)   JSB  C.02
0054*        (P+1) -- COMPLETION RETURN   --
0055*        (P+2) -- CONTINUATION RETURN --
0056*
0057*         - COMPLETION RETURN:
```

```
0058*              (A) = 0, SUCCESSFUL COMPLETION WITH
0059*                 (B) = # WORDS OR CHARACTERS
0060*                       TRANSFERRED.
0061*              (A) = 2  IF *TAPE-SUPPLY-LOW* CONDITION
0062*                       DETECTED AFTER RECORD IS
0063*                       FINISHED.
0064*                 (B), SAME AS FOR (A) = 0
0065*
0066*           - CONTINUATION RETURN: REGISTERS
0067*               MEANINGLESS
0068*
0069*
0070*   - RECORD FORMATS:
0071*
0072*      ASCII:  A STRING OF CHARACTERS, THE NUMBER
0073*      -----   DESIGNATED BY THE BUFFER LENGTH IN
0074*              THE REQUEST, TERMINATED BY A RETURN
0075*              AND LINE-FEED (SUPPLIED BY THE DRIVER).
0076*
0077*        SPECIAL CHARACTER PROCESSING:
0078*
0079*          LEFT-ARROW: IF A LEFT-ARROW IS THE LAST
0080*                      CHARACTER IN THE USER BUFFER,
0081*                      THE RETURN/LINE-FEED AND LEFT
0082*                      ARROW CODES ARE NOT OUTPUT.
0083*
0084*           A ZERO BUFFER LENGTH CAUSES ONLY A RETURN/
0085*               LINE-FEED TO BE OUTPUT.
0086*
0087*      BINARY: A STRING OF CHARACTERS SPECIFIED
0088*      ------ BY THE "BUFFER LENGTH" IN THE REQUEST.
0089*
0090*
0091*   -  CONTROL FUNCTIONS ACCEPTED:
0092*
0093*      10 - TEN INCHES OF ZEROS (FEED-FRAMES) ARE
0094*           OUTPUT FOR LEADER/TRAILER.
0095*
0096*          11 - LINE SPACING: THE PARAMETER WORD OF THE
0097*           CONTROL REQUEST DETERMINES THE NUMBER
0098*           OF LINE-FEEDS TO BE OUTPUT.
0099*
```

```
0101*
0102********* INITIATION  SECTION **********
0103*
0104*
0105  00000 000000  I.02  NOP
0106*
0107  00001 016201R        JSB SETIO     SET I/O INSTRUCTIONS FOR UNIT.
0108*
0109  00002 160213        LDA EQT9,I    GET CONTROL WORD OF REQUEST,
0110  00003 010056        AND .3         ISOLATE.
0111*
0112  00004 050054        CPA .1        ERROR IF REQUEST IS
0113  00005 126000R       JMP I.02,I     FOR INPUT, REJECT CALL.
0114  00006 050055        CPA .2        PROCESS FOR
0115  00007 026043R       JMP D04         WRITE REQUEST.
0116*
0117* CONTROL FUNCTION REQUEST
0118*
0119  00010 160213        LDA EQT9,I    GET CONTROL WORD
0120  00011 001727        ALF,ALF        FROM REQUEST, POSITION AND
0121  00012 001222        RAL,RAL        ISOLATE FUNCTION FIELD.
0122  00013 010073        AND MASK1
0123  00014 002003        SZA,RSS        IS IT A CLEAR?
0124  00015 026024R       JMP CLEAR      YES.
0125  00016 050063        CPA .10B      FIELD = <10> TO GENERATE
0126  00017 026026R       JMP D01        LEADER (10 INCHES OF BLANK TAPE)
0127  00020 050064        CPA .11B      FIELD = <11> FOR LINE
0128  00021 026032R       JMP D02        SPACING.
0129*
0130* REQUEST ERROR - CAUSE REJECT RETURN TO I/O CONTROL
0131*
0132  00022 060055        LDA .2
0133  00023 126000R       JMP I.02,I
0134  00024 106700  CLEAR CLC 0          TURN DEVICE OFF
0135  00025 026066R       JMP I.A.6
0136*
0137* LEADER/TRAILER GENERATOR
0138*
0139  00026 062224R D01   LDA N100      SET INDEX COUNTER FOR FEED FRAMES
0140  00027 170216        STA EQT12,I    = -100.
0141  00030 002400        CLA           (A) = 0 FOR
0142  00031 026041R       JMP D03         FEED FRAME.
0143*
0144* LINE SPACING
0145*
0146  00032 160214  D02   LDA EQT10,I   GET LINE COUNTER WORD.
0147  00033 002021        SSA,RSS       INSURE VALUE
0148  00034 003004        CMA,INA        IS NEGATIVE.
0149  00035 002003        SZA,RSS       PROTECT AGAINST
0150  00036 003400        CCA            A ZERO VALUE.
0151  00037 170216        STA EQT12,I
0152  00040 060065        LDA LINF      (A) = LINE FEED CODE.
0153*
0154  00041 170217  D03   STA EQT13,I   SET ACTION CODE.
0155  00042 026056R       JMP D05
0156*
```

```
0157* WRITE REQUEST PROCESSING
0158*
0159  00043 160214  D04   LDA EQT10,I  CONVERT BUFFER ADDRESS TO EVEN
0160  00044 001200        RAL           CHARACTER ADDRESS AND SET
0161  00045 170216        STA EQT12,I   AS CURRENT BUFFER ADDRESS.
0162  00046 160215        LDA EQT11,I  GET BUFFER LENGTH.
0163  00047 002020        SSA          IF CHARACTER SPECIFIED,
0164  00050 026053R       JMP **3       USE VALUE.
0165  00051 001000        ALS          CONVERT WORDS TO NEGATIVE
0166  00052 003004        CMA,INA       CHARACTERS.
0167  00053 170217        STA EQT13,I  SET CURRENT BUFFER LENGTH.
0168*
0169  00054 002400        CLA
0170  00055 170220        STA EQT14,I   FOR BINARY WRITE.
0171*
0172* CALL *COMPLETION* SECTION TO WRITE FIRST CHAR.
0173*
0174  00056 062223R D05   LDA IEXTA     ADJUST RETURN
0175  00057 072070R       STA C.02      TO *INITIATOR* SECTION.
0176  00060 026075R       JMP D10
0177*
0178  00061 026064R       JMP I.A.4       BINARY READ WITH 0 BUFFER LEN.
0179  00062 002400  IEXIT CLA          RETURN TO I/O CONTROL WITH
0180  00063 126000R       JMP I.02,I    OPERATION INITIATED.
0181*
0182  00064 006400  I.A.4 CLB
0183  00065 174220        STB EQT14,I   CLEAR TLOG.
0184  00066 060057  I.A.6 LDA .4        SET A=4 FOR IMMED.COMPL RETURN
0185  00067 126000R       JMP I.02,I    RETURN
```

```
0187*
0188********* COMPLETION  SECTION **********
0189*
0190*
0191  00070 000000  C.02  NOP
0192*
0193  00071 016201R       JSB SETIO    SET I/O INSTRUCTIONS FOR UNIT.
0194  00072 160207       LDA EQT5,I    GET "CLEAR" FLAG.
0195  00073 002020       SSA           CLEAR?
0196  00074 026155R       JMP IO3      YES.TERMINATE.
0197*
0198  00075 160213  D10   LDA EQT9,I   GET CONTROL WORD
0199  00076 070001       STA B         SAVE FOR CODE TEST.
0200  00077 001727       ALF,ALF      ROTATE MODE BIT
0201  00100 001200       RAL           TO BIT 15
0202  00101 072222R       STA TEMP1    AND SAVE.
0203*
0204  00102 002400       CLA           IF CURRENT BUFFER ADDRESS OR
0205  00103 150216       CPA EQT12,I   FUNCTION INDEX = 0, THEN
0206  00104 026155R       JMP IO3      OPERATION COMPLETED.
0207*
0208  00105 004010       SLB
0209  00106 026133R       JMP D11      = CONTROL FUNCTION =
0210*
0211  00107 150217       CPA EQT13,I   IF CURRENT CHARACTER INDEX =
0212  00110 026137R       JMP D12      0, THEN OUTPUT END OF RECORD.
0213*
0214  00111 164216       LDB EQT12,I   GET CURRENT CHAR. BUFFER ADDRESS.
0215  00112 134216       ISZ EQT12,I   ADD 1 FOR NEXT CHARACTER.
0216  00113 004065       CLE,ERB       CONVERT TO WORD ADDRESS.
0217  00114 160001       LDA B,I      GET WORD AND
0218  00115 002041       SEZ,RSS       POSITION PROPER
0219  00116 001727       ALF,ALF       CHARACTER IN A(07-00).
0220  00117 010074       AND MASK3    REMOVE UPPER POSITION DATA.
0221*
0222  00120 066222R       LDB TEMP1    PUT MODE IN B(15).
0223  00121 134217       ISZ EQT13,I   INDEX CHARACTER COUNTER.
0224  00122 026127R       JMP IO1      = NOT LAST CHARACTER.
0225*
0226  00123 006020       SSB           IF BINARY MODE,
0227  00124 026127R       JMP IO1      WRITE LAST CHARACTER.
0228*
0229  00125 052220R       CPA ARROW    IF CHAR = < + >, THEN OMIT IT
0230  00126 026155R       JMP IO3      AND R/LF ON ASCII RECORD.
0231*
0232* OUTPUT CHARACTER TO PUNCH UNIT.
0233*
0234  00127 102600  IO1   OTA 0         OUTPUT CHARACTER TO INTERFACE
0235  00130 102700  IO2   STC 0         TURN DEVICE ON.
0236  00131 036070R       ISZ C.02     ADJUST RETURN TO (P+2).
0237  00132 126070R       JMP C.02,I    =EXIT=.
0238*
0239* CONTROL FUNCTION OUTPUT
0240*
0241  00133 160217  D11   LDA EQT13,I   (A) = LINE-FEED OR FEED FRAME.
0242  00134 134216       ISZ EQT12,I   INDEX OUTPUT COUNT FOR LEADER/
```

```
0243  00135 000000          NOP             TRAILER OR LINE SPACING.
0244  00136 026127R         JMP I01         GO TO OUTPUT CHARACTER.
0245*
0246* END OF RECORD PROCESSING
0247*
0248  00137 062222R D12     LDA TEMP1       CHECK MODE OF TRANSFER.
0249  00140 002020          SSA
0250  00141 026155R         JMP I03          = BINARY =
0251*
0252  00142 164220          LDB EQT14,I     *ASCII* RECORD
0253  00143 062217R         LDA RETN         OUTPUT FIRST A
0254  00144 056217R         CPB RETN          RETURN AND THEN A
0255  00145 060065          LDA LINF          LINE-FEED.
0256  00146 170220          STA EQT14,I     SET EQT11 FOR LINE-FEED CHECK.
0257  00147 056217R         CPB RETN        IF LINE-FEED IS BEING OUTPUT,
0258  00150 026152R         JMP D14          GO TO SET COMPLETION FLAG.
0259  00151 026127R         JMP I01           = OUTPUT  RETURN =
0260*
0261*
0262  00152 006400  D14     CLB             SET BUFFER ADDRESS = 0
0263  00153 174216          STB EQT12,I      TO INDICATE LAST CHARACTER.
0264  00154 026127R         JMP I01
```

```
0266*
0267* STATUS AND TRANSMISSION COMPLETION SECTION
0268*
0269  00155 102500  I03   LIA 0         GET DEVICE STATUS.
0270  00156 070001        STA B
0271  00157 160206        LDA EQT4,I    REMOVE PREVIOUS
0272  00160 010075        AND MASK2      STATUS.
0273  00161 030001        IOR B         SET NEW
0274  00162 170206        STA EQT4,I     STATUS WORD.
0275  00163 002400        CLA           IF LOW TAPE
0276  00164 006002        SZB            SUPPLY, SET
0277  00165 060055        LDA .2         A = 2 FOR  *EOT*.
0278  00166 164207        LDB EQT5,I     GET "CLEAR" FLAG.
0279  00167 006020        SSB            CLEAR?
0280  00170 026176R       JMP I05        YES.
0281*
0282  00171 164215        LDB EQT11,I   SET  (B) = TRANSMISSION
0283  00172 006020        SSB            LOG AS POSITIVE # OF WORDS
0284  00173 007004        CMB,INB        OR CHARACTERS.
0285*
0286  00174 106700  I04   CLC 0          TURN DEVICE OFF.
0287*
0288  00175 126070R       JMP C.02,I     AND EXIT FOR COMPLETION.
0289*
0290  00176 006400  I05   CLB
0291  00177 174207        STB EQT5,I     RESET "CLEAR" FLAG.
0292  00200 126070R       JMP C.02,I     RETURN
0293*
0294* $SUBROUTINE:  <SETIO>
0295*
0296*  'URPOSE: TO CONFIGURE THE I/O INSTRUCTIONS
0297*           IN THE DRIVER TO REFERENCE THE
0298*           SUBJECT PAPER TAPE PUNCH.
0299*
0300*  CALL:       (A)05-00 CONTAINS I/O ADDRESS
0301*       (P)     JSB  SETIO
0302*       (P+1)  -RETURN- (REGISTERS MEANINGLESS)
0303*
0304  00201 000000  SETIO NOP
0305  00202 032221R       IOR LIA       COMBINE <LIA> WITH I/O ADDRESS
0306  00203 072155R       STA I03        AND SET.
0307*
0308  00204 040067        ADA .100      CONSTRUCT <OTA> INSTRUCTION
0309  00205 072127R       STA I01
0310*
0311  00206 042215R       ADA .1100     CONSTRUCT  <STC,C> INSTRUCTION
0312  00207 072130R       STA I02
0313*
0314  00210 032216R       IOR .4000     CONSTRUCT  <CLC> INSTRUCTION
0315  00211 072174R       STA I04
0316  00212 072024R       STA CLEAR
0317*
0318  00213 126201R       JMP SETIO,I
```

```
0320*
0321* CONSTANT AND VARIABLE STORAGE AREA
0322*
0323  00000           A      EQU 0          DEFINE SYMBOLIC REFERENCE FOR
0324  00001           B      EQU 1           A AND B REGISTERS.
0325*
0326  00214 000040    .40    OCT 40
0327  00215 001100    .1100  OCT 1100
0328  00216 004000    .4000  OCT 4000
0329*
0330*
0331  00217 000015    RETN   OCT 15
0332  00220 000137    ARROW  OCT 137
0333*
0334  00221 102500    LIA    LIA 0
0335*
0336  00222 000000    TEMP1  NOP
0337*
0338  00223 000061R   IEXTA  DEF IEXIT-1
0339  00224 177610    N100   DEC -120
0340*
```

```
0342*
0343*** SYSTEM BASE PAGE  COMMUNICATION  AREA ***
0344*
0345  00053          ..     EQU 53B
0346  00047          N4     EQU ..-4
0347  00054          .1     EQU ..+1
0348  00055          .2     EQU ..+2
0349  00056          .3     EQU ..+3
0350  00057          .4     EQU ..+4
0351  00061          .6     EQU ..+6
0352  00063          .10B   EQU ..+8
0353  00064          .11B   EQU ..+9
0354  00065          LINF   EQU ..+10
0355  00067          .100   EQU ..+12
0356  00073          MASK1  EQU ..+16
0357  00074          MASK3  EQU ..+17
0358  00075          MASK2  EQU ..+18
0359  00100          .      EQU 100B      ESTABLISH ORIGIN OF AREA
0360*
0361*
0362* I/O MODULE/DRIVER COMMUNICATION
0363*
0364  00203          EQT1   EQU .+67
0365  00204          EQT2   EQU .+68
0366  00205          EQT3   EQU .+69
0367  00206          EQT4   EQU .+70
0368  00207          EQT5   EQU .+71
0369  00210          EQT6   EQU .+72
0370  00211          EQT7   EQU .+73
0371  00212          EQT8   EQU .+74
0372  00213          EQT9   EQU .+75
0373  00214          EQT10  EQU .+76
0374  00215          EQT11  EQU .+77
0375  00216          EQT12  EQU .+78
0376  00217          EQT13  EQU .+79
0377  00220          EQT14  EQU .+80
0378  00221          EQT15  EQU .+81
0379  00222          EQT16  EQU .+82
0380  00223          EQT17  EQU .+83
0381                        END
**  NO ERRORS*
```

```
*       00359   00364   00365   00366   00367   00368   00369
        00370   00371   00372   00373   00374   00375   00376
        00377   00378   00379   00380

**      00345   00346   00347   00348   00349   00350   00351
        00352   00353   00354   00355   00356   00357   00358

*1      00347   00112

*100    00355   00308

*10B    00352   00125

*1100   00327   00311

*11B    00353   00127

*2      00348   00114   00132   00277

*3      00349   00110

*4      00350   00184

*.40    00326

*4000   00328   00314

*.6     00351

*A      00323

ARROW   00332   00229

B       00324   00199   00217   00270   00273

C.02    00191   00008   00175   00236   00237   00288   00292

CLEAR   00134   00124   00316

D01     00139   00126

D02     00146   00128

D03     00154   00142

D04     00159   00115

D05     00174   00155

D10     00198   00176

D11     00241   00209

D12     00248   00212
```

```
 D14     00262    00258

*EQT1    00364

 EQT10   00373    00146    00159

 EQT11   00374    00162    00282

 EQT12   00375    00140    00151    00161    00205    00214    00215
         00242    00263

 EQT13   00376    00154    00167    00211    00223    00241

 EQT14   00377    00170    00183    00252    00256

*EQT15   00378

*EQT16   00379

*EQT17   00380

*EQT2    00365

*EQT3    00366

 EQT4    00367    00271    00274

 EQT5    00368    00194    00278    00291

*EQT6    00369

*EQT7    00370

*EQT8    00371

 EQT9    00372    00109    00119    00198

 I,02    00105    00008    00113    00133    00180    00185

 I,A,4   00182    00178

 I,A,6   00184    00135

 IEXIT   00179    00338

 IEXTA   00338    00174

 IO1     00234    00224    00227    00244    00259    00264    00309

 IO2     00235    00312

 IO3     00269    00196    00206    00230    00250    00306

 IO4     00286    00315

 IO5     00290    00280
```

```
DVR02        CROSS-REFERENCE SYMBOL TABLE        PAGE 0003

 LIA     00334    00305
 LINF    00354    00152    00255
 MASK1   00356    00122
 MASK2   00358    00272
 MASK3   00357    00220
 N100    00339    00139
*N4      00346
 RETN    00331    00253    00254    00257
 SETIO   00304    00107    00193    00318
 TEMP1   00336    00202    00222    00248
```

PRIVILEGED INTERRUPT I/O DRIVERS

Privileged interrupt I/O drivers include a third relocatable, closed subroutine in addition to the Initiation Section and the Completion Section. This subroutine is the Privileged Interrupt Section. P.*nn* is the entry point name. The Initiation Section is identical to those written for the standard I/O drivers except that the EQT entry should be saved for subsequent use by the Privileged Interrupt Section. Figure 13-3 is a flowchart of the privileged interrupt driver Initiation Section.

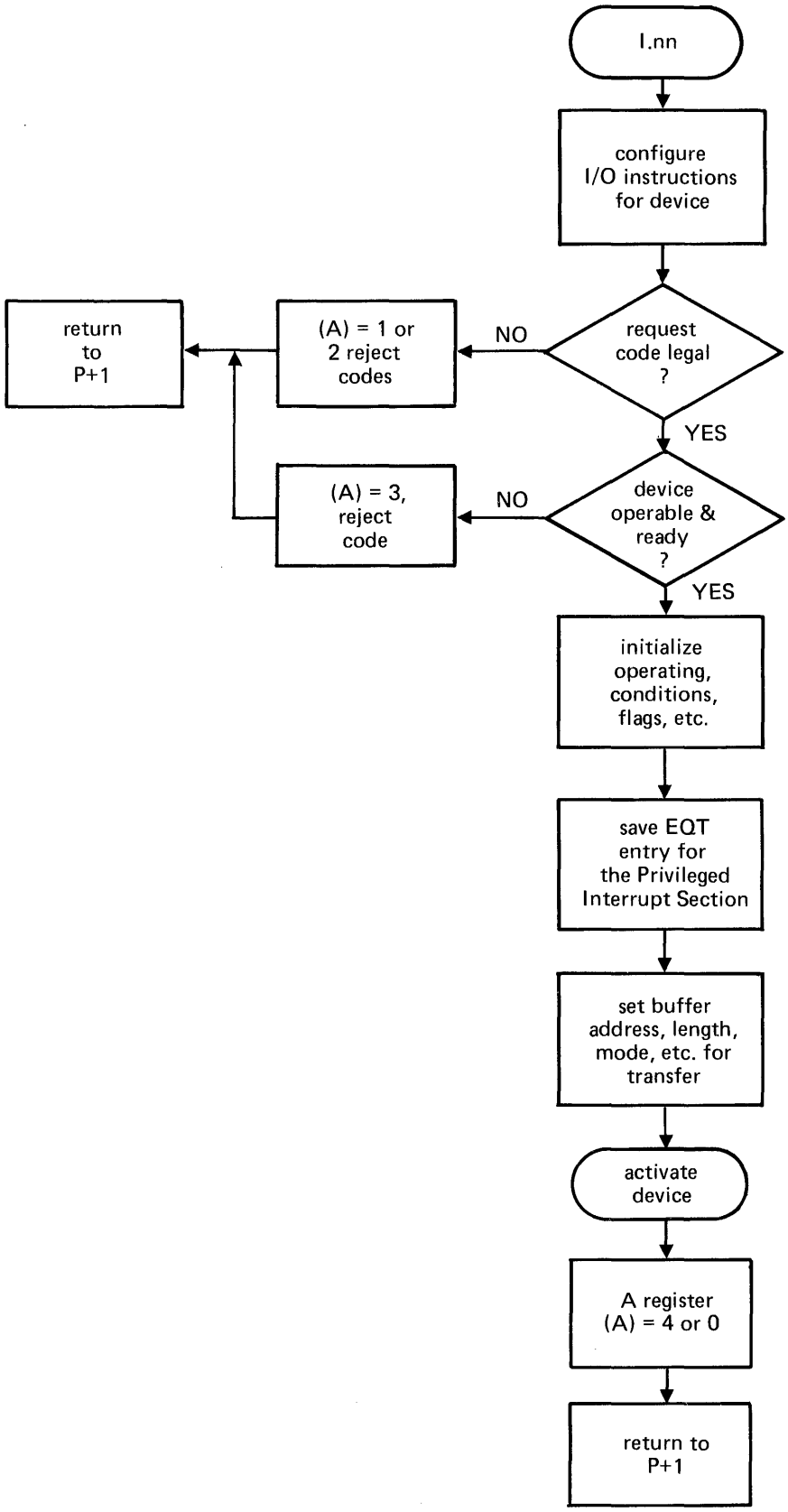


Figure 13-3. Privileged Interrupt I/O Driver Initiation Section

Privileged Interrupt Section

Control passes directly to the Privileged Interrupt Section of the driver (P.*nn*) whenever an interrupt occurs from a device associated with the driver. The address specifying where control is to be passed (that is, the P.*nn* entry point) must be included at generation time while building the interrupt table entries (the ENT option should be used; see Section 10). Since control does not pass through the system's central interrupt routine before entering the Privileged Interrupt Section, the following standard interrupt processing is not performed:

1. The I/O interface flag for the device is not cleared.
2. The A register does not contain the interrupt source code.
3. The EQT entry addresses are not set in the base page.

> *Note: To allow access to the EQT entry, the Initiation Section should save the EQT address, then the Privileged Interrupt Section can use the saved address to reference the EQT entry.*

FUNCTIONS OF THE PRIVILEGED INTERRUPT SECTION: The Privileged Interrupt Section is responsible for the following functions (flowcharted in Figure 13-4):

1. Upon entry to P.*nn*, the driver must save the contents of the A, B, E, and O registers.
2. The driver services the current data item and determines whether or not the transfer is complete.
3. If the transfer is not complete, the Privileged Interrupt Section should set the device for the next transfer and proceed to step 5.
4. If the transfer is complete, the Privileged Interrupt Section should make the following system completion call:

```
EXT   $PCOM
LDA   EQT1      (saved EQT entry)
JSB   $PCOM
```

   This call directs the system to pass control to the standard Completion Section (C.*nn* entry point) as soon as it is possible for a "system" device to interrupt.

5. Prior to returning control to the point of suspension, the Privileged Interrupt Section must restore the A, B, E, and O registers. In addition, since memory protect is automatically disabled whenever an interrupt occurs, the Privileged Interrupt Section is responsible for restoring memory protect to its original state. A memory protect flag exists on the base page (MPTFL = 271_8) to provide the driver with information concerning the state of memory protect. If MPTFL is zero, memory protect was on and an STC 5 instruction should be executed immediately prior to returning to the point of suspension. If MPTFL is one, memory protect was off and an STC 5 instruction should not be issued.

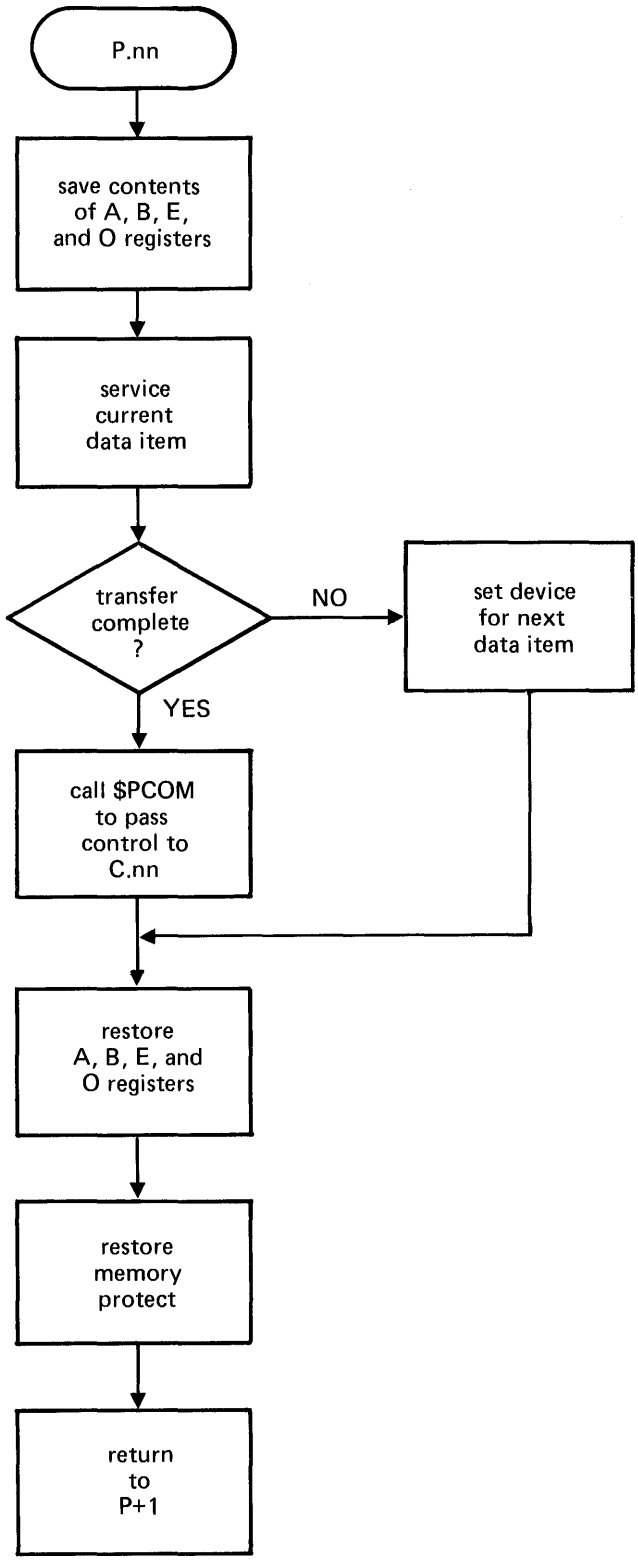


Figure 13-4. Privileged Interrupt I/O Driver Privileged Interrupt Section

Privileged Interrupt Completion Section

The completion section in a privileged interrupt driver is used to perform the following functions (flowcharted in Figure 13-5):

1. Set the actual or simulated device status into bits 0 through 7 of EQT word 4.

2. Set the number of transmitted words or characters (depending on which the user requested) in the B register.

3. Set the A register to indicate successful or unsuccessful completion.

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

4. Clear the device control on end-of-operation, or set device for next transfer. Return to $CIC at P+1.

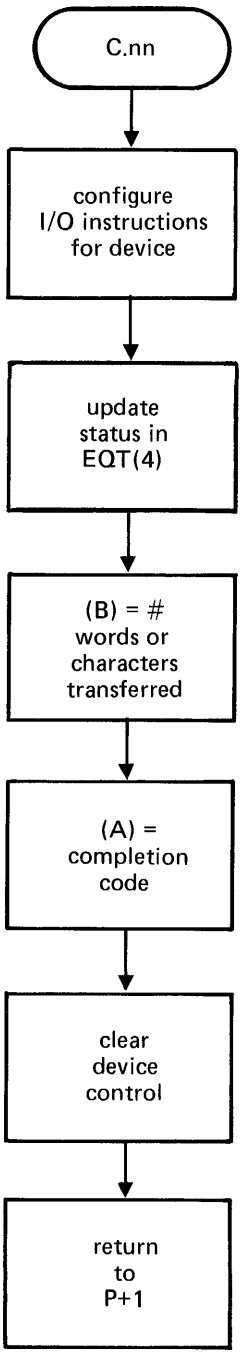


Figure 13-5. Privileged Interrupt I/O Driver Completion Section

SAMPLE PRIVILEGED INTERRUPT I/O DRIVER

The following pages provide an assembly listing and cross-reference symbol table for a sample privileged interrupt I/O driver.

```
 PAGE  0001

0001                    ASMB,R,B,L,C    PRIVILEGED DRIVER FOR PUNCH
**   NO ERRORS*
```

```
0001                ASMB,R,B,L,C   PRIVILEGED DRIVER FOR PUNCH
0003  00000               NAM DVR02,0
0004*
0005                      ENT I.02,P.02,C.02
0006*
0007                      EXT $PCOM,$MOVE
0008*
0009****** PROGRAM DESCRIPTION ******
0010*
0011*     DRIVER 02 IS A SIMPLIFIED VERSION OF THE GENERAL PURPOSE
0012* PUNCH DRIVER TO ILLUSTRATE THE USE OF PRIVILEGED INTERRUPT
0013* FENCE REGISTER.
0014*
0015*     DRIVER 02 OPERATES UNDER THE I/O CONTROL MODULE OF THE DOS
0016* EXECUTIVE FOR INITIATION AND COMPLETION AND DIRECTLY FROM THE
0017* TRAP CELL FOR PRIVILEGED INTERRUPTS.
0018*
0019*     I.02 IS THE ENTRY POINT TO THE *INITIATION* SECTION
0020*     P.02 IS THE ENTRY POINT TO THE *PRIVILEGED* SECTION
0021*     C.02 IS THE ENTRY POINT TO THE *COMPLETION* SECTION
0022*  -  THE INITIATION SECTION IS CALLED FROM I/O CONTROL TO
0023*     INITIALIZE A DEVICE AND INITIATE AN OUTPUT
0024*
0025*     CALLING SEQUENCE:
0026*
0027*          - ADDRESSES OF DEVICE EQT ENTRY SET IN "EQT1-EQT17"
0028*
0029*          (A) = I/O ADDRESS OF DEVICE
0030*
0031*          (P)    JSB I.02
0032*          (P+1)  -RETURN-
0033*                      (A) = 0, OPERATION INITIATED
0034*                          = 4, OPERATION REJECTED-IMMEDIATE COMPLETIO
0035*
0036*  -  THE PRIVILEGED SECTION IS CALLED DIRECTLY FROM THE I/O TRAP
0037*     CELL WHOSE ADDRESS HAS BEEN SET AT SYSTEM GENERATION.
0038*
0039*     CALLING SEQUENCE:
0040*
0041*          (P)    JSB P.02
0042*          (P+1)  -RETURN-
0043*
0044*  -  THE COMPLETION SECTION IS CALLED BY CENTRAL INTERRUPT
0045*     CONTROL TO COMPLETE AN OPERATION.
0046*
0047*     CALLING SEQUENCE:
0048*
0049*          - ADDRESSES OF DEVICE EQT ENTRY SET IN "EQT1-EQT17"
0050*
0051*          (A) = I/O ADDRESS OF DEVICE
0052*
0053*          (P)    JSB C.02
0054*          (P+1)  -RETURN-
0055*                      (A) = 0, SUCCESSFUL COMPLETION
0056*
0057*  -  RECORD FORMAT MUST BE A STRING OF ASCII CHARACTERS
```

```
0059*
0060*      THE FUNCTIONS OF THE INITIATION SECTION ARE:
0061*
0062*           1. CONFIGURE I/O INSTRUCTIONS
0063*           2. SAVE SYSTEM EQT ENTRY ADDRESSES USED IN PRIVILEGED
0064*              SECTION FROM EQT1-EQT17.
0065*           3. CHECK FOR LEGITIMATE REQUEST CODE
0066*           4. FORM CHARACTER BUFFER ADDRESS
0067*           5. FORM NEGATIVE CHARACTER COUNT
0068*           6. OUTPUT FIRST CHARACTER
0069*           7. ENABLE DEVICE
0070*           8. RETURN
0071*
0072*                NOTE: FUNCTION 2 IS THE MAIN DIFFERENCE FROM
0073*                      STANDARD DRIVERS
0074*
0075  00000 000000  I,02  NOP             INITIATOR SECTION ENTRY
0076  00001 016140R       JSB CONFG       CONFIGURE I/O INSTRUCITONS
0077  00002 062160R       LDA DEQT1       MOVE SYSTEM EQT1-EQT17 INTO
0078  00003 066161R       LDB N17          PRIVILEGED DRIVER EQT AREA
0079  00004 016002X       JSB $MOVE
0080  00005 000166R       DEF TEQ1
0081  00006 162175R       LDA TEQ8,I      GET REQUEST CODE
0082  00007 002020        SSA
0083  00010 003004        CMA,INA              MAKE + IF NEGATIVE
0084  00011 050055        CPA .2
0085  00012 002001        RSS
0086  00013 026042R       JMP ERTN        IGNORE REQUEST CODE RETURN
0087  00014 162177R       LDA TEQ10,I     FORM CHARACTER BUFFER ADDRESS
0088  00015 001200        RAL
0089  00016 172201R       STA TEQ12,I     *BUF ADDR USED BY DRIVER*
0090  00017 162200R       LDA TEQ11,I     FORM NEGATIVE CHARACTERS COUNTER
0091  00020 002003        SZA,RSS
0092  00021 026042R       JMP ERTN
0093  00022 002020        SSA                    ERROR IF ZERO CHARACTER
0094  00023 026026R       JMP *+3
0095  00024 001000        ALS
0096  00025 003004        CMA,INA
0097  00026 172202R       STA TEQ13,I     *CHAR COUNT USED BY DRIVER*
0098  00027 166177R       LDB TEQ10,I
0099  00030 136201R       ISZ TEQ12,I
0100  00031 160001        LDA B,I         FORM FIRST CHAARACTER
0101  00032 001727        ALF,ALF
0102  00033 010074        AND .377
0103  00034 102600  I.02A OTA 0           OUTPUT FIRST CHARACTER
0104  00035 136202R       ISZ TEQ13,I     INCR FOR FIRST CHARACTER
0105  00036 000000        NOP
0106  00037 102700  I.02B STC 0           ENABLE DEVICE
0107  00040 002400        CLA             INDICATE NORMAL RETURN
0108  00041 126000R       JMP I.02,I      RETURN TO SYSTEM
0109  00042 060057  ERTN  LDA .4          IMMEDIATE COMPLETION RETURN
0110  00043 126000R       JMP I.02,I      RETURN TO SYSTEM
```

```
0112*
0113*      PRIVILEGED PROCESSOR SECTION
0114*
0115*
0116*      THE FUNCTIONS OF THE PRIVILEGED SECTION ARE:
0117*
0118*          1. TURN OFF INTERRUPTS
0119*          2. SAVE COMPUTER REGISTERS AT INTERRUPT
0120*          3. IF ALL CHARACTERS OUTPUT, GO TO FUNCTION 10
0121*          4. OUTPUT NEXT CHARACTER
0122*          5. ENABLE DEVICE
0123*          6. RESTORE REGISTERS
0124*          7. SET MEMORY PROTECT TO ORIGINAL STATE AT TIME OF INTERRU
0125*          8. TURN ON INTERRUPTS
0126*          9. RETURN TO POINT TO INTERRUPT
0127*
0128*         10. CALL $PCOM TO ENTER DEVICE INTO PRIVILEGED INTERRUPT
0129*             COMPLETION QUEUE
0130*         11. DISABLE DEVICE
0131*         12. RETURN TO POINT OF INTERRUPT
0132*
0133  00044 000000  P.02  NOP
0134  00045 103100        CLF 0          TURN OFF INTERRUPT SYSTEM
0135  00046 016120R       JSB SEQAB       SAVE REGISTERS
0136  00047 162202R       LDA TEQ13,I     CHECK IF LAST CHARACTER SENT OUT
0137  00050 002003        SZA,RSS
0138  00051 026077R       JMP P.02D       YES, SO INITIATE COMPLETION PROC
0139  00052 166201R       LDB TEQ12,I
0140  00053 136201R       ISZ TEQ12,I     INCREMENT BUFFER ADDRESS
0141  00054 004065        CLE,ERB
0142  00055 160001        LDA B,I         PUT DATA IN A REGISTER
0143  00056 002041        SEZ,RSS         CHECK IF UPPER OR LOWER CHARACTE
0144  00057 001727        ALF,ALF             UPPER SO MOVE INTO LOW
0145  00060 010074        AND .377        MASK OFF OTHER CHARACTER
0146  00061 102600  P.02A  OTA 0           OUTPUT A CHARACTER
0147  00062 103700  P.02B  STC 0,C         ENABLE DEVICE
0148  00063 136202R       ISZ TEQ13,I     INCREMENT CHARACTER COUNT
0149  00064 000000        NOP
0150  00065 064271  P.MPT  LDB MPTFL       CHECK IF MEM PROTECT TO BE ENABL
0151  00066 006002        SZB                 YES
0152  00067 026074R       JMP MPOFF           NO
0153  00070 016130R       JSB REQAB       RESTORE REGISTERS
0154  00071 102100        STF 0           TURN ON INTERRUPTS
0155  00072 102705        STC 5           ENABLE MEMORY PROTECT
0156  00073 126044R       JMP P.02,I      RETURN TO POINT OF INTERRUPT
0157  00074 016130R MPOFF JSB REQAB       RESTORE REGISTERS
0158  00075 102100        STF 0           TURN ON INTERRUPTS
0159  00076 126044R       JMP P.02,I      RETURN TO POINT OF INTERRUPT
0160  00077 062166R P.02D LDA TEQ1       CAUSE COMPLETION INTERRUPT
0161  00100 016001X       JSB $PCOM ENTER DEVICE INTO PRIV INT COMPLETIO
0162  00101 106700  P.02E CLC 0          CLEAR DEVICE
0163  00102 026065R       JMP P.MPT       GO TO RETURN PROCESSOR
```

```
0165*
0166*      COMPLETION PROCESSOR SECTION
0167*
0168*
0169*      THE FUNCTIONS OF THE COMPLETION SECTION ARE:
0170*
0171*           1. UPDATE STATUS IN EQT4
0172*           2. SET TRANSMISSION LOG IN B
0173*           3. CLEAR A TO INDICATE OKAY COMPLETION
0174*           4. RETURN TO CENTRAL INTERRUPT PROCESSOR
0175*
0176  00103 000000  C.02   NOP
0177  00104 102500  C.02A  LIA 0           UPDATE STATUS
0178  00105 010071         AND .37
0179  00106 070001         STA B
0180  00107 162171R        LDA TEQ4,I
0181  00110 010075         AND MASK2
0182  00111 030001         IOR B
0183  00112 172171R        STA TEQ4,I     STATUS IN EQT4
0184  00113 166200R        LDB TEQ11,I   TRANSMISSION LOG IN B
0185  00114 006020         SSB
0186  00115 007004         CMB,INB
0187  00116 002400         CLA            CLEAR A TO INDICATE OKAY STATUS
0188  00117 126103R        JMP C.02,I     RETURN TO SYSTEM
```

```
0190*
0191*     SAVE A,B,E,O
0192*
0193  00120 000000  SEOAB NOP
0194  00121 072207R       STA XA          A
0195  00122 076210R       STB XB          B
0196  00123 001520        ERA,ALS
0197  00124 102201        SOC
0198  00125 002004        INA
0199  00126 072211R       STA XEO         E AND O
0200  00127 126120R       JMP SEOAB,I
0201*
0202*     RESTORE A,B,E,O
0203*
0204  00130 000000  REOAB NOP
0205  00131 062211R       LDA XEO         E AND O
0206  00132 103101        CLO
0207  00133 000036        SLA,ELA
0208  00134 102101        STF 1
0209  00135 062207R       LDA XA          A
0210  00136 066210R       LDB XB          B
0211  00137 126130R       JMP REOAB,I
0212*
0213*     CONFIGURE I/O INSTRUCTIONS
0214*
0215  00140 000000  CONFG NOP
0216  00141 070001        STA B           SAVE SELECT CODE
0217  00142 032162R       IOR OTAC
0218  00143 072034R       STA I,02A       CONFIGURE OTA SC
0219  00144 072061R       STA P,02A
0220  00145 062163R       LDA STCC
0221  00146 030001        IOR B
0222  00147 072037R       STA I,02B       CONFIGURE STC SC,C
0223  00150 072062R       STA P,02B
0224  00151 062164R       LDA CLCC
0225  00152 030001        IOR B
0226  00153 072101R       STA P,02E       CONFIGURE CLC SC
0227  00154 062165R       LDA LIAC
0228  00155 030001        IOR B
0229  00156 072104R       STA C,02A       CONFIGURE LIA SC
0230  00157 126140R       JMP CONFG,I     RETURN
```

```
0232*
0233  00001           B       EQU 1
0234*
0235  00053           ..      EQU 53B
0236  00055           .2      EQU ..+2        DEC 2
0237  00057           .4      EQU ..+4        DEC 4
0238  00071           .37     EQU ..+14       OCT 37
0239  00074           .377    EQU ..+17       OCT 377
0240  00075           MASK2   EQU ..+18       OCT 177400
0241*
0242  00100           .       EQU 100B
0243  00203           EQT1    EQU .+67        EQUIPMENT TABLE ADDRESS
0244  00271           MPTFL   EQU .+121       MEMORY PROTECT FLAG
0245*
0246  00160 000203    DEQT1   DEF EQT1
0247*
0248  00161 177757    N17     DEC -17
0249  00162 102600    OTAC    OTA 0
0250  00163 103700    STCC    STC 0,C
0251  00164 106700    CLCC    CLC 0
0252  00165 102500    LIAC    LIA 0
0253*
0254  00166 000000    TEQ1    NOP             INITIATION ADDRESS
0255  00167 000000    TEQ2    NOP             COMPLETION ADDRESS
0256  00170 000000    TEQ3    NOP             D,R,UNIT,CHANNEL
0257  00171 000000    TEQ4    NOP             AV,TYPE,STATUS
0258  00172 000000    TEQ5    NOP             -
0259  00173 000000    TEQ6    NOP             -
0260  00174 000000    TEQ7    NOP             REQUEST RETURN
0261  00175 000000    TEQ8    NOP             REQUEST CODE
0262  00176 000000    TEQ9    NOP             I/O REQUEST CONTROL WORD
0263  00177 000000    TEQ10   NOP             REQUEST BUFFER ADDRESS
0264  00200 000000    TEQ11   NOP             REQUEST BUFFER LENGTH
0265  00201 000000    TEQ12   NOP             -
0266  00202 000000    TEQ13   NOP             -
0267  00203 000000    TEQ14   NOP             -
0268  00204 000000    TEQ15   NOP             -
0269  00205 000000    TEQ16   NOP             -
0270  00206 000000    TEQ17   NOP             -
0271*
0272  00207 000000    XA      NOP             A REGISTER TEMPORARY
0273  00210 000000    XB      NOP             B REGISTER TEMPORARY
0274  00211 000000    XEO     NOP             E AND O REGISTER TEMPORARY
0275                          END
**  NO ERRORS*
```

```
$MOVE   00007    00079

$PCOM   00007    00161

.       00242    00243    00244

..      00235    00236    00237    00238    00239    00240

.2      00236    00084

.37     00238    00178

.377    00239    00102    00145

.4      00237    00109

B       00233    00100    00142    00179    00182    00216    00221
        00225    00228

C.02    00176    00005    00188

C.02A   00177    00229

CLCC    00251    00224

CONFG   00215    00076    00230

DEQT1   00246    00077

EQT1    00243    00246

ERTN    00109    00086    00092

I.02    00075    00005    00108    00110

I.02A   00103    00218

I.02B   00106    00222

LIAC    00252    00227

MASK2   00240    00181

MPOFF   00157    00152

MPTFL   00244    00150

N17     00248    00078

OTAC    00249    00217

P.02    00133    00005    00156    00159

P.02A   00146    00219
```

```
 P.02B  00147    00223
 P.02D  00160    00138
 P.02E  00162    00226
 P.MPT  00150    00163
 REOAB  00204    00153    00157    00211
 SEOAB  00193    00135    00200
 STCC   00250    00220
 TEQ1   00254    00080    00160
 TEQ10  00263    00087    00098
 TEQ11  00264    00090    00184
 TEQ12  00265    00089    00099    00139    00140
 TEQ13  00266    00097    00104    00136    00148
*TEQ14  00267
*TEQ15  00268
*TEQ16  00269
*TEQ17  00270
*TEQ2   00255
*TEQ3   00256
 TEQ4   00257    00180    00183
*TEQ5   00258
*TEQ6   00259
*TEQ7   00260
 TEQ8   00261    00081
*TEQ9   00262
 XA     00272    00194    00209
 XB     00273    00195    00210
 XEO    00274    00199    00205
```

SECTION XIV
Privileged Mode

Certain situations may arise where a user wishes to process his own errors instead of having the operating system handle them for him. In addition, there may be cases where he wishes to determine when an I/O operation (initiated without wait) is complete.

Both of these options are available with use of the system's privileged mode flag (MDFLG = location 133_8). In order to operate in this privileged mode (i.e., user processing of I/O errors and/or determining I/O completions) the user

- must be programming in Assembly language
- is responsible for setting MDFLG properly

  Bit 0 set - user error processing

  Bit 15 set - I/O completion processing

DOS-III uses MDFLG as follows:

1. After an I/O initiation (performed by an EXEC call) MDFLG bit 0 is checked, and if it is equal to one, control returns to the user program with the A register set as follows:

Contents (decimal)	Meaning
0	Operation initiated
1	Read or write illegal
2	Control request ignored
3	Device down
4	Immediate completion
5	DMA busy
6	Driver busy
7	Driver overlay area busy
8	EXEC overlay area busy
9	Operation rejected
10	Memory protect error
11	Request code error
12	Execution time exceeded
13	Spare

Contents (decimal)	Meaning
14	Illegal logical unit
15	Unassigned logical unit
16	Illegal buffer address
17	Memory wrap around
18	Illegal track address
19	File cannot be found

2. After an I/O completion, MDFLG bit 15 is checked, and if it is equal to one, control is passed to a user subroutine which must immediately follow the EXEC call. Upon entry to the routine, the B register contains the driver transmission log and the A register contains the device status as follows:

A register Contents	Meaning
0	I/O completed without errors
–1	Device was not ready
–2	End-of-tape
–3	Parity error
–4	Batch input detected a colon (:)

If the I/O completion resulted from an I/O error (not ready, parity, or end-of-tape) and the device is not the system console or the disc, bit 14 of EQT4 (the fourth word of the current Equipment Table entry) is set to indicate that the device is down.

MDFLG bit 0 is then checked, and if it is equal to one, control returns to user (thus bypassing system processing of the error).

3. During a FILE NAME SEARCH EXEC call (RCODE = 18) where the search is requested without wait, no subsequent EXEC calls are allowed. If a second EXEC call is requested during execution of a file search, the system will wait for the search to complete before processing the second EXEC call. If the user does not want the system to wait, he should set Bit 1 of MDFLG. If Bit 1 of MDFLG is set and the above condition is encountered, control will be returned to the user following the second EXEC call with the A register = 8 (EXEC busy).

4. The system clears all bits of MDFLG following any program completion.

5. An I/O calling sequence operating in privileged mode might look something like this:

```
        JSB     EXEC
        DEF     END
        DEF     RCODE
        DEF     CONWD     (must be an I/O without wait)
          .
          .
          .
END     JMP     INIT
COMP    NOP               } If present, the completion routine must be located here.
          .               } Executed following I/O completion, and should include
          .               } a check for completion errors. This routine must not
          .               } use any routine that is not re-entrant.
        JMP     COMP,I    }
          .
          .
          .
INIT                      } Executed following an I/O initiation, and should
          .               } include a check for initiation errors.
          .
          .
```

PART 5
Error Codes and Messages

SECTION XV
Halt Codes and Error Messages

This section describes the error conditions which can occur while DOS-III is being generated, loaded and operated. Error conditions are reported to the user by one of the following:

- a computer halt; the halt code is displayed in the DISPLAY register
- an error message; the message is displayed on the system console
- an error message (displayed on the system console) followed by a computer halt (halt code displayed in the DISPLAY register)
- an error code returned to a user program (by EFMP); the error code is also returned in the A register

This section contains halt code and error message tables, including corrective action (when applicable) for the following:

- DSGEN ERROR CONDITIONS
  - DSGEN Error Halts
  - DSGEN Error Messages
- DOS-III BOOTSTRAP ERROR HALTS
- DOS-III ERROR CONDITIONS
  - DOS-III Error Halts
  - DOS-III Error Messages
- EFMP ERROR CODES

Note: The ALGOL, FORTRAN and Assembler subsystems also print error messages. These subsystem error messages are documented in the SOFTWARE OPERATING PROCEDURE module "Assembler, FORTRAN and ALGOL Error Messages" (5951-1377). FORTRAN IV error messages are described in HP FORTRAN IV (5951-1321).

Table 15-1. DSGEN Error Conditions

DSGEN ERROR HALTS

Halt Code	Cause	Recovery Action
102000	Follows an irrecoverable error message.	Irrecoverable
	Generator unable to find $STRT in DISCM. DISCM is probably missing.	Irrecoverable
102002	Follows ERR02.	See ERR02 in error messages.
102003	Follows ERR03.	See ERR03 in error messages.
102004	Follows ERR04.	See ERR04 in error messages.
102007	Normal halt. Disc initialization of sub-channel has completed.	Start the computer executing to initialize another subchannel or to generate a system.
102022	Disc error after ten attempts. Disc address in A, disc status in B.	Start execution to retry ten more times. When preceded by ERR12 continues to next track.
102032	Disc not ready or disc should be unprotected. Disc address in A and disc status in B.	Ready or unprotect the disc. Start the computer executing.
102077	Normal halt. Ready to receive another program tape.	Continue generation. Enter next tape and start the computer executing.
102000	If DSGEN is above 10000_8 an impossible condition has occurred.	Either a hardware/software failure has occurred or DSGEN has overflowed its work area because the system was too large.

DSGEN ERROR MESSAGES

Messages During Initialization and Input Phases

Message	Meaning	Action
ERR01	Invalid response to initialization request.	Request is repeated. Enter valid reply.
ERR02	Checksum error on program input.	Computer halts; to try again, reposition tape to beginning of program and start the computer.
ERR03	Record out of sequence.	Same as ERR02.

Table 15-1. DSGEN Error Conditions (continued)

Message	Meaning	Action
ERR04	Illegal record type.	Same as ERR02. If input is from disc, error is irrecoverable; remove non-relocatable files from disc.
ERR05 name	Duplicate entry point.	The current entry point replaces the previous entry point.
ERR06	Invalid base page length in BCS-produced relocatable tape (must be zero).	Base page area is ignored, but memory protect error will occur if program is executed.
ERR07	Program name or entry point table overflow of available memory.	Irrecoverable error. Revise or delete programs.
ERR08 name	Duplicate program name.	The current program replaces the previous program.
Messages During the Parameter Phase		
ERR09	Parameter name error (no such program).	Enter valid parameter statement.
ERR10	Parameter type error.	Same as ERR09.
General Messages		
ERR11	System directory track overflow.	Irrecoverable.Regenerate system and reduce the value of the response to the "FIRST SYSTEM SECTOR?" message.
ERR12	Disc error during disc initialization.	Start the computer executing to bypass the faulty tracks.
ERR13	User segment precedes user main program.	Irrecoverable.
ERR14	Absolute code overlays relocatable code in the disc scratch area.	Irrecoverable. Regenerate the system and select one of the following two options: 1. Reduce number of programs being loaded 2. Load the library after all other programs are loaded. If this is not successful, increase the size of the system disc and/or lower the starting track/sector of the system.
ERR15	More than 63 subprograms called by a main program.	Revise main program (subsequent calls to subprograms are ignored).

Table 15-1. DSGEN Error Conditions (continued)

Message	Meaning	Action
ERR16	Base page linkage overflow.	Diagnostic printed once when overflow occurs. Bounds field indicates the number of words overflowed. 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 once when overflow occurs. Bounds field indicates the number of words overflowed. (Absolute code is generated beyond LWA). Revise program.
ERR19	Program overlay (current word of absolute code has identical location to previous word).	Current word is ignored (the address is printed).
ERR20	Binary DBL record overflow of internal table.	Records overlay previous DBL records (diagnostic printed for each overflow record). Revise program.
ERR21	Module containing entry point $CIC not loaded.	Irrecoverable error. Regenerate the system; include DISCM.
ERR22	Read parity/decode disc error. A register bits 8-14 show track number; bits 0-7 show sector number.	After ten attempts to read or write the disc sector, the computer halts. To try ten more times, start the computer executing.
ERR23	EQT not entered for disc-resident I/O module.	Restart at 100_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 number.	Enter valid DRT statement.
ERR28	Invalid channel number.	Enter valid INT statement.
ERR29	Channel number decreasing.	Same as ERR28.
ERR30	Invalid INT mnemonic.	Same as ERR28.
ERR31	Invalid EQT number.	Same as ERR28.
ERR33	Invalid entry point.	Same as ERR28.
ERR34	Invalid absolute value.	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.

Table 15-2. DOS-III Bootstrap Error Halts

Halt Code	Cause	Recovery Action
102011	Disc error status is in the A register. If A register contains 0, the subchannel did not contain a system.	Check that the device is ready and the proper disc cartridge is being used; then call maintenance.
102031	Same as above.	Occurs during execution of disc-resident part of Bootstrap. Check that the disc is ready; then call maintenance.

Table 15-3. DOS-III Error Conditions

DOS-III ERROR HALTS

Halt Code	Location	Cause	Recovery Action
102002 102003	location 2_8 location 3_8	Possible memory wrap-around when memory protect is not present.	Program error. Bootstrap DOS-III from the disc and correct the program.
102004	DISCM	Power has gone up or down with powerfail option present.	Bootstrap DOS-III from disc and restart.
102005	DISCM	Memory parity error occurred.	A-register contains address of word containing the parity error. Run the memory diagnostic programs, then bootstrap DOS-III from disc and restart.
102011	$EX20	Disc parity error. Halt occurs after a message is printed giving location of error.	Unprotect the disc and start the computer executing. DOS-III assigns next spare track.
102031	DVR31	Trying to write on disc cylinder that is flagged "protected" without first unprotecting the disc.	Start the computer executing to exit DVR31 with no action taken.
102077	$EX20	Follows message telling operator to protect the disc after spare track assignment.	Protect the disc and start the computer executing. DOS-III aborts the job that was running.

DOS-III ERROR MESSAGES

During the operation of DOS-III certain messages may be output on the system console. These messages may be error reports or simply informative; they are generated by various parts of DOS-III. The messages are listed 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 III. Enter correct directive on system console.[1]
BEGIN 'DEBUG' OPERATION	DEBUG	Any legal DEBUG operations may now be entered. Enter any legal DEBUG operations.
BP BND [L,U]?	LOADR	Specify the base page bounds desired for the program being loaded by the Loader. The bounds should be entered as two octal constants separated by a comma.
CHECKSUM ERROR	JOBPR	Checksum error in input to ST,R,file or ST,X,file directive. Correct tape.[1]

[1] This error causes a batch abort if the command is entered in batch mode. See "Batch Abort" in Section 1.

Table 15-3. DOS-III Error Conditions (continued)

Message	Source	Description
CW nnnnn	DISCM	In an I/O READ/WRITE EXEC call at nnnnn, buffer extends beyond memory bounds. Correct program.
DEVICE #nn DOWN	JOBPR	EQT #nn is unavailable (down). Use the UP,nn directive to make the device available. (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 disc is not recognizable by DOS-III. Must be labeled or unlabeled with :IN, or formatted with DSGEN, before using in DOS-III.
DISC GEN CODE nnnn NOT SYS GEN CODE mmmm 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.
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 OK TO PURGE?	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	Doubly defined file name found in a STORE directive (other than STORE,P); an EDIT directive with a new file name; on DD,U; or on a RENAME directive. Remove file or rename file.[1]
$END ALGOL	ALGOL	End of ALGOL compilation. No response required.
$END ASMB	ASMB	Assembly has 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.

[1] This error causes a batch abort if the command is entered in batch mode. See "Batch Abort" in Section 1.

Table 15-3. DOS-III Error Conditions (continued)

Message	Source	Description
$END ASMB PASS	ASMB	Another pass of the source program through the input device is required. Printed on the system console 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 triple colon occurred in the first 3 columns of a source statement. Check input to the EDIT program.[1]
$END,FTN[4]	FTN[4]	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 console and standard list device if a Time-base Generator is present.
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.
EOF-NO DATA STORED	JOBPR	An attempt was made to read an EOF without first reading data. A file is not created when this message is output.
EQT xx CH xx DVRxx D R Ux Sx	JOBPR	Equipment table entry output by the EQ directive. No action required.
EXTRA PARAMETERS	JOBPR	More than 15 parameters in a directive. Excess parameters are not processed.
FI nnnnn	DISCM	In a FILE READ/WRITE EXEC call (1) the file requested at nnnnn cannot be found. If nnnnn is not present, enter the file. (2) The length of the buffer requested at nnnnn extends beyond the end of the file. Correct the buffer length. Either case causes calling program to abort.
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, control is given to the JOBPR and the input is processed as a directive. nnnnn is the memory location of the input request.

[1] This error causes a batch abort if the command is entered in batch mode. See "Batch Abort" in Section 1.

Table 15-3. DOS-III Error Conditions (continued)

Message	Source	Description
IGNORED	DISCM	Input from system console during program execution cannot be processed. Correct input.
*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.[1] A file name begins with a non-alphabetic character. Rename the file.[1]
ILLEGAL DIGIT	JOBPR	In a decimal number, character is other than 0-9. Enter correct decimal number. In an octal number, digit is other than 0-7. Enter correct octal number.[1]
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.[1]
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.[1]
INPUT ERROR	DISCM	Equipment table entry number or logical unit number in :EQ, :LU, :UP or :DN is illegal. Enter correct equipment table or logical unit entry number.
INPUT :DATE, XXXXXXXXXX[,H,M]	DISCM	When system is initiated from the disc, DOS-III requires a DATE directive. The [,H,M] is ignored in DOS-III 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,mm directive.
I/O ERR NR EQT #mm	DISCM	The device #mm is not ready. To make the device available (up), use the UP,mm directive.
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 0. Call maintenance.

[1] This error causes a batch abort if the command is entered in batch mode. See "Batch Abort" in Section 1.

Table 15-3. DOS-III Error Conditions (continued)

Message	Source	Description
I/O ERR {PE / NR} USER DISC	DISCM	A parity error or device not ready occurred when attempting to assign a user disc. Disc may not be formatted; format it with DSGEN.
I/O ERR {PE / NR} USER DISK	DISCM	Disc error in completion section of DVR31. Retry previous operation.
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.
JBIN OVF	FTN [,4], ASMB, ALGOL	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 output at the beginning of each job. The time information is deleted in DOS-III if a Time-base Generator is not included in the system. Start job.
L01	LOADR	Checksum error on tape.
L02	LOADR	Illegal record.
L03	LOADR	Memory overflow.
L04	LOADR	Base page overflow.
L05	LOADR	Symbol table overflow.
L06	LOADR	Duplicate main or segment name (may be caused by attempting to run the Loader twice in one job).
L07	LOADR	Duplicate entry point.
L08	LOADR	No main or segment transfer address.
L09	LOADR	Record out of sequence.
L10	LOADR	Insufficient directory work area or user area space.
L11	LOADR	Program table overflow.
L12	LOADR	User file specified cannot be found.
L13	LOADR	Program name duplication.

Table 15-3. DOS-III Error Conditions (continued)

Message	Source	Description
L14	LOADR	Non-zero base page length.
L15	LOADR	Segment occurred before main.
L16	LOADR	Program overlay (illegal ORG).
L17	LOADR	Illegal library record.
L18	LOADR	Illegal octal digit in base page bounds specification; or the lower base page bound is greater than the upper base page bound; or the lower or upper base page bound is greater than 2000_8. In keyboard mode, re-enter new base page bounds. In batch mode, Loader aborts.
L19	LOADR	Illegal octal digit in main memory bounds specification; or the lower program bound is greater than the upper program bound. In keyboard mode, re-enter new program bounds. In batch mode, Loader aborts.
LBL = 111111	DISCM	Disc subchannel referenced is labeled 111111. 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), number of words requested exceeds number of words available (:MMGT), or beginning source statement number is greater than final statement number (:EDIT). Correct directive and re-enter.[1]
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 nnnn	DISCM	Logical unit requested by an EXEC call at nnnnn is unassigned. Program is aborted. Assign logical unit.
LOADR COMPLETE	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. Correct 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.

[1] This error causes a batch abort if the command is entered in batch mode. See "Batch Abort" in Section 1.

Table 15-3. DOS-III Error Conditions (continued)

Message	Source	Description
LU nnnnn	DISCM	Illegal logical unit in EXEC call at nnnnn. Program is aborted. Enter correct logical unit number.
LUxx EQTyy	JOBPR	Logical unit table entry; EQT #yy 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.[1]
xxxxx MISSING	DISCM	Segment xxxxx requested by an EXEC call is not in system or user directory. Job is aborted. Correct job.
MISSING PARAMETER	JOBPR	A parameter is missing in a directive. Retype the directive correctly.[1]
MP nnnnn	DISCM	Memory protect violation at location nnnnn. Program is aborted. Correct the program.
NAME *IGNORED	JOBPR	Illegal JOB name; numeric 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.[1]
NO PROGRAMS LOADED	LOADR	No programs were loaded by the Loader. Loading terminates.
NO SOURCE	JOBPR	No source statements following a /R or /I in an EDIT directive. Enter source statements after the /R or /I.[1]
NO SOURCE	ALGOL	Source file from disc not pre-set.
NO SUBSYSTEMS DEFINED	JOBPR	Informs the user that a :MM directive was attempted but no subsystems were defined during system generation.
NUMBER OVERFLO	JOBPR	An integer is too large.[1]
OR nnnnn	DISCM	I/O operation requested by EXEC call at nnnnn is rejected. Program is aborted. Check program.
OVERFLOW JBIN	JOBPR	There is not enough room in the JBIN for storing the relocatable binary output from the Assembler or compilers.[1]
PARAMETER ILLEGAL	JOBPR	A parameter of a directive is illegal. Re-enter directive.[1]
PARITY ERROR SC=m,TRK=ttt,SCTR=sss	JOBPR	Parity error during disc read or write. Call maintenance.

[1] This error causes a batch abort if the command is entered in batch mode. See "Batch Abort" in Section 1.

Table 15-3. DOS-III Error Conditions (continued)

Message	Source	Description
PAUSE xxxx	LIBR (Formatter)	Program has temporarily suspended itself. xxxx is an octal number acting as an identifier. Restart program using the GO directive.
PROG BND [L,U]?	LOADR	Enter the program bounds for the program being loaded by the Loader. The bounds consist of two octal numbers separated by a comma.
RE-ENTER STATEMENT ON TTY	JOBPR	Follows most error messages that do not cause abort. Type in the correct statement.
RQ nnnnn	DISCM	Illegal parameter 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.
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 word tracks for the compiler. Enter :OFF then purge disc of unnecessary 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.[1]
TSB DISC	DISCM	Informs user that the user disc was labeled by a non-DOS-III system. May be made DOS-III disc by labeling or unlabeling with :IN.

[1] This error causes a batch abort if the command is entered in batch mode. See "Batch Abort" in Section 1.

Table 15-3. DOS-III Error Conditions (continued)

Message	Source	Description
TURN {ON / OFF} DISC PROTECT OVERRIDE SWITCH	DISCM	Unprotect [ON] or protect [OFF] the disc.
UD nnnnn	DISCM	Unable to find user disc requested by EXEC call at nnnnn. Mount required disc and type :GO; or terminate program with :ABORT or :OFF.
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 as a parameter of a directive. Retype correct file name on the system console.[1]
subsystem name UNDEFINED	JOBPR	Undefined subsystem name as a parameter of :MMGT directive. Subsystem names must be defined at system generation.
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,0[,n].
WRONG INPUT	JOBPR	Relocatable binary input furnished for a source file request or vice-versa. Enter correct input.[1]
name: nn xx	ERR0	Library routine error code, where name is the name of the user's program, nn is the routine identifier and xx is the error type.
@	JOBPR/ DISCM	Directives may be entered. Enter desired directive.
*	DISCM	Operator attention directives may be entered. Enter desired directive.

[1] This error causes a batch abort if the command is entered in batch mode. See "Batch Abort" in Section 1.

DOS-III EFMP ERROR CODES

These error numbers are returned to the user program (in ERRNO) by the EFMP. The error numbers are also returned in the A register.

Error No.	Description
0	No errors.
1	Invalid EFMP function number.
2	Duplicate file name.
3	File name not in directory.
4	File too long for this pack.
5	Invalid record length.
6	Pack number not available (or name not in directory if a search was made on all available pack directories).
7	Invalid security code.
8	A temporary file must be opened with a CREATE function. An OPEN function can only change the Temporary Record Buffer number of the starting record number for a temporary file.
9	Buffer area specified in Exec call is not valid.
10	Invalid Record Number.
11	File not open.
12	DEFINE not previously executed or Opened-File table used in previous DEFINE has been altered. Issue a new DEFINE.
13	Backspaced beyond "start-of-file."
14	No pack space available.
15	Invalid pack number.
16	No pack number entry is available in Opened-File table.
17	Work Area space not sufficient.
18	No Opened-File table space available.
19	Invalid temporary record buffer number.
20	Invalid number of EXEC call parameters.

Error No.	Description
21	End-of-File.
22	COPY terminated.
23	Invalid argument(s).
24	Maximum number of files exceeded.
25	File already OPEN.
26	Record size larger than one-half of a temporary record buffer.
27	Pack number previously initialized.
28	Pack number not initialized.
29	Directory requested is too large.
30	Invalid number of active pack numbers.

PART 6
Appendix and Indexes

APPENDIX A
System Tables

This appendix contains figures and tables which represent the structure of the following

- Main-memory layout, including
  - main memory allocations in DOS-III
  - DOS-III base page constants
  - DOS-III base page communication area
- Disc layout, including
  - disc structure in DOS-III
  - disc directory entry format
  - disc labels
- System I/O tables, including
  - the equipment table
  - the logical unit table
  - the interrupt table

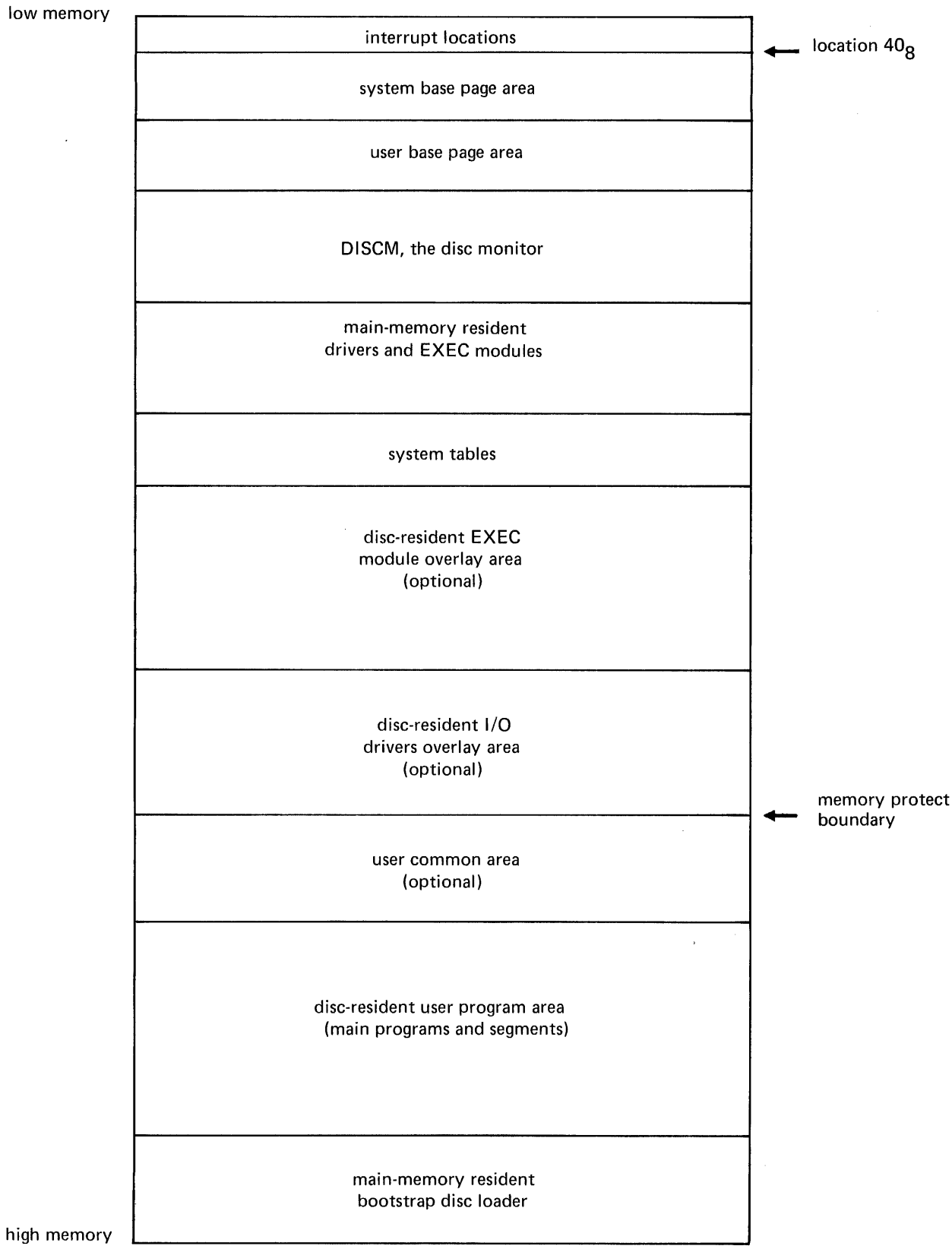


Figure A-1. Main Memory Allocations in DOS-III

Table A-1. DOS-III Base Page Constants

Location	Type	Value
40	DEC	-64
41	DEC	-10
42	DEC	-9
43	DEC	-8
44	DEC	-7
45	DEC	-6
46	DEC	-5
47	DEC	-4
50	DEC	-3
51	DEC	-2
52	DEC	-1
53	DEC	0
54	DEC	1
55	DEC	2
56	DEC	3
57	DEC	4
60	DEC	5
61	DEC	6
62	DEC	7
63	DEC	8
64	DEC	9
65	DEC	10
66	DEC	17
67	DEC	64
70	OCT	17
71	OCT	37
72	OCT	77
73	OCT	177
74	OCT	377
75	OCT	177400
76	OCT	3777
77	OCT	177700

Table A-2. DOS-III Base Page Communication Area

Location	Name	Contents
100	UMLWA	Last word address of user available memory
101	JBINS	Start track/sector of Job Binary Area
102	JBINC	Current track/sector of Job Binary Area
103	TBG	Time-base Generator I/O channel address
104-5	CLOCK	Current system clock time (2 words)
106-7	CLEX	Execution clock time (2 words)
110	CXMX	Maximum allowable execution time
111	BATCH	Logical unit # of batch input device
112	SYSTY	Logical unit # of system console
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
120	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
126-32	RONBF	Parameter buffer (5 words)
133	MDFLG	Mode flag for privileged I/O
134	DISP	(Reserved for System use)
135	AEPF	Alternate entry point flag
136	SGRTN	Segment return address
137	XIRT	System transfer address for interrupt-completion routine
140	SVEQT	EQT address for I/O operations
141-53	EXPG	Directory entry for current program (11 words)
154	DISCO	Disc I/O channel/last track on disc
155	SYSSC	System subchannel

Table A-2. DOS-III Base Page Communication Area (continued)

Location	Name	Contents
156	SCCNT	Number of subchannels on system minus 1
157	UDNTS	Next user disc track/sector
160	SYNTS	Next system disc track/sector
161	CUDSC	Current user disc subchannel
162	CRFLG	Current disc request flag: 0 for system, non-0 for user
163	CUDLA	Current user disc last access
164	FSFLG	File search flag
165	CUMID	Computer identification
166-70	DBUFR	System disc triplet parameter buffer (3 words)
171-73	UBUFR	User disc triplet parameter buffer (3 words)
174	TSONE	Last track/sector referenced +1
175	GUDSC	Default user disc subchannel
176	SYSCD	System generation code
177	JFLSC	Source file subchannel
200	DISCL	User label track/sector
201	INTAB	First word address of interrupt table
202	INT#	Number of interrupt entries
203	EQT1	EQT1-EQT17 are addresses of current equipment table entry
204	EQT2	
205	EQT3	
206	EQT4	
207	EQT5	
210	EQT6	
211	EQT7	
212	EQT8	
213	EQT9	
214	EQT10	
215	EQT11	
216	EQT12	
217	EQT13	

Table A-2. DOS-III Base Page Communication Area (continued)

Location	Name	Contents
220	EQT14	EQT1-EQT17 are addresses of current equipment table entry
221	EQT15	
222	EQT16	
223	EQT17	
224	RQCNT	Number of request parameters
225	RQRTN	Current request return address
226	RQP1	RQP1-RQP8 are addresses of current request parameters
227	RQP2	
230	RQP3	
231	RQP4	
232	RQP5	
233	RQP6	
234	RQP7	
235	RQP8	
236	NABRT	Illegal request code abort/no abort option
237	XA	A register contents at time of interrupt
240	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 memory addresses (2 words)
250-51	EXBAS	Exec module low and high base page memory addresses (2 words)
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	UBFWA	First word address of user base page area
256	UBLWA	Last word address of user base page area

Table A-2. DOS-III Base Page Communication Area (continued)

Location	Name	Contents
257	CHAN	Current DMA channel number
260	OPATN	Operator/keyboard attention flag
261	OPFLG	Operator communication flag
262	SWAP	Job processor resident flag
263-64	JOBPM	Job processor disc address/number of words in main (2 words)
265	JOBPB	Job processor base page number of words
266	EJOBF	End-job flag
267	RTRK	Real time simulation track number
270	DUMMY	Reserved for system use
271	MPTFL	Memory protect flag
272	$GOPT	Point of suspension continuation address
273	$IDCD	Input request code check
274-75	$MDBF	Exec module data buffer (2 words)
276-304	TEMP	Reserved for data communications (7 word buffer)
305	TEMP0	Reserved for System use
306	TEMP1	
307	TEMP2	
310	UTMP0	User-available Temporary
311	UTMP1	
312	UTMP2	
313	MSECT	Negative number of sectors/track
314	VADR	Address of instruction causing memory protect violation
315	IODMD	Current resident I/O driver module flag
316	RCODE	Current request code value
317	SXA	Operator attention restore A register value
320	SXB	Operator attention restore B register value
321	SXEO	Operator attention E and O register value
322	SXSUS	Operator attention return address
323	EFMP	Extended File Management Package flag

Table A-2. DOS-III Base Page Communication Area (continued)

Location	Name	Contents
324	DSCLB	Disc track/sector of Relocatable Library
325	DSCL#	Number of Relocatable Library routines
326	LSTCH	Last disc referenced
327	FLFLG/TRAC#	User file table validity flag/#Bad tracks found
330	XFLG	Entry address for disc not ready
331	SSFLG	System search flag
332	CHARC	Batch input character count
333	TYEQT	System console EQT4 address
334	DMFLG	Data Management Flag
335	SSTBL	Address of Subsystem Table
336	TMBEG	Address of Timer List

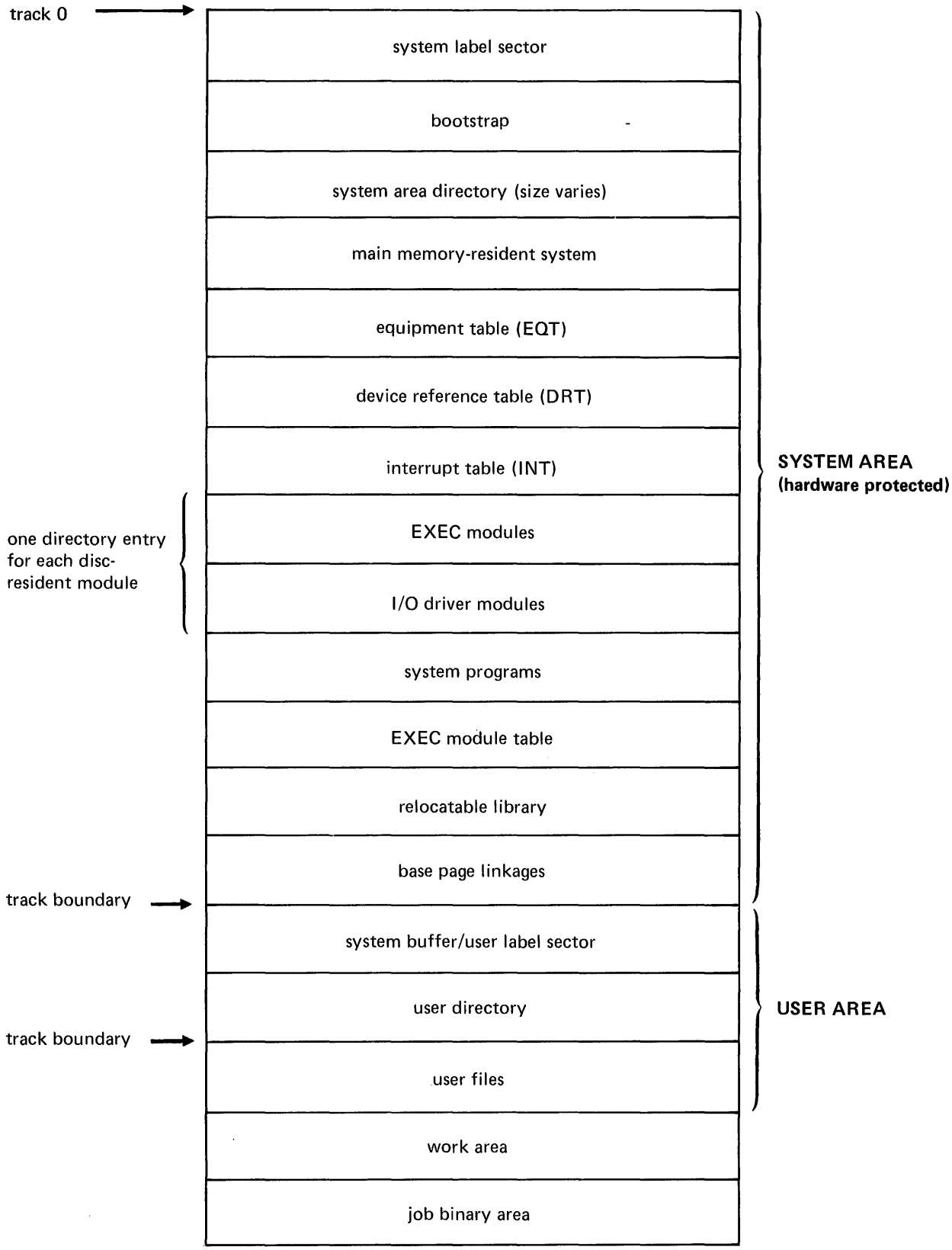
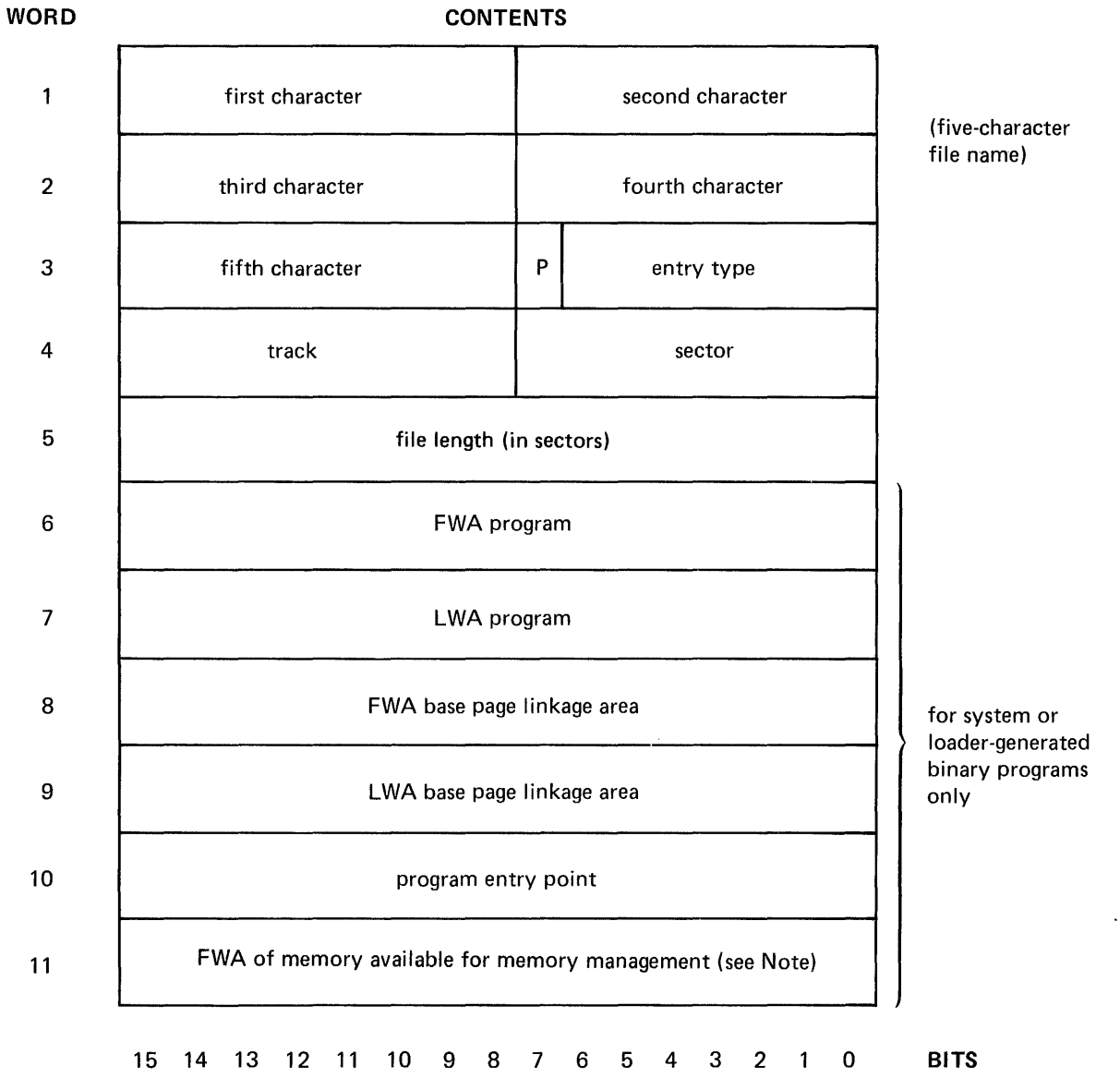


Figure A-2. Disc Structure in DOS-III



Note: For overlays, word 11 value is the last word (plus 1) of the overlay. For a main program, word 11 value is the last word (plus 1) of the largest segment.

Figure A-3. Disc Directory Entry Format

'P' Bit

0 = Permanent file—no action is taken at end-of-job.

1 = Temporary file—purge this entry at end-of-job.

This bit is set by the Relocating Loader and cleared by a STORE,P directive.

Entry Type

Type	File
0	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
14_8	Absolute binary

Note: 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 (0).

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 exists in Sector 0 of the first track after the system area. The first 64 words (words 0-63) are reserved for label information. Word 64 contains the next available track and sector. Words 65 and 66 contain the number of bad tracks and the next available spare track.

The contents of the label include:

Word 0: Label presence code (ASCII "LB" for labeled, zero for unlabeled)

Word 1: System proprietary code:

1. "DO" for DOS-III

2. "TS" for Time-shared BASIC

Word 2: System generation code assigned at system generation time. The code can be any four 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 EQUIPMENT TABLE

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

D	= 1 if DMA channel required.	
R	= 1 if driver type is main-memory resident.	
Unit #	May be used for subchannel addressing.	
Channel #	I/O select code for device (lower number if multiboard interface).	
Av	= 0	Unit not busy and available
	= 1	Unit disabled (down)
	= 2	Unit busy

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:

00-07	Paper Tape Devices
00	Teleprinter
01	Punched Tape Reader
02	High Speed Punch
04	Display Terminal
05	System Console
10-17	Unit Record Devices
11	Card Reader
12	Line Printer
20-37	Magnetic Tape/Mass Storage and other devices capable of both input and output
23	7970 Magnetic Tape
26	2762A Terminal Printer
31	Moving-Head Disc
33	Writable Control Store

For equipment type codes 01 through 17, odd numbers indicate input devices and even numbers indicate output devices (except 05, which is both input and output).

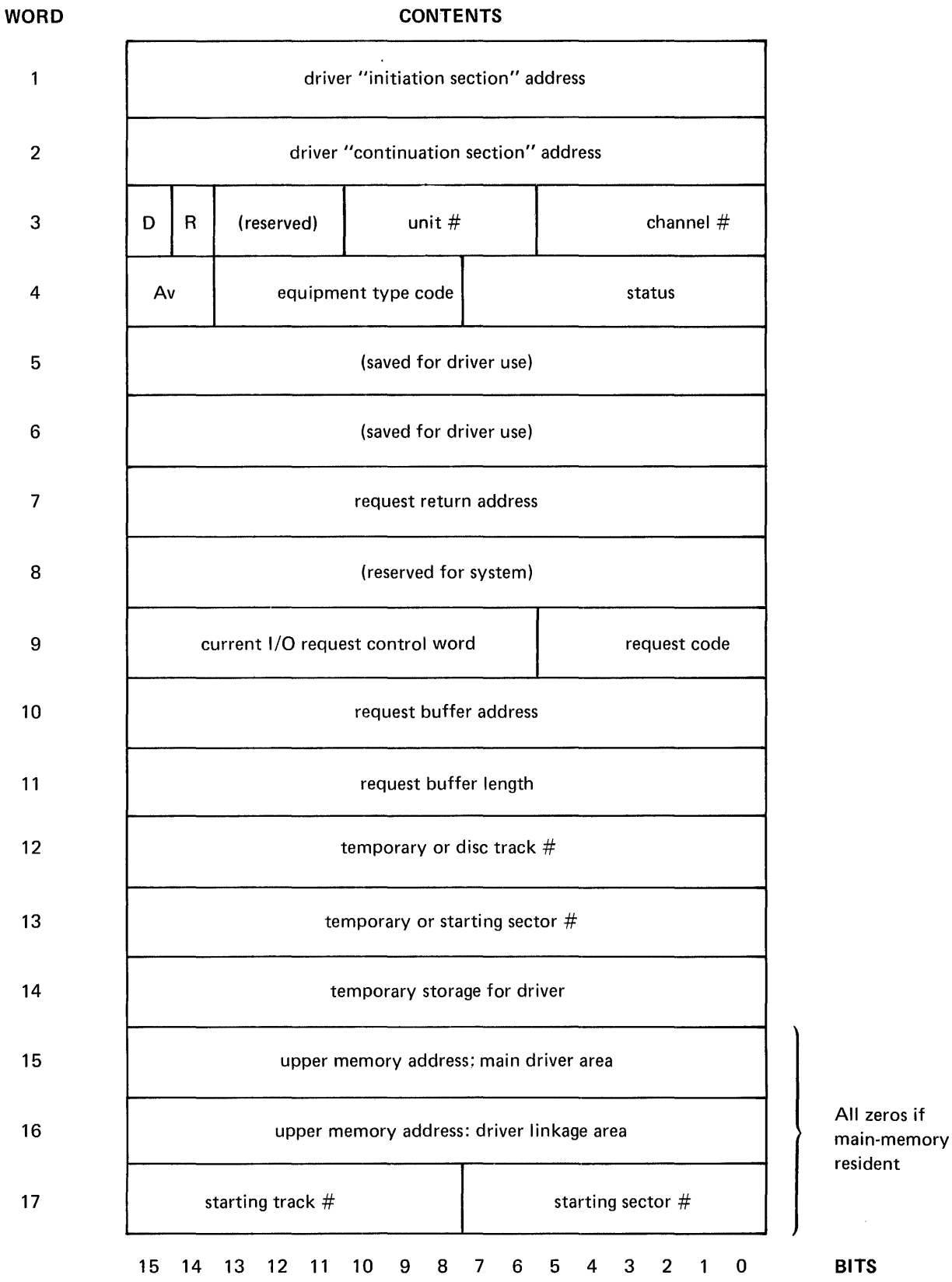


Figure A-4. The Equipment Table

THE LOGICAL UNIT TABLE

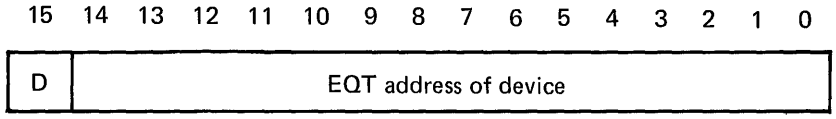
The logical unit table (LUT) has an entry for each logical unit defined at system generation time (maximum number is 63). These entries provide logical addressing of the physical devices defined in the EQT. Logical unit numbers 4-63 may be modified within a job by using the LU directive. At end-of-job, logical unit number 1-9 are restored to their original system generation values. The LUT entries reside in the permanent main-memory resident part of the system and have the following format:

Word	Contents
1	Device EQT number
.	
.	
.	
n	Device EQT number

THE INTERRUPT TABLE

The interrupt table (INT) contains an entry, established at system generation time, for each I/O channel which can cause an interrupt (beginning with I/O channel 6). The INT entries reside in the main-memory resident portion of the system and have the following format:

The entry is in the following form:



D = 0 no DMA interrupts expected

D = 1 DMA interrupts expected

Bit 15 is set and cleared by calls to $SDMA and $CDMA, respectively.

INDEX 1
Summary of Directives

Directive	Description	Page
:EJOB	Terminate the current batch and/or job normally	2-22
:EQ[,n]	List the complete equipment table, or just one line	2-23
:GO[,P_1,P_2, . . . ,P_5]	Continue processing a suspended program	2-25
:IN,label	Label or unlabel ("*") the current user disc	2-26
:JFILE,file	Specify a source file on the disc for the Assembler or a compiler	2-43
:JOB[,name]	Initiate a user job	2-28
:LIST,S,logical unit,file[,m[,n]]	List all or part of a source statement file	2-29
:LIST,U,logical unit[,$file_1$, . . .]	List all or part of the user directory	2-29
:LIST,X,logical unit[,$file_1$, . . .]	List all or part of the system directory	2-29
:LU[,n_1[,n_2]]	Assign or list logical unit assignments	2-33
:MMGT[,subsystem name,wwwww,. . . , subsystem name,wwwww]	Reserve memory address space (in words) for specific subsystems or obtain a list of previously reserved memory space	2-35
:OFF	Abort the currently executing program or operation without terminating the job	2-37
:PAUSE [comment string]	Suspend the current job or program (optionally, output a comment on the system console)	2-38
:PDUMP[,FWA[,LWA]] [,B] [,S]	Dump a program after normal completion	2-13
:PROG,name[,P_1,P_2,. . . ,P_5]	Turn on a system or user program	2-39
:PURGE[,$file_1$,$file_2$ . . .]	Delete all temporary file or specified user file directory entries.	2-40
:RNAME,oldname,newname[,type]	Rename a specified user file and optionally, change its program type	2-42
:RPACK	Repack disc user file area eliminating purged files (see :PURGE directive)	2-44
:RWND[,logical unit]	Rewind a magnetic tape	2-43
:RUN,name[,time] [,N]	Run a user program	2-45
:SA,track,sector[,number]	Dump disc in ASCII to standard list device	2-15
:SO,track,sector[,number]	Dump disc in octal to standard list device	2-15
:SS	Set up system search for file names over all subchannels	2-52
:SS,n_1,n_2 . . .	Set up system search for file names over specified subchannels	2-52

Directive	Description	Page
:SS,99	Restrict search for file names to current user disc (plus system directory for RUN and PROG)	2-52
:STORE,A,file,sectors	Reserve space for an ASCII data file	2-47
:STORE,B,file,sectors	Reserve space for a binary data file	2-47
:STORE,P[,name$_1$,name$_2$ . . .]	Store all or specified temporary Loader-Generated programs as permanent files	2-47
:STORE,R,file[,logical unit]	Store a relocatable file from the JBIN area of disc after an assembly or compilation or from a peripheral I/O device	2-47
:STORE,S,file,logical unit[,C]	Store a source statement file from a peripheral I/O device	2-47
:STORE,X,file,logical unit	Store absolute binary programs	2-47
:TOF[,logical unit]	Issue a top-of-form to a list device	2-54
:TRACKS	Print the disc track status of the current user disc	2-55
:TYPE	Return to keyboard mode from batch mode	2-57
:UD[,[label] [,n]]	Change the subchannel assignment for the user disc, or request label and subchannel information for a user disc	2-59
:UP,n	Declare an I/O device up	2-58

INDEX 2
Summary of EXEC Calls

Consult Section III for the complete details on each EXEC call.

RCODE	Name	Function	Page
-19	BASE PAGE STORE	Store values into base page memory locations (Value to be stored in the A register, absolute location address in the B register)	3-6
1, 2	I/O READ/WRITE	Transfer input or output (1 = read or 2 = write)	3-20
3	I/O CONTROL	Carry out control operations	3-17
6	PROGRAM COMPLETION	Signal end of program	3-30
7	PROGRAM SUSPENSION	Suspend calling program	3-33
8	SEGMENT LOAD	Load segment of calling program	3-35
10	PROGRAM LOAD	Transfer a main program into main memory	3-31
11	TIME REQUEST	Request the time-of-day	3-38
13	I/O STATUS	Request device status	3-23
14, 15	FILE READ/WRITE	Read or write a user data file (14 = read or 15 = write)	3-13
16	WORK AREA STATUS	Ascertain if *n* contiguous work tracks are available	3-41
17	WORK AREA LIMITS	Ascertain first and last tracks of work area	3-39
18	FILE NAME SEARCH	Ascertain if a file name exists in the directory	3-9

RCODE	Name	Function	Page
23	USER DISC CHANGE	Change the current user disc subchannel	3-43
24	EFMP CALLS	Execute EFMP functions	Section VII
27, 28	USER EXEC CALLS	Execute user EXEC modules $EX36 or $EX37 (RCODE = 27 for $EX36; RCODE = 28 for $EX37; up to five words of parameter information)	Section XII
29	SEGMENT RETURN	Return from a segment to the main program at the instruction immediately following the segment load call	3-37
30	MEMORY PROTECT CONTROL	Control memory protect from a user program	3-29
31	(Reserved for future assignment)		
32	FILE CREATE	Allows user to create a user disc file under program control.	3-7
33	FILE PURGE	Allows user to purge a user disc file under program control.	3-11
34	FILE RENAME	Allows user to rename a user disc file under program control.	3-15
35	MEMORY MANAGE-MENT (INITIALIZE)	Reserves a block of memory under a unique block name identifier specified by the user.	3-26
36	MEMORY MANAGE-MENT (STATUS REQUEST)	Requests number of words allocated to specified block name identifier, or number of remaining unallocated words if block name identifier is omitted.	3-28
37	(Reserved for future assignment)		
38	MEMORY MANAGE-MENT (BUFFER ALLOCATION)	Allocates buffer area from memory space. If the block name identifier is specified, the buffer allocation is from the area reserved for the block name. If not, the allocation is from the available memory area.	3-24
39	(Reserved for future assignment)		
40	(Reserved for future assignment)		
41	MEMORY MANAGE-MENT (BUFFER RELEASE)	Permanently releases buffer space. If the buffer resides within an area reserved under a block name identifier, the logical address space remains reserved.	3-25

INDEX 3
Index of Terms

U

W

$

/

WORLD WIDE SALES & SERVICE OFFICES

UNITED STATES

ALABAMA
8290 Whitesburg Dr., S.E.
P.O. Box 4207
Huntsville 35802
Tel: (205) 881-4591
TWX: 810-726-2204

***Birmingham**
Medical Service only
Tel: (205) 879-2081

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 244-1361
TWX: 910-951-1331

2424 East Aragon Rd.
Tucson 85706
Tel: (602) 889-4661

CALIFORNIA
1430 East Orangethorpe Ave.
Fullerton 92631
Tel: (714) 870-1000
TWX: 910-592-1288

3939 Lankershim Boulevard
North Hollywood 91604
Tel: (213) 877-1282
TWX: 910-499-2170

6305 Arizona Place
Los Angeles 90045
Tel: (213) 649-2511
TWX: 910-328-6147

***Los Angeles**
Tel: (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel: (408) 249-7000
TWX: 910-338-0518

***Ridgecrest**
Tel: (714) 446-6165

2220 Watt Ave.
Sacramento 95825
Tel: (916) 482-1463
TWX: 910-367-2092

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200
TWX: 910-335-2000

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel: (303) 771-3455
TWX: 910-935-0705

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel: (203) 389-6551
TWX: 710-465-2029

FLORIDA
P.O. Box 24210
2806 W. Oakland Park Blvd.
Ft. Lauderdale 33307
Tel: (305) 731-2020
TWX: 510-955-4099

***Jacksonville**
Medical Service only
Tel: (904) 725-6333

P.O. Box 13910
6177 Lake Ellenor Dr.
Orlando 32809
Tel: (305) 859-2900
TWX: 810-850-0113

21 East Wright St.
Suite 1
Pensacola 32501
Tel: (904) 434-3081

GEORGIA
P.O. Box 28234
450 Interstate North
Atlanta 30328
Tel: (404) 434-4000
TWX: 810-766-4890

HAWAII
2875 So. King Street
Honolulu 96814
Tel: (808) 955-4455

ILLINOIS
(Calculators Only)
100 S. Wacker Drive
Suite 1100
Chicago 60606
Tel: (312) 346-9701

5500 Howard Street
Skokie 60076
Tel: (312) 677-0400
TWX: 910-223-3613

***St. Joseph**
Tel: (217) 469-2133

INDIANA
3839 Meadows Drive
Indianapolis 46205
Tel: (317) 546-4891
TWX: 810-341-3263

IOWA
1902 Broadway
Iowa City 52240
Tel: (319) 338-9466
Night: (319) 338-9467

***KANSAS**
Derby
Tel: (316) 267-3655

LOUISIANA
P.O. Box 840
3239 Williams Boulevard
Kenner 70062
Tel: (504) 721-6201
TWX: 810-955-5524

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel: (301) 944-5400
TWX: 710-862-9157

4 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9685
710-828-0487

P.O. Box 1648
2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave.
Lexington 02173
Tel: (617) 861-8960
TWX: 710-326-6904

MICHIGAN
23855 Research Drive
Farmington 48024
Tel: (313) 476-6400
TWX: 810-242-2900

MINNESOTA
2400 N. Prior Ave.
Roseville 55113
Tel: (612) 636-0700
TWX: 910-563-3734

MISSISSIPPI
***Jackson**
Medical Service only
Tel: (601) 982-9363

MISSOURI
11131 Colorado Ave.
Kansas City 64137
Tel: (816) 763-8000
TWX: 910-771-2087

148 Weldon Parkway
Maryland Heights 63043
Tel: (314) 567-1455
TWX: 910-764-0830

NEBRASKA
(Medical Only)
11902 Elm Street
Suite 4C
Omaha 68144
Tel: (402) 333-6017

***NEVADA**
Las Vegas
Tel: (702) 382-5777

NEW JERSEY
W. 120 Century Rd.
Paramus 07652
Tel: (201) 265-5000
TWX: 710-990-4951

NEW MEXICO
P.O. BOx 8366
Station C
6501 Lomas Boulevard N.E.
Albuquerque 87108
Tel: (505) 265-3713
TWX: 910-989-1665

156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2485
TWX: 910-983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel: (518) 458-1550
TWX: 710-441-8270

New York City
Manhattan, Bronx
Contact Paramus, NJ Office
Tel: (201) 265-5000
Brooklyn, Queens, Richmond
Contact Woodbury, NY Office
Tel: (516) 921-0300

201 South Avenue
Poughkeepsie 12601
Tel: (914) 454-7330
TWX: 510-248-0012

39 Saginaw Drive
Rochester 14623
Tel: (716) 473-9500
TWX: 510-253-5981

5858 East Molloy Road
Syracuse 13211
Tel: (315) 455-2486
TWX: 710-541-0482

1 Crossways Park West
Woodbury 11797
Tel: (516) 921-0300
TWX: 510-221-2168

NORTH CAROLINA
P.O. Box 5188
1923 North Main Street
High Point 27262
Tel: (919) 885-8101
TWX: 510-926-1516

OHIO
16500 Sprague Road
Cleveland 44130
Tel: (216) 243-7300
Night: 243-7305
TWX: 810-423-9431

330 Progress Rd.
Dayton 45449
Tel: (513) 859-8202
TWX: 810-459-1925

1041 Kingsmill Parkway
Columbus 43229
Tel: (614) 436-1041

OKLAHOMA
P.O. Box 32008
Oklahoma City 73132
Tel: (405) 721-0200
TWX: 910-830-6862

OREGON
17890 SW Boones Ferry Road
Tualatin 97062
Tel: (503) 620-3350
TWX: 910-467-8714

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400
Night: 782-0401
TWX: 710-795-3124

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
TWX: 510-660-2670

SOUTH CAROLINA
6941-0 N. Trenholm Road
Columbia 29260
Tel: (803) 782-6493

TENNESSEE
***Memphis**
Medical Service only
Tel: (901) 274-7472

***Nashville**
Medical Service only
Tel: (615) 244-5448

TEXAS
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75080
Tel: (214) 231-6101
TWX: 910-867-4723

P.O. Box 27409
6300 Westpark Drive
Suite 100
Houston 77027
Tel: (713) 781-6000
TWX: 910-881-2645

205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241
TWX: 910-871-1170

UTAH
2890 South Main Street
Salt Lake City 84115
Tel: (801) 487-0715
TWX: 910-925-5681

VIRGINIA
***Norfolk**
Medical Service only
Tel: (804) 497-1026

P.O. Box 9854
2914 Hungary Springs Road
Richmond 23228
Tel: (804) 285-3431
TWX: 710-956-0157

WASHINGTON
Bellefield Office Pk.
1203-114th SE
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2446

***WEST VIRGINIA**
Charleston
Tel: (304) 345-1640

WISCONSIN
9431 W. Beloit Road
Suite 117
Milwaukee 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office nearest you: Atlanta, Georgia... North Hollywood, California... Rockville, (4 Choke Cherry Rd.) Maryland...Skokie, Illinois. Their complete addresses are listed above.

***Service Only**

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11748 Kingsway Ave.
Edmonton T5G 0X5
Tel: (403) 452-3670
TWX: 610-831-2431

Hewlett-Packard (Canada) Ltd.
915-42 Avenue S.E. Suite 102
Calgary T2G 1Z1
Tel: (403) 287-1672

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd.
837 E. Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059

MANITOBA
Hewlett-Packard (Canada) Ltd.
513 Century St.
St. James
Winnipeg R3H 0L8
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd.
800 Windmill Road
Dartmouth B3C 1L1
Tel: (902) 469-7820

ONTARIO
Hewlett-Packard (Canada) Ltd.
1785 Woodward Dr.
Ottawa K2C 0P9
Tel: (613) 225-6530
TWX: 610-562-8968

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga L4V 1L9
Tel: (416) 678-9430
TWX: 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd.
275 Hymus Blvd.
Pointe Claire H9R 1G7
Tel: (514) 697-4232
TWX: 610-422-3022
TLX: 05-821521 HPCL

Hewlett-Packard (Canada) Ltd.
2376 Galvani Street
Ste-Foy G1N 4G4
Tel: (418) 688-8710

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada) Ltd. in Mississauga.

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina S.A.C.e.I
Lavalle 1171-3° Piso
Buenos Aires
Tel: 35-0436, 35-0627, 35-0341
Telex: 012-1009
Cable: HEWPACK ARG

BOLIVIA
Stambuk & Mark (Bolivia) Ltda.
Av. Mariscal, Santa Cruz 1342
La Paz
Tel: 40626, 53163, 52421
Telex: 3560014
Cable: BUKMAR

BRAZIL
Hewlett-Packard Do Brasil I.E.C. Ltda.
Rua Frei Caneca, 1.152-Bela Vista
01307-**São Paulo**-SP
Tel: 288-71-11, 287-81-20, 287-61-93
Telex: 309151/2/3
Cable: HEWPACK São Paulo

Hewlett-Packard Do Brasil I.E.C. Ltda.
Praca Dom Feliciano, 78-8° andar (Sala 806/8)
9000-**Pôrto Alegre**-RS
Tel: 25-84-70-DDD (0512)
Cable: HEWPACK Pôrto Alegre

Hewlett-Packard Do Brasil I.E.C. Ltda.
Rua Siqueira Campos, 53, 4° andar Copacabana
2000-**Rio de Janeiro**-GB
Tel: 257-80-94-DDD (021)
Telex: 2100 79 HEWPACK
Cable: HEWPACK Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda.
Calle Lira 81, Oficina 5
Casilla 2118
Santiago
Tel: 398613
Cable: CALMET

COLOMBIA
Instrumentación Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-59
Apartado Aéreo 6287
Bogota, 1 D.E.
Tel: 45-78-06, 45-55-46
Cable: AARIS Bogota
Telex: 44400INSTCO

COSTA RICA
Lic. Alfredo Gallegos Gurdián
Apartado 10159
San José
Tel: 21-86-13
Cable: GALGUR San José

GUATEMALA
IPESA
Avenida La Reforma 3-48, Zona 9
Guatemala
Tel: 63627, 64736
Telex: 4192 TELTRO GU

MEXICO
Hewlett-Packard Mexicana, S.A. de C.V.
Torres Adalid No. 21, 11° Piso
Col. del Valle
Mexico 12, D.F.
Tel: (905) 543-42-32
Telex: 017-74-507

Hewlett-Packard Mexicana, S.A. de C.V.
Ave. Constitución No. 2184
Monterrey, N.L.
Tel: 48-71-32, 48-71-84

NICARAGUA
Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel: 3451, 3452
Cable: ROTERAN Managua

PANAMA
Electrónico Balboa, S.A.
P.O. Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel: 64-2700
Telex: 3431103 Curunda, Canal Zone
Cable: ELECTRON Panama

PARAGUAY
Z.J. Melamed S.R.L.
Division: Aparatos y Equipos Medicos
Division: Aparatos y Equipos Scientificos y de Investigacion
P.O. Box 676
Chile, 482, Edificio Victoria
Asunción
Tel: 4-5069, 4-6272
Cable: RAMEL

PERU
Compañia Electro Médica S.A.
Ave. Enrique Canaual 312
San Isidro
Casilla 1030
Lima
Tel: 22-3900
Cable: ELMED Lima

PUERTO RICO
San Juan Electronics, Inc.
P.O. Box 5167
Ponce de León 154
Pda. 3-PTA de Tierra
San Juan 00906
Tel: (809) 725-3342, 722-3342
Cable: SATRONICS San Juan
Telex: SATRON 3450 332

URUGUAY
Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370

Montevideo
Tel: 40-3102
Cable: RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela C.A.
Apartado 50933
Edificio Segre
Tercera Transversal
Los Ruices Norte
Caracas 107
Tel: 35-00-11
Telex: 21146 HEWPACK
Cable: HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard Inter-Americas
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1260
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

EUROPE

AUSTRIA
Hewlett-Packard Ges.m.b.H.
Handelska 52/3
P.O. Box 7
A-1205 **Vienna**
Tel: (0222) 33 66 06 to 09
Cable: HEWPAK Vienna
Telex: 75923 hewpak a

BELGIUM
Hewlett-Packard Benelux S.A./N.V.
Avenue de Col-Vert, 1, (Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494 paloben bru

DENMARK
Hewlett-Packard A/S
Datavej 52
DK-3460 **Birkerød**
Tel: (01) 81 66 40
Cable: HEWPACK AS
Telex: 166 40 hp as

Hewlett-Packard A/S
Navervej 1
DK-8600 **Silkeborg**
Tel: (06) 82 71 66
Telex: 166 40 hp as
Cable: HEWPACK AS

FINLAND
Hewlett-Packard Oy
Nahkahousuntie 5
P.O. Box 6
SF-00211 **Helsinki** 21
Tel: 6923031
Cable: HEWPACKOY Helsinki
Telex: 12-15363

FRANCE
Hewlett-Packard France
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401 **Orsay**
Tel: (1) 907 78 25
Cable: HEWPACK Orsay
Telex: 60048

Hewlett-Packard France
Agence Régional
Chemin des Mouilles
Boite Postale No. 12
F-69130 **Ecully**
Tel: (78) 33 81 25, 83 65 25
Telex: 31 617

Hewlett-Packard France
Agence Régionale
Zone Aéronautique
Avenue Clément Ader
F-31770 **Colomiers**
Tel: (61) 78 11 55
Telex: 51957

Hewlett-Packard France
Agence Régionale
Centre d'aviation générale
F-13721 **Aéroport de Marignane**
Tel: (91) 89 12 36
TWX: 41770 F

Hewlett-Packard France
Agence Régionale
63, Avenue de Rochester
F-35000 **Rennes**
Tel: 74912 F
Telex: 74 912 F

Hewlett-Packard France
Agence Régionale
74, Allée de la Robertsau
F-67000 **Strasbourg**
Tel: (88) 35 23 20/21
Telex: 89141
Cable: HEWPACK STRBG

GERMAN FEDERAL REPUBLIC
Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Bernerstrasse 117
Postfach 560 140
D-6000 **Frankfurt** 56
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Telex: 41 32 49 fra

Hewlett-Packard GmbH
Vertriebsbüro Böblingen
Herrenbergerstrasse 110
D-7030 **Böblingen**, Württemberg
Tel: (07031) 66 72 87
Cable: HEPAK Böblingen
Telex: 72 65 739 bbn

Hewlett-Packard GmbH
Vertriebsbüro Düsseldorf
Vogelsanger Weg 38
D-4000 **Düsseldorf**
Tel: (0211) 63 80 31/5
Telex: 85/86 533 hpdd d

Hewlett-Packard GmbH
Vertriebsbüro Hamburg
Wendenstrasse 23
D-2000 **Hamburg** 1
Tel: (040) 24 13 93
Cable: HEWPACKSA Hamburg
Telex: 21 63 032 hphh d

Hewlett-Packard GmbH
Vertriebsbüro Hannover
Mellendorfer Strasse 3
D-3000 **Hannover-Kleefeld**
Tel: (0511) 55 60 46
Telex: 092 3259

Hewlett-Packard GmbH
Vertriebsbüro Nuremberg
Hersbrückerstrasse 42
D-8500 **Nuremberg**
Tel: (0911) 57 10 66
Telex: 623 860

Hewlett-Packard GmbH
Vertriebsbüro München
Unterhachinger Strasse 28
ISAR Center
D-8012 **Ottobrunn**
Tel: (089) 601 30 61/7
Telex: 52 49 85
Cable: HEWPACKSA München

(West Berlin)
Hewlett-Packard GmbH
Vertriebsbüro Berlin
Keith Strasse 2-4
D-1000 **Berlin** 30
Tel: (030) 24 90 86
Telex: 18 34 05 hpbln d

GREECE
Kostas Karayannis
18, Ermou Street
GR-**Athens** 126
Tel: 3230-303 Sales/SVC
3230-305 Adm. Order Proc.
Cable: RAKAR Athens
Telex: 21 59 62 rkar gr

Hewlett-Packard S.A.
Mediterranean & Middle East Operations
35 Kolokotroni Street
Platia Kefallariou
Gr-Kifissia-**Athens**
Tel: 8080337, 8080359, 8080429, 8018693
Telex: 21 6588
Cable: HEWPACKSA Athens

IRELAND
Hewlett-Packard Ltd.
King Street Lane
Winnersh, Workingham
GB-**Berkshire** RG11 5AR
Tel: Wokingham 784774
Telex: 847178/848179

Hewlett-Packard Ltd.
"The Graftons"
Stamford New Road
GB-**Altrincham**, Cheshire
Tel: (061) 928-9021
Telex: 668068

ITALY
Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci 2
I-20124 **Milan**
Tel: (2) 6251 (10 lines)
Cable: HEWPACKIT Milan
Telex: 32046

Hewlett-Packard Italiana S.p.A.
Via Pietro Maroncelli 40
(ang. Via Visentin)
I-35100 **Padova**
Tel: 66 40 62/66 31 88
Telex: 32046 via Milan

Hewlett-Packard Italiana S.p.A.
Via Medaglie d'Oro, 2
I-56100 **Pisa**
Tel: (050) 500022
Telex: 32046 via Milan

Hewlett-Packard S.p.A.
Via G. Armellini 10
I-00143 **Rome**-Eur
Tel: (6) 5912544/5
Telex: 61514
Cable: HEWPACKIT Rome

Hewlett-Packard Italiana S.p.A.
Via San Quintino, 46
I-10121 **Turin**
Tel: (11) 53 82 64
Telex: 32046 via Milan

LUXEMBURG
Hewlett-Packard Benelux S.A./N.V.
Avenue de Col-Vert, 1, (Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494

NETHERLANDS
Hewlett-Packard Benelux N.V.
Weerdestein 117
P.O. Box 7825
NL-**Amsterdam**, 1011
Tel: (020) 5411522
Cable: PALOBEN Amsterdam
Telex: 13 216 hepa nl

NORWAY
Hewlett-Packard Norge A/S
Nesveien 13
Box 149
N-1344 **Haslum**
Tel: (02) 53 83 60
Telex: 16621 hpnas n

PORTUGAL
Telectra-Empresa Técnica de Equipamentos Eléctricos S.a.r.l.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
P-**Lisbon** 1
Tel: (19) 68 60 72
Cable: TELECTRA Lisbon
Telex: 12598

SPAIN
Hewlett-Packard Española, S.A.
Jerez No. 3
E-**Madrid** 16
Tel: 458 26 00
Telex: 23515 hpe

Hewlett-Packard Española, S.A.
Milanesado 21-23
E-**Barcelona** 17
Tel: (3) 2036200-08, 2044098/9
Telex: 52603 hpbe e

Hewlett-Packard Española, S.A.
Av Ramon y Cajal, 1
Edificio Sevilla I, planta 9°
E-**Seville**
Tel: 64 44 54/58

Hewlett-Packard Española S.A.
Edificio Albia II 7° B
E-**Bilbao**
Tel: 23 83 06/23 82 06

SWEDEN
Hewlett-Packard Sverige AB
Enighetsvägen 1-3
Fack
S-161 20 **Bromma** 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS Stockholm
Telex: 10721

Hewlett-Packard Sverige AB
Hagakersgatan 9C
S-431 41 **Mölndal**
Tel: (031) 27 68 00/01
Telex: Via Bromma

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zürcherstrasse 20
P.O. Box 64
CH-8952 Schlieren **Zurich**
Tel: (01) 98 18 21
Cable: HPAG CH
Telex: 53933 hpag

Hewlett-Packard (Schweiz) AG
9, chemin Louis-Pictet
CH-1214 Vernier-**Geneva**
Tel: (022) 41 49 50
Cable: HEWPACKSA Geneva
Telex: 27 333 hpsa ch

TURKEY
Telekom Engineering Bureau
Saglik Sok No. 15/1
Ayaspasa-Beyoglu
P.O. Box 437 Beyoglu
TR-**Istanbul**
Tel: 49 40 40
Cable: TELEMATION Istanbul

UNITED KINGDOM
Hewlett-Packard Ltd.
King Street Lane
Winnersh, Wokingham
GB-**Berkshire** RG11 5AR
Tel: Workingham 784774
Telex: 847178/848179

Hewlett-Packard Ltd.
"The Graftons"
Stamford New Road
GB-**Altrincham**, Cheshire
Tel: (061) 928-9021
Telex: 668068

Hewlett-Packard Ltd.
c/o Makro
South Service Wholesale Centre
Amber Way
Halesowen Industrial Estate
GB-**Halesowen**, Worcs
Tel: Birmingham 7860

Hewlett-Packard Ltd.
4th Floor
Wedge House
799, London Road
GB-**Thornton Heath** CR4 6XL, Surrey
Tel: (01) 684 0105
Telex: 946825

Hewlett-Packard Ltd.
c/o Makro
South Service Wholesale Centre
Wear Industrial Estate
Washington
GB-**New Town**, County Durham
Tel: Washington 464001 ext. 57/58

Hewlett-Packard Ltd.'s registered address for V.A.T. purposes only:
70, Finsbury Pavement
London, EC2A1SX
Registered No. 690597

USSR
Hewlett-Packard
Representative Office USSR
Hotel Budapest/Room 201
Petrovskie Linii 2/18
Moscow
Tel: 221-79-71

YUGOSLAVIA
Iskra-Standard/Hewlett-Packard
Topniska 58/3
61000 **Ljubljana**
Tel: 314561 or 314927
Telex: 31300

SOCIALIST COUNTRIES PLEASE CONTACT:
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box 349
CH-1217 Meyrin 1 **Geneva** Switzerland
Tel: (022) 41 54 00
Cable: HEWPACKSA Geneva
Telex: 2 24 86

AFRICA, ASIA, AUSTRALIA

ANGOLA
Telectra
Empresa Técnica de Equipamentos
Eléctricos, S.A.R.L
R. Barbosa Rodrigues, 42-1°DT.°
Caixa Postal, 6487-**Luanda**
Tel: 355156
Cable: TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia Pty. Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
Tel: 89-6351, 89-6306
Telex: 31-024
Cable: HEWPARD Melbourne

Hewlett-Packard Australia Pty. Ltd.
31 Bridge Street
Pymble,
New South Wales, 2073
Tel: 449-6566
Telex: 21561
Cable: HEWPARD Sydney

Hewlett-Packard Australia Pty. Ltd.
97 Churchill Road
Prospect 5082
South Australia
Tel: 44 8151
Cable: HEWPARD Adelaide

Hewlett-Packard Australia Pty. Ltd.
141 Stirling Highway
Nedlands, W.A., 6009
Tel: 86 5455

Hewlett-Packard Australia Pty. Ltd.
121 Wollongong Street
Fyshwick, A.C.T., 2609
Tel: 95 3733

Hewlett-Packard Australia Pty. Ltd.
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel: 29-1544
Telex: AA-42133

CEYLON
United Electricals Ltd.
P.O. Box 681
60, Park St.
Colombo 2
Tel: 26696
Cable: HOTPOINT Colombo

CYPRUS
Kypronics
19 Gregorios & Xenopoulos Rd.
P.O. Box 1152
CY-**Nicosia**
Tel: 45628/29
Cable: KYPRONICS PANDEHIS

ETHIOPIA
African Salespower & Agency Private Ltd., Co.
P.O. Box 718
58/59 Cunningham St.
Addis Ababa
Tel: 12285
Cable: ASACO Addisababa

HONG KONG
Schmidt & Co.(Hong Kong) Ltd.
P.O. Box 297
Connalight Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel: 240168, 232735
Telex: HX4766
Cable: SCHMIDTCO Hong Kong

INDIA
Blue Star Ltd.
Kasturi Buildings
Jamshedji Tata Rd.
Bombay 400 020
Tel: 29 50 21
Telex: 3751
Cable: BLUEFROST

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel: 45 78 87
Telex: 4093
Cable: FROSTBLUE

Blue Star Ltd.
Band Box House
Prabhadevi
Bombay 400 025
Tel: 45 73 01
Telex: 3751
Cable: BLUESTAR

Blue Star Ltd.
14/40 Civil Lines
Kampur 208 001
Tel: 6 88 82
Cable: BLUESTAR

Blue Star Ltd.
7 Hare Street
P.O. Box 506
Calcutta 700 001
Tel: 23-0131
Telex: 655
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House.
34 Ring Road
Lajpat Nagar
New Delhi 110 024
Tel: 62 32 76
Telex: 2463
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel: 55668
Telex: 430
Cable: BLUESTAR

Blue Star Ltd.
Meeakshi Mandiran
xxx/1678 Mahatma Gandhi Rd.
Cochin 682 016 Kerala

Blue Star Ltd.
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel: 7 63 91, 7 73 93
Cable: BLUEFROST
Telex: 459

Blue Star Ltd.
23/24 Second Line Beach
Madras 600 001
Tel: 23954
Telex: 379
Cable: BLUESTAR

Blue Star Ltd.
Nathraj Mansions
2nd Floor Bistupur
Jamshedpur 831 001
Tel: 38 04
Cable: BLUESTAR
Telex: 240

INDONESIA
BERCA Indonesia P.T.
P.O. Box 496
1st Floor JL. Cikini Raya 61
Jakarta
Tel: 56038, 40369, 49886
Telex: 2895 Jakarta

IRAN
Multi Corp International Ltd.
Avenue Soraya 130
P.O. Box 1212
IR-**Teheran**
Tel: 83 10 35-39
Cable: MULTICORP Tehran
Telex: 2893 mci tn

ISRAEL
Electronics & Engineering Div. of Motorola Israel Ltd.
17 Aminadav Street
Tel-Aviv
Tel: 36941 (3 lines)
Cable: BASTEL Tel-Aviv
Telex: 33569

JAPAN
Yokogawa-Hewlett-Packard Ltd.
Ohashi Building
1-59-1 Yoyogi
Shibuya-ku, **Tokyo**
Tel: 03-370-2281/92
Telex: 232-2024YHP
Cable: YHPMARKET TOK 23-724

Yokogawa-Hewlett-Packard Ltd.
Nisei Ibaragi Bldg.
2-2-8 Kasuga
Ibaragi-Shi
Osaka
Tel: (0726) 23-1641
Telex: 5332-385 YHP OSAKA

Yokogawa-Hewlett-Packard Ltd.
Nakamo Building
No. 24 Kamisasazima-cho
Nakamura-ku, **Nagoya** City
Tel: (052) 571-5171

Yokogawa-Hewlett-Packard Ltd.
Nitto Bldg.
2-4-2 Shinohara-Kita
Kohoku-ku
Yokohama 222
Tel: 045-432-1504
Telex: 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg.
Rm. 603 3,
2-Chome
IZUMI-CHO
Mito, 310
Tel: 0292-25-7470

KENYA
Technical Engineering Services
P.O. Box 18311
Nairobi, Kenya
Tel: 57726
Cable: PROTON

KOREA
American Trading Company Korea
I.P.O. Box 1103
Dae Kyung Bldg., 8th Floor
107 Sejong-Ro,
Chongro-Ku, **Seoul**
Tel: (4 lines) 73-8924-7
Cable: AMTRACO Seoul

KUWAIT
Al-Khaldiya Trading & Contracting Co.
Al Soor Street
Michaan Bldg. No. 4
Kuwait
Tel: 42 99 10
Cable: VISCOUNT

LEBANON
Constantin E. Macridis
Clemenceau Street 34
P.O. Box 7213
RL-**Beirut**
Tel: 220846
Telex: 21114 Leb
Cable: ELECTRONUCLEAR Beirut

MALAYSIA
MECOMB Malaysia Ltd.
2 Lorong 13/6A
Section 13
Petaling Jaya, **Selangor**
Cable: MECOMB Kuala Lumpur

MOZAMBIQUE
A.N. Goncalves, Lta.
162, 1° Apt. 14 Av. D. Luis
Caixa Postal 107
Lourenco Marques
Tel: 27091, 27114
Telex: 6-203 Negon Mo
Cable: NEGON

NEW ZEALAND
Hewlett-Packard (N.Z.) Ltd.
94-96 Dixon Street
P.O. Box 9443
Courtenay Place,
Wellington
Tel: 59-559
Telex: 3898
Cable: HEWPACK Wellington

Hewlett-Packard (N.Z.) Ltd.
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel: 569-651
Cable: HEWPACK, Auckland

NIGERIA
The Electronics Instrumentatiofns Ltd.
N6B/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel: 22325
Cable: THETEIL Ibadan

The Electronics Instrumentations Ltd. (TEIL)
16th Floor Cocoa House
P.M.B. 5402
Ibadan
Tel: 22325
Cable: THETEIL Ibadan

PAKISTAN
Mushko & Company, Ltd.
Oosman Chambers
Abdullah Haroon Road
Karachi 3
Tel: 511027, 512927
Cable: COOPERATOR Karachi

Mushko & Company, Ltd.
38B, Satellite Town
Rawalpindi
Tel: 41924
Cable: FEMUS Rawalpindi

PHILIPPINES
Electromex, Inc.
6th Floor, Amalgamated Development Corp. Bldg.
Ayala Avenue, Makati, Rizal
C.C.P.O. Box 1028
Makati, Rizal
Tel: 86-18-87, 87-76-77,
Cable: ELEMEX Manila

SINGAPORE
Mechanical & Combustion Engineering Company Pte., Ltd.
10/12, Jalan Kilang
Red Hill Industrial Estate
Singapore, 3
Tel: 647151 (7 lines)
Cable: MECOMB Singapore

Hewlett-Packard Singapore (Pte.) Ltd.
Blk. 2, 6th FLOOR, Jalan Bukit Merah
Redhill Industrial Estate
Alexandra P.O. Box 87,
Singapore 3
Tel: 633022
Telex: HPSG RS 21486
Cable: HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa (Pty.), Ltd.
Hewlett-Packard House
Daphne Street, Wendywood
Sandton, Transvaal 2001
Tel: 802-1040
Telex: SA43-4782JH
Cable: HEWPACK

Hewlett-Packard South Africa (Pty.), Ltd.
Breecastle House
Bree Street
Cape Town
Tel: 2-6941/2/3
Cable: HEWPACK Cape Town
Telex: 0006 CT

Hewlett-Packard South Africa (Pty.), Ltd.
641 Ridge Road, Durban
P.O. Box 99
Overport, Natal
Tel: 88-6102
Telex: 567954
Cable: HEWPACK

TAIWAN
Hewlett-Packard Taiwan
39 Chung Shiao West Road
Sec. 1 Overseas Insurance Corp. Bldg. 7th Floor
Taipei
Tel: 389160,1,2, 375121, Ext: 240-249
Telex: TP824 HEWPACK
Cable: HEWPACK Taipei

Hewlett-Packard Taiwan
38, Po-Ai Lane, San Min Chu,
Kaohsiung
Tel: 297319

THAILAND
UNIMESA Co., Ltd.
Elsom Research Building
Bangjak Sukumvit Ave.
Bangkok
Tel: 932387, 930338
Cable: UNIMESA Bangkok

UGANDA
Uganda Tele-Electric Co., Ltd.
P.O. Box 4449
Kampala
Tel: 57279
Cable: COMCO Kampala

VIETNAM
Peninsular Trading Inc.
P.O. Box H-3
216 Hien-Vuong
Saigon
Tel: 20-805, 93398
Cable: PENTRA, SAIGON 242

ZAMBIA
R.J. Tilbury (Zambia) Ltd.
P.O. Box 2792
Lusaka
Zambia, Central Africa
Tel: 73793
Cable: ARJAYTEE, Lusaka

MEDITERRANEAN AND MIDDLE EAST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard S.A.
Mediterranean and Middle East Operations
35, Kolokotroni Street
Platia Kefallariou
GR-Kifissia-**Athens**
Telex: 21-6588
Cable: HEWPACKSA Athens

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Export Trade Company
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1267
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

READER COMMENT SHEET

24307-90006 Feb 1975

DOS-III Disc Operating System
Reference Manual

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

Is this manual technically accurate?

Is this manual complete?

Is this manual easy to read and use?

Other comments?

FROM:

Name _____

Company _____

Address _____

_____

_____

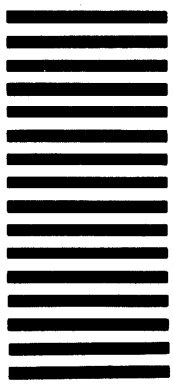
FOLD FOLD

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, Systems Engineering
Hewlett-Packard Company
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014



FOLD FOLD

MANUAL PART NO. 24307-90006
MICROFICHE PART NO. 24307-90007

Printed: FEB 1975
Printed in U.S.A.