UPDATE NOTICE NO. 1

RT-11
System User's Guide

AD-5279B-T1

March 1981

NEW AND CHANGED INFORMATION

This update contains changes and additions to the *RT-11 System User's Guide* (AA-5279B-TC).

Additional copies of this update to the *RT-11 System User's Guide* may be ordered from the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754. Order Number: AD-5279B-T1. The order number of the base manual is AA-5279B-TC.

INSTRUCTIONS

The enclosed pages are to be placed in the *RT-11 System User's Guide* as replacements for, or additions to, current pages. The changes made on replacement pages are indicated in the outside margin by change bars (▌) for additions and bullets (•) for deletions.

KEEP THIS NOTICE IN YOUR MANUAL TO MAINTAIN AN UP-TO-DATE RECORD OF CHANGED PAGES

Old Page	New Page
Title page/Copyright	Title page/Copyright
3-3/3-4	3-3/3-4
4-1 through 4-176	4-1 through 4-176
5-3/5-4	5-3/5-4
7-5/7-6	7-5/7-6
7-9/7-10	7-9/7-10
8-5/8-6	8-5/8-6
8-9 through 8-12	8-9 through 8-12
8-17	8-17
11-11/11-12	11-11 through 11-12.1
11-45/11-46	11-45/11-46
12-5/12-6	12-5/12-6
19-1/19-2	19-1 through 19-2.1
24-1 through 24-6	24-1 through 24-6
Reader's Comments	Reader's Comments

March 1981

This document describes how to use the RT-11 operating system. It provides the information required to perform ordinary tasks such as program development, program execution, and file maintenance.

RT-11
System User's Guide

Order No. AA-5279B-TC

SUPERSESSION/UPDATE INFORMATION: This manual supersedes the *RT-11 System User's Guide*, Order No. DEC-11-ORGDA-A-D, DN1. This manual includes Update Notice No. 1 (AD-5279B-T1).

OPERATING SYSTEM AND VERSION: RT-11 V4.0

SOFTWARE VERSION: RT-11 V4.0

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, March 1980
Updated, March 1981

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage-paid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EduSystem	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

M13100

7/81/14

3.3 Physical Device Names

When you request services from the monitor, you must sometimes specify a peripheral device. You can specify devices by means of a standard two-character physical device name. Table 3–1 lists each name and its related device. If you do not specify a unit number for devices with more than one unit, the system assumes unit 0.

Table 3–1: Permanent Device Names

Permanent Name	I/O Device
CR:	CR11/CM11 Card Reader
CTn:	TA11 Cassette (*n* is 0 or 1)
DDn:	TU58 DECtape II (*n* is an integer in the range 0–3)
DK:	The default logical storage device for all files (DK: is initially the same as SY:)
DKn:	The specified unit of the same device type as the system device
DLn:	RL01, RL02 Disk (*n* is an integer in the range 0–3)
DMn:	RK06, RK07 Disk (*n* is an integer in the range 0–7)
DPn:	RP02, RP03 Disk (*n* is an integer in the range 0–7)
DSn:	RJS03/4 Fixed-Head Disks (*n* is an integer in the range 0–7)
DTn:	DECtape (*n* is an integer in the range 0–7)
DXn:	RX01 Diskette (*n* is an integer in the range 0–3)
DYn:	RX02 Diskette (*n* is an integer in the range 0–3)
LP:	Line Printer
LS:	Serial Line Printer (a hard copy output device connected to a DL11 interface)
MMn:	TJU16/TU45 (industry-compatible) Magtape (*n* is an integer in the range 0–7)
MQ:	Message queue pseudo-device for inter-job communication under the FB monitor.
MSn:	TS11 Magtape (*n* is an integer in the range 0–7)
MTn:	TM11/TMA11/TS03/TE16 (industry-compatible) Magtape (*n* is an integer in the range 0–7)
NL:	Null device
PC:	PC11 Combined High-Speed Paper Tape Reader and Punch
PDn:	The mass storage volume for the PDT–130/150 intelligent terminal. Volumes are DECtape II or single-density diskettes (*n* is either 0 or 1).
RF:	RF11 Fixed-Head Disk Drive
RKn:	RK05 Disk Cartridge Drive (*n* is an integer in the range 0–7)
SY:	The default logical system device; the device and unit from which the system is bootstrapped
SYn:	The specified unit of the same device type as SY:
TT:	Console Terminal Keyboard and Printer
XTn:	Down-line load handler (MRRT–11 only)

In addition to using the permanent names shown in Table 3–1, you can assign logical names to devices. A logical name takes precedence over a physical name and thus provides device independence. With this feature, you do not have to rewrite a program that is coded to use a specific device if the device becomes unavailable. You associate logical names with physical devices by using the ASSIGN command, which is described in Section 4.4.

3.4 File Names and File Types

You can reference files symbolically by a name of one to six alphanumeric characters (followed, optionally, by a period and a file type of up to three alphanumeric characters). No spaces or tabs are allowed in the file name or file type. The file type generally indicates the format or contents of a file. It is good practice to conform to the standard file types for RT–11. If you do not specify a file type for an input or output file, most system programs use or assign an appropriate default file type. Table 3–2 lists the standard file types used in RT–11.

Table 3–2: Standard File Types

File Type	Meaning
.BAC	Compiled BASIC program
.BAD	Files with bad (unreadable) blocks; you can assign this file type whenever bad areas occur on a device. The .BAD file type makes the file permanent in that area, preventing other files from using it and consequently becoming unreadable
.BAK	Editor backup file
.BAS	BASIC source file (BASIC input)
.BAT	BATCH command file
.CND	System generation conditional file
.COM	Indirect command file
.CTL	BATCH control file generated by the BATCH compiler
.CTT	BATCH internal temporary file
.DAT	BASIC or FORTRAN data file
.DBL	DIBOL source file
.DDF	DIBOL data file
.DIF	SRCCOM output file
.DIR	Directory listing file
.DMP	DUMP output file
.FOR	FORTRAN IV source file (FORTRAN input)
.LDA	Absolute binary (load image) file (optional linker output)
.LOG	BATCH log file
.LST	Listing file (MACRO, FORTRAN, LIBR, or DIBOL output)
.MAC	MACRO source file (MACRO or SRCCOM input, LIBR input and output)
.MAP	Map file (linker output)

(continued on next page)

Chapter 4
Keyboard Commands

Keyboard commands allow you to communicate with the RT–11 system. You enter keyboard commands at the terminal in response to the keyboard monitor dot (.), and the operating system invokes the appropriate system programs to service these commands.

This chapter uses some symbolic conventions to describe the monitor command language. The preface to this manual contains a more detailed list of the symbolic conventions used throughout the manual. You should familiarize yourself with the symbols and their meaning before reading this chapter.

4.1 Command Syntax

The system accepts commands as either: (1) a complete string containing all the information necessary to execute a command, or (2) as a partial string. In the latter case the system prompts you to supply the rest of the information. Terminate each command with a carriage return.

The general syntax for a command is:

```
command[/option...] input-filespec[/option...]
output-filespec[/option...]
```

or

```
command[/option...]
prompt1? input-filespec[/option...]
prompt2? output-filespec[/option...]
```

where:

command	is the command name
/option	represents a command qualifier that specifies the exact action to be taken. Any option you supply immediately following the command applies to the entire command string
prompt	represents the keyboard monitor prompt for more information. The keyboard monitor prints an appropriate prompt only if you omit input and/or output file or device specifications in the initial command line (Note that not all keyboard monitor commands print prompts.)
input-filespec	represents the file on which the action is to be taken

/option	represents a file qualifier that specifies more detailed information about that particular file or action to be taken
output-filespec	represents the file that is to receive the results of the operation
/option	represents a file qualifier that specifies more detailed information about that particular file or action to be taken

In the alphabetical listing of keyboard monitor commands in Section 4.4, each command begins with a graphic presentation of the syntax involved (see Figure 4–1 for an illustration of a typical command). These presentations provide a ready-reference list of the options that the commands accept, as well as information that makes the commands easier to use. The following list describes the conventions used.

1. Capital letters represent command names or options, which you must type as shown. (Abbreviations are discussed later in this section.)

2. Lower-case letters represent arguments or variables, for which you must supply values. For options that accept numeric arguments, the system interprets the values as decimal, unless otherwise stated. Some values, usually memory addresses, are interpreted as octal; these cases are noted in the accompanying text.

3. Square brackets ([]) enclose options; you can include the item that is enclosed in the brackets or you can omit it, as you choose. If a vertical list of items is enclosed in square brackets, you can combine the options that appear in the list. However, an option set off from the others by blank lines (see /BOOT and /DEVICE in Figure 4–1) indicates that you cannot combine that option with any other option in the list.

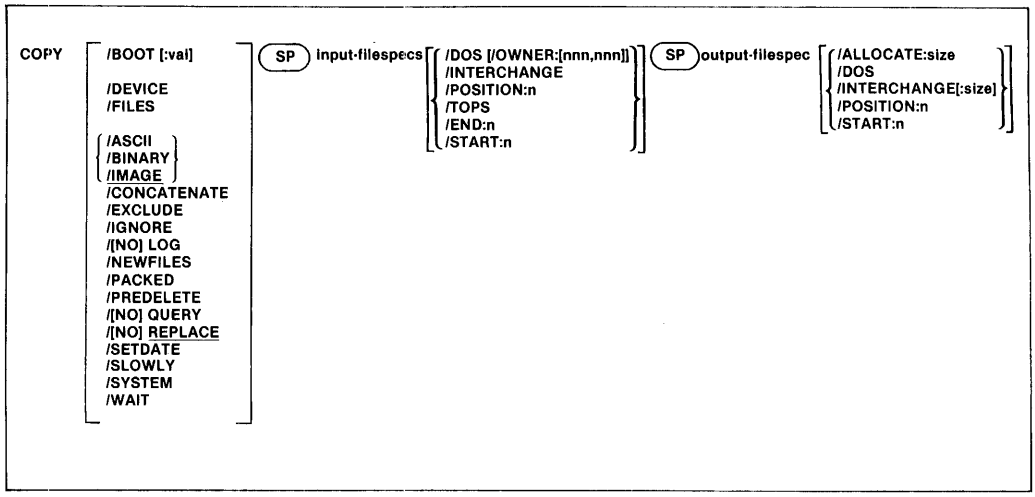
4. Braces ({ }) enclose options that are mutually exclusive. You can choose only one option from a group of options that appear in braces.

5. It is conventional to place command options (those qualifiers that apply to the entire command line) immediately after the command. However, it is also acceptable to specify a command option after a file specification. File options (those that qualify a particular file specification) must appear in the command line directly after the file to which they apply. The graphic representation of each command shows which options are file qualifiers, and whether they must follow input or output file specifications.

6. A line such as [NO]QUERY represents two mutually exclusive options: QUERY and NOQUERY.

7. Underlining indicates default options, that is, the option that the system uses if you do not specify any choice of action.

Figure 4–1: Sample Command Syntax Illustration



A filespec represents a specific file and the device on which it is stored. Its syntax is:

dev:filnam.typ

where:

dev: represents either a logical device name or a physical device name, which is a two- or three-character name from Table 3–1

filnam represents the one- to six-character alphanumeric name of the file

.typ represents the one- to three-character alphanumeric file type, some of which are listed in Table 3–2

There are several ways to indicate the device on which a file is stored. You can explicitly type the device name in the file specification:

```
DX1:TEST.LST
```

You can omit the device name:

```
TEST.LST
```

In this case, the system assumes that the file is stored on device DK:.

4.1.1 Factoring File Specifications

If you want to specify several files on the same device, you can use factoring. That is, you can enclose multiple file names in parentheses, as in the following example:

```
DTO:(TEST.LST,TESTA.LST,TESTB.LST)
```

The command shown above has the same meaning as and is easier to use than the next command:

```
DTO:TEST.LST,DTO:TESTA.LST,DTO:TESTB.LST
```

When you use factoring the device name outside the parentheses applies to each file specification inside the parentheses. Without factoring, the system interprets each file specification to be *DK:filespec* unless you explicitly specify another device name.

Factoring is useful for complicated command lines. It is a general method of string replacement that you can use in many different situations. The monitor uses the following algorithm to interpret command lines that require factoring:

Format of the command line you type:

D1 T1 (T3 D3 T4 D4...Tn) T2 D2

Format of the command line after the monitor performs the factoring:

D1 T1T3T2 D3 T1T4T2 D4...T1TnT2 D2

In the skeleton examples shown above, the symbols have the following meaning:

D represents a delimiter

D1 is one of the following delimiters:

comma
space
beginning of line

D2 is one of the following delimiters:

comma
space
slash
end of line

D3 through Dn can be one of the following delimiters:

comma
space

T represents a text string

The following example shows how a command line expands after factoring. Note that the /SYSTEM option appears only once in the resulting output line.

Original command line:

```
COPY DX:FIL(1,2,3).SYS/SYSTEM RK1:
```

Resulting command line (after factoring):

```
COPY DX:FIL1.SYS,DX:FIL2.SYS,DX:FIL3.SYS/SYSTEM RK1:
```

NOTE

There is a restriction on the use of factoring. The command string that results from the expansion of the line you enter must not exceed 80 characters in length. If you use six-character file names and you also use factoring, specify only five files in a command line.

4.1.2 File Type Specification

If you omit the file type in a file specification, the system assumes one of a number of defaults, depending on which command you issue. The MACRO command, for example, assumes a file type of .MAC for the input file specification, and the PRINT command assumes .LST. Some commands (such as COPY) do not assume a particular file type, and may assume a wildcard default (see Section 4.2). If you need to specify a file that has no file type in a command that assumes a default file type, type a period after the file name. For example, to run the file called TEST, type:

```
RUN TEST.
```

If you omit the period after the file name, the system assumes a .SAV file type and tries to execute a file called TEST.SAV.

You can enter up to six input files and up to three output files for some commands. If the command string does not fit on one line of your terminal, use the hyphen (-) continuation character as the last character in the line to break the string into smaller sections. Use a carriage return to terminate the command string. Note that there is still a limit of 80 characters for an expanded command line.

Some of the command and file options are mutually exclusive. You should avoid using combinations of options that give contradictory instructions to the system. For example,

```
.DELETE/QUERY/NOQUERY TEST.LST
```

This command is not meaningful. Some mutually exclusive options are less obvious; these are noted, where necessary, in the list of options following each command and are enclosed by braces ({ }) in the graphic representations of the command syntax.

4.1.3 Abbreviating Keyboard Commands

Although the keyboard monitor commands are all English-language words and therefore easy to use, it can become tedious to type words like CROSS-REFERENCE and ALLOCATE frequently. You can use as abbreviations the minimum number of characters that are needed to make the command

or option unique. Table B–1 in Appendix B lists the minimum abbreviations for the commands and options. Note also that in Section 4.4, the abbreviations are in red.

An easy way to abbreviate the commands, and one that is always correct, is to use the first four characters or the first six characters if the qualifier starts with NO. For example:

CONCATENATE can be shortened to CONC

NOCONCATENATE can be shortened to NOCONC

The system prints an error message if you use an abbreviation that is not unique. For example, typing the following command produces an error, because C could mean COPY or COMPILE.

```
C TEST.LST
```

4.1.4 Keyboard Prompts

The prompting form of the command may be easier for you to learn if you are a new user. If you type a command followed by a carriage return, the system prompts you for an input file specification:

```
COPY/CONCATENATE
From?
```

You should enter the input file specification and a carriage return:

```
DX1:(TEST.LST,TESTA.LST)
```

The system prompts you for an output file specification:

```
To ?
```

You should enter the output file specification and a carriage return:

```
DX2:TEST.LST
```

The command now executes.

The system continues to prompt for an input and output file specification until you provide them. If you respond to a prompt by entering only a carriage return, the prompt prints again. You can combine the normal form of a command with the prompting form, as this example shows:

```
.COPY    ABC.LST
To  ?    DEF.LST
```

The system always prompts you for information if any required part of the command is missing. You can also enter just an option in response to a prompt. The two following examples are equivalent.

```
.COPY
From ?    *.MAC/NOLOG
To   ?    *.BAK
```

```
.COPY
From ?    /NOLOG
From ?    *.MAC
To   ?    *.BAK
```

4.2 Wildcards

Some commands accept wildcards (% and *) in place of the file name, file type, or characters in the file name or file type. The system ignores the contents of the wild field and selects all the files that match the remaining fields.

An asterisk (*) can replace a file name:

```
*.MAC
```

The system selects all files on device DK: that have a .MAC file type, regardless of their name.

An asterisk (*) can replace a file type:

```
TEST.*
```

The system selects all files on device DK: that are named TEST, regardless of their file type.

An asterisk (*) can replace both a file name and a file type:

```
*.*
```

The system selects all files on device DK:.

An embedded asterisk (*) can replace any number of characters in the input file name or file type:

```
A*B.MAC
```

The system selects all files on device DK: with a file type of .MAC whose file names start with A and end with B. For example, AB, AXB, AXYB, etc., would be selected.

The percentage symbol (%) is always considered to be an embedded wildcard. It can replace a single character in the input file name or file type:

```
A%B.MAC
```

The system selects all files on device DK: with a file type of .MAC whose file names are three characters long, start with A, and end with B. For example, AXB, AYB, AZB, etc., would be selected.

Table 4–1 lists commands that support wildcards.

Table 4–1: Commands Supporting Wildcards

Command	Specification	
	Input File	Output File
COPY	X	X
DELETE	X	
DIRECTORY	X	
HELP	X	
PRINT	X	
RENAME	X	X
TYPE	X	

For commands that support wildcards the system has a special way of interpreting the file specifications you type. You can omit certain parts of the input and output specifications, and the system assumes an asterisk (*) for the omitted item. Table 4–2 shows the defaults that the system assumes for the input and output specifications of the valid commands.

Table 4–2: Wildcard Defaults

Command	Default	
	Input	Output
COPY, RENAME	*.*	*.*
DIRECTORY	DK:*.*	
PRINT, TYPE	*.LST	
DELETE	filnam.*	

For example, if you need to copy all the files called MYPROG from DK: to DX1:, use this command:

```
.COPY/NOQUERY MYPROG  DX1:
```

The system interprets this command to mean:

```
.COPY/NOQUERY  DK:MYPROG.*  DX1:*.*
```

The system copies all the files called MYPROG, regardless of their file type, to device DX1: and gives them the same names.

If you need a directory listing of all the files on device DK:, type the following command:

```
.DIRECTORY
```

The system interprets this command to mean:

```
.DIRECTORY  DK:*.*
```

To list on the printer all the files on device DK: that have a .LST file type, use this command:

```
.PRINT .DK:
```

The system interprets this command to mean:

```
.PRINT  DK:*.LST
```

To delete all the files on device DK: called MYPROG, regardless of their file type, use this command:

```
.DELETE/NOQUERY  MYPROG
```

The system interprets this to mean:

```
.DELETE/NOQUERY  DK:MYPROG.*
```

You can use the SET WILDCARDS EXPLICIT command (described in Section 4.4) to change the way the system interprets these commands.

4.3 Indirect Files

You can group together as a file a collection of keyboard commands that you want to execute sequentially. This collection is called an indirect command file, or indirect file. Indirect files are best suited to perform tasks that require a significant amount of computer time and that do not require your supervision or intervention. Any series of commands that you are likely to type often can also run easily as an indirect file. The indirect file concept is similar to BATCH processing. Although indirect files lack some of the capabilities of BATCH, they are easier to use, use the same commands as normal operations, and generally require less memory overhead than the BATCH processor. (RT–11 BATCH is described in Appendix A of this manual.) This section describes how to create indirect files and how to execute them.

4.3.1 Creating Indirect Files

Create an indirect file by using the EDIT/CREATE command described in Section 4.4. It is conventional to use a .COM file type for an indirect file, but you can choose any file name that you wish. Structure the lines of text to look like keyboard input, placing one command on each line of the file and terminating each line with a carriage return. Do not include the prompt character (.) in the line. Any keyboard monitor command you can type at the terminal you can also include in an indirect file. The following file, for example, prints the date and time, and creates backup copies of all FORTRAN source files:

```
DATE
TIME
COPY *.FOR *.BAK
```

Control returns to the monitor at the console terminal after this indirect file executes.

In addition to using the keyboard monitor commands, you can also run one of the RT–11 system utility programs in an indirect file. In this case, structure your input to conform to the Command String Interpreter syntax described in Chapter 6. Do not include the CSI asterisk (*) in any line that provides input or output to a utility program. The following file starts the directory system utility program and lists the directory of two devices on the line printer.

```
R DIR
LP:=CT0:/C:3
LP:=DT1:/C:3
^C
```

Note that the last command line is ^C. This is not the standard CTRL/C sequence you enter by holding down the CTRL key and typing a C. Rather, it is a readable character sequence that consists of two separate characters: a circumflex (uparrow) followed by a C. This sequence represents CTRL/C in indirect files. This sequence terminates the directory program so that control returns to the monitor when the indirect file finished executing. Otherwise, the directory program would be left waiting for input from the console terminal when the indirect file finished executing.

Remember to terminate the last command line with a carriage return, as you would any other line.

NOTE

If you have a small (12K) configuration or a very large indirect command file, use frequent CTRL/Cs in your indirect files. When the system processes an indirect file, it first places each line in a special memory buffer. This memory buffer must expand to accommodate each line in an indirect file, and if there are too many lines before the system reaches a CTRL/C, the processor's memory area may become filled. Placing a CTRL/C every ten or so lines avoids this problem.

Some commands normally require a response from you as they execute. The INITIALIZE command, for example, prints the *Are you sure?* message and waits for you to type Y and a carriage return before it executes. The DELETE command also requests confirmation from you before it deletes a file. There are three ways to control interaction with the executing command. One way is to use the /NOQUERY option on each command that allows it. This option suppresses the confirmation messages entirely when you use the command in an indirect file.

Another method of interacting applies to a command like DELETE. This command can have a variable number of confirmation queries, especially if you use a wildcard in the file specification. This type of command accepts your responses directly from the terminal and allows you to make a decision before deleting each file. However, in this case the indirect file cannot operate unattended.

There is yet another way to deal with commands that require a response from you. Both the INITIALIZE and LINK commands have options that cause the system to prompt you for data. This section describes two methods of responding to these prompts, where more than just a Y response is required. The INITIALIZE command with the /VOLUMEID option permits you to specify a volume ID and owner name for a device. You can place your responses in the indirect file, as this example shows:

```
INITIALIZE/NOQUERY/VOLUMEID DT:
TAPE6
PAYROLL
```

You can change the indirect file so that the prompts appear on the console terminal and you can type your responses there:

```
INITIALIZE/NOQUERY/VOLUMEID DT:
^C
```

The ^C informs the system that the responses are to be entered at the terminal. Execution of the indirect file pauses until you enter the responses.

Similarly, the LINK command lets you specify some data either in the indirect file or from the console terminal. The following example contains the response to the TRANSFER prompt.

```
LINK/TRANSFER MYPROG,ODT
0,ODT
```

You can specify the same information interactively, as this example shows:

```
LINK/TRANSFER MYPROG,ODT
^C
```

The ^C informs the system that the response to the prompt is to be entered at the terminal. Execution of the indirect file pauses until you enter your response.

You can specify overlays to the LINK command by either of these two methods. The following indirect file links an overlaid program consisting of a root module and four overlay modules that reside in two overlay segments.

```
LINK/PROMPT ROOT
OVR1/O:1
OVR2/O:1
OVR3/O:2
OVR4/O:2//
```

Note in the above example that two slashes (//) terminate the module list. You can also enter all or part of the overlay information interactively, as this example shows:

```
LINK/PROMPT ROOT
OVR1/O:1
^C
```

The ^C informs the system that more overlay information is to be entered from the terminal. Execution of the indirect file pauses when the system requires the information. Respond to the asterisk prompt by entering the overlay information. Terminate the last overlay line with two slashes (//). Execution of the indirect file then proceeds. Chapter 11 describes the LINK program and explains how to use overlays.

If you need to link more than six modules, you can specify the extra modules on the next line in the indirect file, as this example shows:

```
LINK/PROMPT FIL1,FIL2,FIL3,FIL4,FIL5,FIL6
FIL7,FIL8//
```

Or, you can enter the extra modules from the terminal:

```
LINK/PROMPT FIL1,FIL2,FIL3,FIL4,FIL5,FIL6
^C
```

Execution of the indirect file pauses until you enter the remaining module names. Remember to follow the last name with two slashes (//).

You can include comments in an indirect file to help you document your work. These comments do not print on the console terminal when the indirect file executes. You begin each line of comment with an exclamation point (!). The system ignores any characters it finds between the exclamation point and the end of the current line. The following example shows an indirect file that contains comments.

```
!INDIRECT FILE
DATE                    !PRINT DATE
TIME                    !PRINT TIME
RENAME *.MAC *.BAK      !SAVE .MAC FILES
@PROCES                 !CALL ANOTHER INDIRECT FILE
DIRECTORY               !LIST DIRECTORY OF DK:
```

NOTE

You cannot place in indirect files responses to prompts that result in destruction of data. For example, you cannot use the INITIALIZE command followed by a Y on the following line in an indirect file. Commands like INITIALIZE and DELETE require responses that you must enter at the terminal. (You can avoid the need for a response by using the /NOQUERY option.)

4.3.2 Executing Indirect Files

You can execute indirect files under the SJ monitor, or in the background area under the FB or XM monitor.

To execute an indirect file, specify a command string according to the following syntax:

@filespec

where:

@	is the monitor command that indicates an indirect file
filespec	represents the name and file type of the indirect file, as well as the device on which it is stored. The default file type is .COM

If you omit the device specification, DK: is assumed. If you specify any other block-replaceable device, the monitor automatically loads the handler for that device. It is conventional to type the indirect file command directly in response to the monitor's prompt, as this example shows:

```
.@INDCT
```

However, you can place the indirect command anywhere in a keyboard monitor command string, as long as it is the last element in the string, not including comments. For example:

```
.DELETE/NOQUERY @INDCT!COMMENTS
```

This is a valid command string. The first line of the file should contain the list of files to be deleted. In the example above, assume the first line of the indirect file is:

```
*.BAK
```

This is the command that will actually execute:

```
DELETE/NOQUERY *.BAK
```

Check your indirect file carefully for errors before you execute it. When the monitor or any program that has control of the system encounters an illegal command line, or if an execution error of any kind occurs, that particular line does not execute properly. Execution of the indirect file does proceed, however, until any program that may be running relinquishes control to the monitor. Be careful of this if you run a system utility program in an indirect file, as this example shows:

```
R PIP
DX1:*.*=DX0:*.*
DX0:*.MAC/D
^C
PRINT DX0:*.LST
```

If device DX1: becomes full before all the files from DX0: are copied to it, the second line of the indirect file does not execute completely. Execution then passes to the next line and the system deletes all MACRO files from DX0:. The ^C returns control to the monitor, which aborts the rest of the indirect file. This example shows that it is possible to destroy files accidentally because of the way indirect files execute. To be safe, use only keyboard mon-

itor commands in an indirect file. This way the monitor regains control after each operation and can abort the indirect file as soon as it detects an error. A better way to perform the same operations as the indirect file shown above is as follows:

```
COPY DX0:*.* DX1:*.*
DELETE DX0:*.MAC
PRINT DX0:*.LST
```

You can use the SET ERROR command, described in Section 4.4, to define the severity of error that causes an indirect file to stop executing.

Normally, as each line of an indirect file executes, it echoes on the console terminal so that you can observe the progress of the job. However, you can use the SET TT QUIET command, described in Section 4.4, to suppress this printout. In this case, only the prompting messages, if any, print. You can stop execution of an indirect file at any time by typing two CTRL/C characters. Control returns to the monitor and you can enter a new command. You can also abort the indirect file by typing a single CTRL/C in response to a query or prompt. If you use an indirect file to execute a MACRO program, read the appropriate section in the *RT–11 Programmer's Reference Manual* to learn about certain restrictions on using the .EXIT call with indirect files.

You can call another indirect file from within an indirect file. This procedure is called nesting. Restrict nesting to three levels of indirect files (see the *RT–11 Installation and System Generation Guide* for details on selecting the indirect file nesting depth). The following example shows two-level nesting. Assume a programmer types this command at the console terminal in response to the monitor's prompt:

```
@FIRST
```

The file FIRST.COM contains these lines:

```
DATE
TIME
COPY *.MAC *.BAK
@SECOND
PRINT C
DIRECTORY/PRINTER DK:
DELETE/NOQUERY *.MAC
```

When this file executes it calls another indirect file, SECOND.COM, which contains this line:

```
MACRO/CROSSREFERENCE A+B+C/LIST
```

When file SECOND.COM finishes executing, control returns to file FIRST.COM, at the line following the indirect file specification. FIRST.COM then prints the contents of the file C.LST on the line printer, followed by a directory listing of device DK:. Then control returns to the monitor at the console terminal.

4.3.3 Startup Indirect Files

Section 3.1 introduced the startup indirect command files: STARTS.COM (for SJ), STARTF.COM (for FB), and STARTX.COM (for XM). Each monitor automatically invokes its own indirect command file when you bootstrap the system, and you can modify these files to perform standard system configurations. Since many of the system parameters are reset by a bootstrap operation (see the SET command, Section 4.4), you should use the startup indirect files to set the system parameters you normally use. For example, if you use the FB monitor and have a visual display console terminal that supports hardware tabs, add the SET TT: SCOPE and SET TT: TAB commands to the file STARTF.COM. You could also include a SET TT: QUIET command at the beginning of STARTF.COM and a SET TT: NOQUIET command at the end to suppress extra type-out at bootstrap time. If you have a list of commands that you need to execute, regardless of the monitor you bootstrap, include these commands in a separate indirect file, such as COMMON.COM, and invoke this file from all three startup indirect files. The following example shows a typical STARTF.COM file.

```
SET   TT:   QUIET                    !TURN OFF TTY PRINTING
SET   TT:   SCOPE
SET   TT:   TAB
@COMMON                              !PERFORM COMMON OPERATIONS
SET   TT:   NOQUIET                  !TURN ON TTY PRINTING
```

If you use BATCH frequently, use a startup indirect file to assign devices and load handlers. You can also use the startup indirect files to run your own programs, set the date, or do other housekeeping chores.

4.4 Keyboard Monitor Commands

The keyboard monitor commands are your means of communicating with the system and controlling the monitor. This section lists the keyboard monitor commands in alphabetical order. Each command description includes the command syntax, a table of valid options, and some sample command lines, as well as a general discussion of how to use the command.

You can type almost all the commands to any of the three monitors. The exceptions are FRUN, SRUN, SUSPEND, and RESUME. These are not valid for the SJ monitor because they apply to foreground programs.

Any reference to the background program applies also to the program running under the SJ monitor. Any reference to FB operation also applies to the XM operation.

NOTE

Unless noted otherwise, all numeric values you supply to keyboard commands should be in decimal.

If you make a mistake in a command line, or if the system cannot perform the action you request, an error message prints on your terminal. The error message indicates which error occurred; see the *RT–11 System Message Manual* for a more complete description of the error and for the recommended action to take. The error message also indicates which system utility program detected the error. For example, if your keyboard monitor command line contains a syntax error, the keyboard monitor prints an error message. If the utility program the keyboard monitor invokes cannot execute a command, that utility prints the error message.

RT–11 permits you to remove some of the monitor commands at system generation time. If you type a command that is not part of your system, the system prints an error message.

APL

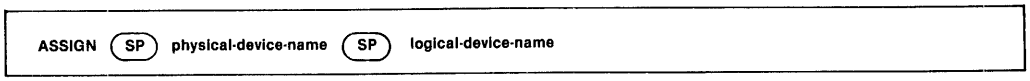
The APL command invokes the APL interpreter.

```
APL
```

Because APL has its own command language, the APL command accepts no options and no file specifications. For information on using the APL interpreter, see the *APL–11 Programmer's Reference Manual.*

ASSIGN

The ASSIGN command associates the logical name you specify with a physical device.



In the command syntax illustrated above, *physical-device-name* represents the RT–11 standard permanent name that refers to a particular device that is installed on your system. Table 3–1 contains a list of these names. The term *logical-device-name* represents an alphanumeric name, from one to three characters long, that you assign to a particular device. Note that you can not use spaces or tabs in the logical device name. If you type ASSIGN, followed by a carriage return, the system prompts: *Physical device name?*. If you follow the physical device name with a carriage return, the system prompts: *Logical device name?*.

If the logical device name you supply is already associated with a physical device, the system disassociates the logical name from that physical device and assigns it to the current device. You can assign only one logical name with each ASSIGN command, but you can use several ASSIGN commands to assign different logical names to the same device. You can also use the ASSIGN command to assign FORTRAN logical units to physical devices (see the *RT–11/RSTS/E FORTRAN IV User's Guide*).

The ASSIGN command simplifies programming. When you write a program, for example, you can request input from a device called INP: and direct output to a device called OUT:. When you are ready to execute the program, you can assign those logical names to the physical devices you need to use for that job. The ASSIGN command is especially helpful when a program refers to a device that is not available on a certain system; the ASSIGN command allows you to direct input and ouput to an available device.

Note that BA and SY are always invalid as logical device names.

The following command, for example, causes data that you write to device LST: to print on the line printer.

```
.ASSIGN LP: LST:
```

If your program attempts to access a device by using a logical name (such as LST:) and you do not issue an appropriate ASSIGN command, an error occurs in the program.

The following command redirects printer output to the terminal.

```
.ASSIGN TT: LP:
```

The command shown above illustrates how you can run a program that specifically references LP: without using a line printer.

The next command redefines the default file device.

```
.ASSIGN RK1: DK:
```

If you supply a file specification and omit the device name, it now defaults to RK1:. Note that this does not affect the default system device, SY:.

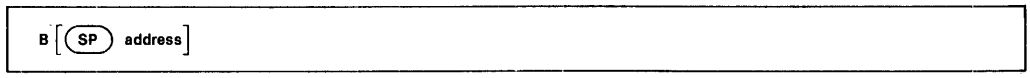
The last example is typical for a system that uses a dual-drive diskette device. Several users can share the same system software on DX0: and maintain their own data files on diskettes that they run in drive 1. When you use the following command, references to files without an explicit device name automatically access DX1:.

```
.ASSIGN DX1: DK:
```

Use the SHOW command to display logical device name assignments on the terminal.

B

The B (Base) command sets a relocation base. To obtain the address of the location to be referenced in a subsequent Examine or Deposit command, the system adds this relocation base to the address you specify.



In the command syntax shown above, *address* represents an octal address that the system uses as a base address for subsequent Examine and Deposit commands. If the address you supply is an odd number, the system decreases it by one to make the address even. Note that if you do not specify an address, this command sets the base to zero.

Use the B command when using the Examine and Deposit commands to reference linked modules that you have loaded into memory with the GET command. (Note that the Base command has no effect on program execution.) The system adds the current base address to the value you supply in an Examine or Deposit command. You can set the current base address to the address where a particular module is loaded. Then you can use the relocatable addresses printed in the assembler, compiler, or map listing of that module to reference locations within the module.

The following command sets the base to 0.

```
.B
```

The next two commands both set the base to 1000.

```
.B 1000
.B 1001
```

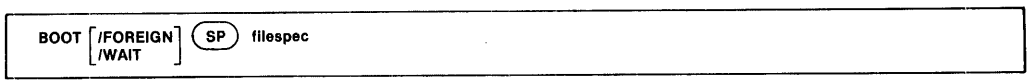
The BASIC command invokes the BASIC language interpreter.

```
BASIC
```

Because BASIC has its own command language, the BASIC command accepts no options and no file specifications. For information on using the BASIC interpreter, see the *BASIC–11 Language Reference Manual.*

BOOT

The BOOT command directs a new monitor to take control of the system. It can also read into memory a new copy of the monitor that is currently controlling the system.



In the command syntax illustrated above, *filespec* represents the device or monitor file to be bootstrapped. If you omit *filespec,* the system prompts you with *Device or file?*. The BOOT command can perform either of two operations: (1) a hardware bootstrap of a specific device, or (2) a direct bootstrap of a particular monitor file that does not use the bootstrap blocks on the device. When you bootstrap a volume, make sure that the appropriate device handler is present on that volume.

To perform a hardware bootstrap, specify only a device name in the command line. The following devices are valid for this operation:

DT0:-DT7:	DX0:-DX1:
RK0:-RK7:	PD0:-PD1:
RF:	DD0:-DD1:
SY:	DL0:-DL3:
DK:	DY0:-DY1:
DP0:-DP7:	DM0:-DM7:
DS0:-DS7:	

You can also boot any of the above storage volumes by specifying its logical name, if assigned (see the ASSIGN command). The hardware bootstrap operation gives control of the system to the monitor whose bootstrap is written on the device. (You can change this monitor by using the COPY/BOOT command.) This example bootstraps the single-job monitor, RT11SJ, whose bootstrap information is written on device DK:.

```
.BOOT DK:

RT-11SJ  V04.00
```

To bootstrap a particular monitor file, specify that file name and the device on which it is stored, if necessary, in the command line. SY: is the default device, and .SYS is the default file type.

You can use the BOOT command to alternate between the single-job and foreground/background monitors. When you use the BOOT command to change monitors you do not have to reenter the date and time. The system clock, however, may lose a few seconds during a reboot. The next example bootstraps the foreground/background monitor on device SY:, which is currently RK0:.

```
.BOOT RT11FB

RT-11FB  V04.00
```

NOTE

If you are running a foreground or system job that is sending I/O to the system volume, using the BOOT command may cause your system to hang. You should terminate such a job in the foreground before using the BOOT command.

/FOREIGN Use this option to boot a pre-version 4 volume or a non-RT-11 system. You may not specify a file name with /FOREIGN. The /FOREIGN option does not preserve the date or time.

/WAIT The /WAIT option is useful if you have a single-disk system. When you use this option, the system initiates the BOOT procedure but then pauses and waits for you to mount the volume you want to bootstrap. When the system pauses, it prints *Mount input volume in <device>; Continue?* at the terminal, where *<device>* represents the device into which you mount the volume. Mount the volume you want to bootstrap, then type Y followed by a carriage return.

The following sample command line boots an RK05 disk:

```
.BOOT/WAIT RK0:
Mount input volume in RK0:; Continue? Y
```

CLOSE

The CLOSE command closes and makes permanent all output files that are currently open in the background job.

```
CLOSE
```

The CLOSE command accepts no options or arguments.

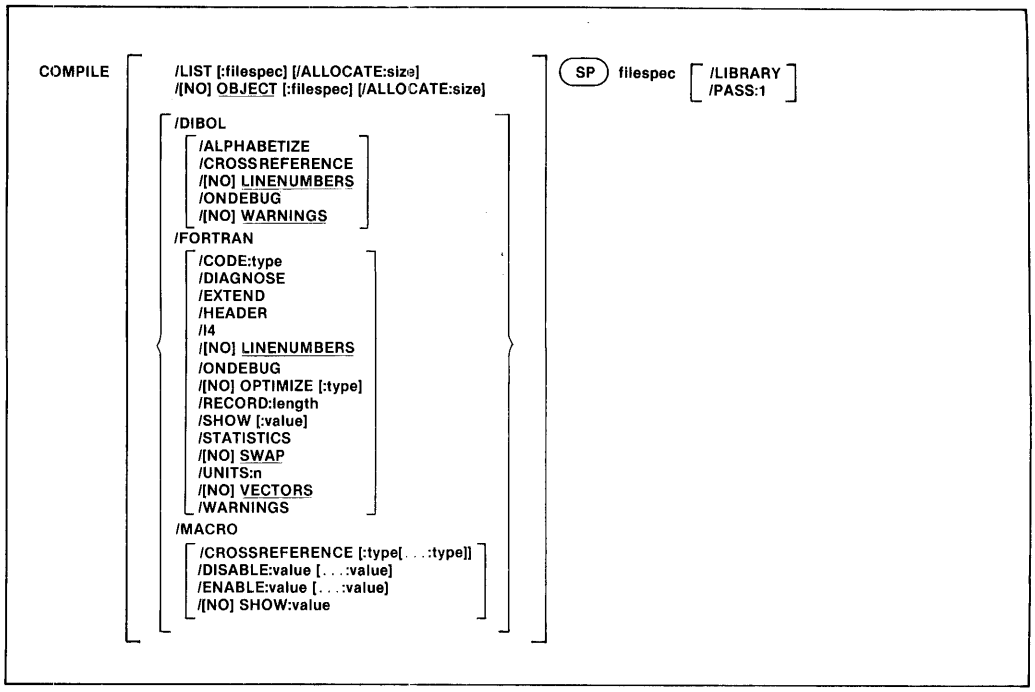
You can use the CLOSE command to make tentative open files permanent; otherwise, they do not appear in a normal directory listing and the space associated with the files is available for reuse. The CLOSE command is particularly useful after you type a CTRL/C to abort a background job. You can also use it after an unexpected program termination to preserve any new files that were being used by the terminated program. Note that the CLOSE command has no effect on a foreground job and that you cannot use CLOSE on files opened on magnetic tape or cassette.

The CLOSE command does not work if your program defines new input or output channels (with the .CDFN programmed request). Because CTRL/C or .EXIT resets channel definitions, the CLOSE command has no effect on channels it does not recognize.

The following example shows how the CLOSE command makes temporary files permanent.

```
.R PROG
    .
    .
    .
^C ^C
.CLOSE
```

The COMPILE command invokes the appropriate language processor to assemble or compile the files you specify.



In the command line shown above, *filespecs* represents one or more files to be included in the assembly or compilation. The default file types for the output files are .LST for listing files and .OBJ for object files. The defaults for input files depend on the particular language processor involved and include .MAC for MACRO files, .FOR for FORTRAN files, and .DBL for DIBOL files.

To compile (or assemble) multiple source files into a single object file, separate the files by plus (+) signs in the command line. Unless you specify otherwise, the system creates an object file with the same name as the first input file and gives it an .OBJ file type. To compile multiple files in independent compilations, separate the files by commas (,) in the command line. This generates a corresponding object file for each set of input files. You can combine up to six files for a compilation producing a single object file.

Language options are position-dependent — that is, they have different meanings depending on where you place them in the command line. Options that qualify a command name apply across the entire command string. Options that follow a file specification apply only to the file (or group of files separated by plus signs) that they follow in the command string.

You can specify the entire COMPILE command as one line, or you can rely on the system to prompt you for information. The COMPILE command prompt is *Files?*.

COMPILE

There are three ways to establish which language processor the COMPILE command invokes.

1. Specify a language-name option, such as /MACRO, which invokes the MACRO assembler.
2. Omit the language-name option and explicitly specify the file type for the source files. The COMPILE command then invokes the language processor that corresponds to that file type. Specifying the file SOURCE.MAC, for example, invokes the MACRO assembler.
3. Let the system choose a file type of .MAC, .DBL, or .FOR for the source file you name. To do this, the handler for the device you specify must be loaded. If you specify DX1:A and the DX handler is loaded, the system searches for source files A.MAC and A.DBL, in that order. If it finds one of these files, the system invokes the corresponding language processor. If it cannot find one of these files, or if the device handler associated with the input file is not resident, the system assumes a file type of .FOR and invokes the FORTRAN compiler.

If the language processor selected as a result of one of the procedures described above is not on the system device (SY:), the system issues an error message.

The following sections explain the options you can use with the COMPILE command.

/ALLOCATE:size Use this option with /LIST or /OBJECT to reserve space on the device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of –1 is a special case that creates the largest file possible on the device.

/ALPHABETIZE Use this option with /DIBOL to alphabetize the entries in the symbol table listing. This is useful for program maintenance and debugging.

/CODE:type Use this option with /FORTRAN to produce object code that is designed for a particular hardware configuration. The argument *type* represents a three-letter abbreviation for the type of code to produce. The legal values are: EAE, EIS, FIS, and THR. See the *RT–11/RSTS/E FORTRAN IV User's Guide* for a complete description of the types of code and their function.

/CROSSREFERENCE[:type[...:type]] Use this option with /MACRO or /DIBOL to generate a symbol cross-reference section in the listing. This information is useful for program maintenance and debugging. Note that the system does not generate a listing by default. You must also specify /LIST in the command line to get a cross-reference listing.

With /MACRO, this option takes an optional argument. The argument *type* represents a one-character code that indicates which sections of the cross-reference listing the assembler should include. See the MACRO command in this chapter for a summary of valid arguments and their meaning.

/DIAGNOSE Use this option with /FORTRAN to help analyze an internal compiler error. /DIAGNOSE expands the crash dump information to include internal compiler tables and buffers. Submit the diagnostic printout to DIGITAL with an SPR form. The information in the listing can help the DIGITAL programmers locate the compiler error and correct it.

/DIBOL This option invokes the DIBOL language processor to compile the associated files.

/DISABLE:value[...:value] Use this option with /MACRO to specify a .DSABL directive. See the MACRO command in this chapter for a summary of the arguments and their meaning. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all legal values.

/ENABLE:value[...:value] Use this option with /MACRO to specify an .ENABL directive. See the MACRO command in this chapter for a summary of the arguments and their meaning. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all legal values.

/EXTEND Use this option with /FORTRAN to change the right margin for source input lines from column 72 to column 80.

/FORTRAN This option invokes the FORTRAN language processor to compile the associated files.

/HEADER Use this option with /FORTRAN to include in the printout a list of options that are currently in effect.

/I4 Use this option with /FORTRAN to allocate two words for the default integer data type (FORTRAN uses only one-word integers) so that it takes the same physical space as real variables.

/LIBRARY Use this option with /MACRO to identify a macro library file; use it only after a library file specification in the command line. The MACRO assembler looks first to any MACRO libraries you specify before going to the default system macro library, SYSMAC.SML, to satisfy references (made with the .MCALL directive) from MACRO programs. In the example below, the two files A.FOR and B.FOR are compiled together, producing B.OBJ and B.LST. The MACRO assembler assembles C.MAC, satisfying .MCALL references from MYLIB.MAC and SYSMAC.SML. It produces C.OBJ and C.LST.

```
.COMPILE A+B/LIST/OBJECT,MYLIB/LIBRARY+C.MAC/LIST/OBJECT
```

/LINENUMBERS Use this option with /DIBOL or /FORTRAN to include internal sequence numbers in the executable program. These numbers are especially useful in debugging programs. This is the default operation.

/NOLINENUMBERS Use this option with /DIBOL or /FORTRAN to suppress the generation of internal sequence numbers in the executable program. This produces a smaller program and optimizes execution speed. Use this option to compile only those programs that are already debugged; other-

wise the line numbers in DIBOL or FORTRAN error messages are difficult to interpret.

/LIST[:filespec] You must specify this option to produce a compilation or assembly listing. The /LIST option has different meanings depending on its position in the command line.

If you specify /LIST without a file specification in the list of options that immediately follows the command name, the system generates a listing that prints on the line printer. If you follow /LIST with a device name, the system creates a listing file on that device. If the device is a file-structured device, the system stores the listing file on that device, assigning it the first input file name and a .LST file type. The following command produces a listing on the terminal:

```
.COMPILE/LIST:TT:  A.FOR
```

The next command creates a listing file called A.LST on RK3:.

```
.COMPILE/LIST:RK3: A.MAC
```

If the /LIST option contains a name and file type to override the default of .LST, the system generates a listing file with that name. The following command, for example, compiles A.FOR and B.FOR together, producing files A.OBJ and FILE1.OUT on device DK:.

```
.COMPILE/FORTRAN/LIST:FILE1.OUT A+B
```

Another way to specify /LIST is to type it after the file specification to which it applies. To produce a listing file with the same name as a particular input file, you can use a command similar to this one:

```
.COMPILE/DIBOL A+B/LIST:RK3:
```

The command shown above compiles A.DBL and B.DBL together, producing files DK:A.OBJ and RK3:B.LST. If you specify a file name on a /LIST option following a file specification in the command line, it has the same meaning as when it follows the command. The following two commands have the same results:

```
.COMPILE/MACRO A/LIST:B

.COMPILE/MACRO/LIST:B A
```

Both the commands shown above generate as output files A.OBJ and B.LST on device DK:.

Remember that file options apply only to the file (or group of files that are separated by plus signs) they follow in the command string. For example:

```
.COMPILE A.MAC/LIST,B.FOR
```

This command compiles A.MAC, producing A.OBJ and A.LST on DK:. It also compiles B.FOR, producing B.OBJ on DK:. However, it does not produce any listing file for the compilation of B.FOR.

/MACRO This option invokes the MACRO assembler to assemble the associated files.

/OBJECT[:filespec] Use this option to specify a file name or device for the object file. Because the COMPILE command creates object files by default, the following two commands have the same meaning:

```
.COMPILE/FORTRAN A

.COMPILE/FORTRAN/OBJECT A
```

Both commands compile A.FOR and produce A.OBJ as output. The /OBJECT option functions like the /LIST option; it can be either a command or a file qualifier.

As a command option, /OBJECT applies across the entire command string. The following command, for example, assembles A.MAC and B.MAC separately, creating object files A.OBJ and B.OBJ on RK1:.

```
.COMPILE/OBJECT:RK1: (A,B).MAC
```

Use /OBJECT as a file option to create an object file with a specific name or destination. The following command compiles A.DBL and B.DBL together, creating files B.LST and B.OBJ.

```
.COMPILE/DIBOL A+B/LIST/OBJECT
```

/NOOBJECT Use this option to suppress creation of an object file. As a command option, /NOOBJECT suppresses all object files; as a file option, it suppresses only the object file produced by the related input files. In this command, for example, the system compiles A.FOR and B.FOR together, producing files A.OBJ and B.LST. It also compiles C.DBL and produces C.LST, but it does not produce C.OBJ.

```
.COMPILE A.FOR+B.FOR/LIST,C.DBL/NOOBJECT/LIST
```

/ONDEBUG Use this option with /DIBOL to include a symbol table in the object file. You can then use a debugging program to find and correct errors in the object file.

Use /ONDEBUG with FORTRAN to include debug lines (those that have a D in column 1) in the compilation. You do not, therefore, have to edit the file to include these lines in the compilation or to logically remove them. This option means that you can include messages, flags, and conditional branches to help you trace program execution and find errors.

/OPTIMIZE[:type] Use this option with /FORTRAN to enable certain options that optimize object code for various conditions. The argument *type* represents the three-letter code for the type of optimization to enable. Table 4–4 summarizes the codes and their meaning. This option is not available in version 2.5 of the FORTRAN compiler.

/NOOPTIMIZE[:type] Use this option with /FORTRAN to disable certain options that optimize object code for various conditions. The argument *type* represents the three-letter code for the type of optimization to disable. Table

4–4 summarizes the codes and their meaning. This option is not available in version 2.5 of the FORTRAN compiler.

/PASS:1 Use this option with /MACRO on a prefix macro file to process that file during pass 1 of the assembly only. Using this option means that you can assemble a source program together with a prefix file that contains only macro definitions, because these definitions do not need to be redefined in pass 2 of the assembly. The following command assembles a prefix file and a source file together, producing files PROG1.OBJ and PROG1.LST.

```
.COMPILE/MACRO PREFIX/PASS:1+PROG1/LIST/OBJECT
```

/RECORD:length Use this option with /FORTRAN to override the default record length of 132 characters for ASCII sequential formatted input and output. The meaningful range for the argument *length* is from 4 to 4095.

/SHOW:value Use this option with /FORTRAN to control FORTRAN listing format. The argument *value* represents a code that indicates which listings the compiler is to produce. Table 4–5 summarizes the codes and their meaning.

Use this option with /MACRO to specify any MACRO .LIST directive. Table 4–12 summarizes the valid arguments and their meaning. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives.

/NOSHOW:value Use this option with MACRO to specify any MACRO .NLIST directive. Table 4–12 summarizes the valid arguments and their meaning. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives.

/STATISTICS Use this option with /FORTRAN to include compilation statistics in the listing, such as amount of memory used, amount of time elapsed, and length of the symbol table.

/SWAP Use this option with /FORTRAN to permit the USR (User Service Routine) to swap over the FORTRAN program in memory. This is the default operation.

/NOSWAP Use this option with /FORTRAN to keep the USR resident during execution of a FORTRAN program. This may be necessary if the FORTRAN program uses some of the RT–11 system subroutine calls (see the *RT–11 Programmer's Reference Manual*). If the program frequently updates or creates a large number of different files, making the USR resident can improve program execution. However, the cost for making the USR resident is 2K words of memory.

/UNITS:n Use this option with FORTRAN to override the default number of logical units (6) to be open at one time. The maximum value you can specify for *n* is 16.

/VECTORS This option directs FORTRAN to use tables to access multidimensional arrays. This is the default mode of operation.

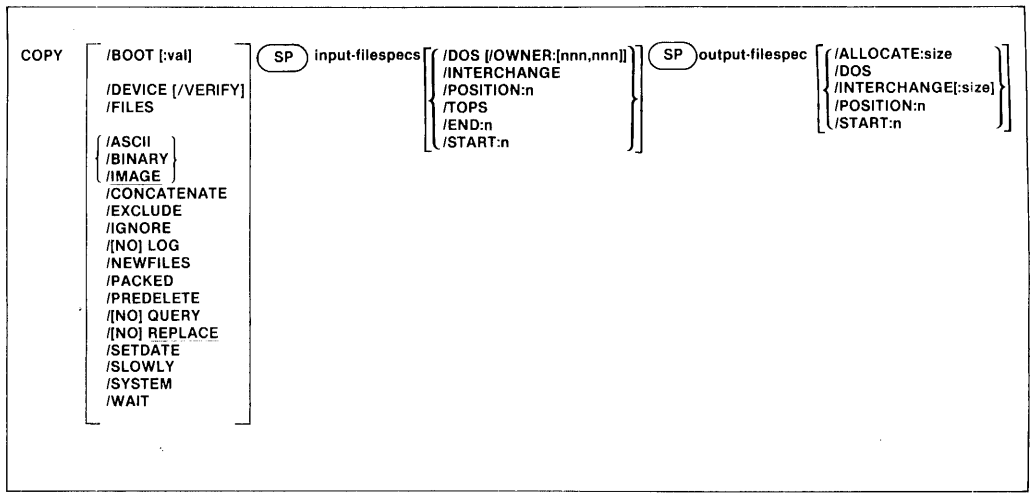
/NOVECTORS This option directs FORTRAN to use multiplication operations to access multidimensional arrays.

/WARNINGS Use this option to include warning messages in DIBOL or FORTRAN compiler diagnostic error messages. These messages call certain conditions to your attention but do not interfere with the compilation. This is the default operation for DIBOL.

/NOWARNINGS Use this option with /DIBOL or /FORTRAN to suppress warning messages during compilation. These messages are for your information only; they do not affect the compilation. This is the default operation for FORTRAN.

COPY

The COPY command performs a variety of file transfer and maintenance operations.



The COPY command transfers:

- one file to another file
- a number of files to a single file by concatenation
- the contents of a device to another device
- the contents of a bootstrap to a device
- the contents of a device to a file and vice versa

In the command syntax shown above, *input-filespecs* represents the data to copy. The *input-filespec* can be a device name, if you use the /DEVICE option. Otherwise, you can specify as many as six files for input. *Output-filespec* represents the device or file to receive the data. You can specify only one output device or file.

Normally, commas separate the input files if you specify more than one. However, you can separate them by plus (+) signs if you want to combine them, as the following example shows:

```
.COPY A.FOR+B.FOR C.FOR
```

This command combines DK:A.FOR with DK:B.FOR and stores the results in DK:C.FOR.

Note that because of the file protection feature, you cannot execute any COPY operations that result in the deletion of a protected file. For example, you cannot copy a file from one volume to another if a protected file of the same name already exists on the output volume.

You can use wildcards in the input or output file specification of the command. However, the output file specification cannot contain embedded wildcards. Note that for all operations except CONCATENATE, if you use a wildcard in the input file specification, the corresponding output file name or file type must be an asterisk (*). This example uses wildcards correctly:

```
.COPY A%B.MAC *.BAK
```

In the CONCATENATE operation, the output specification must represent a single file. Therefore, no wildcards are allowed.

You can enter the COPY command as one line, or you can rely on the system to prompt you for information. If you type COPY followed by a carriage return, the system prompts *From?*. If you type the input specification followed by a carriage return, the system prompts *To?*.

The system has a special way of handling system (.SYS) files and files that cover bad blocks (.BAD files). The system requires you to use the /SYSTEM option when you need to copy system files. You cannot copy system files simply by placing wildcards in file specifications. To copy a .BAD file, you must specify it by explicitly giving its file name and file type. Since .BAD files cover bad blocks on a device, you usually do not need to copy, delete, or otherwise manipulate these files. You can copy protected files (see RENAME), but you cannot copy the protection status of a protected file (except with the COPY/DEVICE command).

NOTE

If you transfer files to a storage volume that has never been initialized with RT–11, a system failure may result.

The following sections describe the COPY command options and include command examples.

/ALLOCATE:size Use this option after the output file specification to reserve space on the device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of –1 is a special case that creates the largest file possible on the device.

/ASCII This option copies files in ASCII mode, ignoring and eliminating nulls and rubout characters. It converts data to the ASCII 7-bit format and treats CTRL/Z (32 octal) as the logical end-of-file on input. Files that consist of ASCII-format data include source files you create with the editor, map files, and list files. The following example copies a FORTRAN source program from DX0: to DX1:, giving it a new name, and reserving 50 blocks of space for it.

```
.COPY/ASCII DX0:MATRIX.FOR DX1:TEST.FOR/ALLOCATE:50
```

/BINARY Use this option to copy formatted binary files, such as .OBJ files produced by the assembler or the FORTRAN compiler, and .LDA files produced by the linker. The system verifies checksums and prints a warning if a

checksum error occurs. If this happens, the copy operation does not complete. The following command copies a binary file from DK: to a diskette.

```
.COPY/BINARY ANALYZ.OBJ DX1:*.*
```

Note that you cannot copy library files with the /BINARY option because a checksum error occurs. Copy them in image mode, or when you are creating a bootable RX01 system while the current system is on an RX02.

/BOOT[:val] This option copies bootstrap information from a monitor and handler files to blocks 0 and 2 through 5 of a random-access volume, permitting you to use that volume as a system volume. The optional argument *val* represents a two-letter target system device name that you use when you are creating a bootable PDT system volume on a PDP–11, and vice versa. You can also use this notation to create a bootable RX01 system while the current system is on an RX02 diskette. Note that you cannot combine /BOOT with any other option, and that your input and output volume must be the same. Also note that you can name your monitor file any name you wish. When you perform this operation, you must have the correct device handler to go with the volume. For example, to create a bootable RK05 disk, you must have the handler file RK.SYS on that RK05.

To create a bootable system volume, follow the procedure below:

1. Initialize the volume, using the keyboard monitor command INITIALIZE. (Note that if the volume is an RK06/07 or an RL01/02, you should also use the /REPLACE option.)
2. Copy files onto the volume, using the COPY/SYSTEM command.
3. Write the monitor bootstrap onto the volume, using COPY/BOOT.

The following example creates a system diskette.

```
.INITIALIZE DX1:
DX1:/Initialize? Are you sure? Y

.COPY/SYSTEM DX0:*.* DX1:*.*
 Files copied:
DX0:RT11SJ.SYS to DX1:RT11SJ.SYS
DX0:DT.SYS     to DX1:DT.SYS
DX0:DX.SYS     to DX1:DX.SYS
DX0:TT.SYS     to DX1:TT.SYS
DX0:LP.SYS     to DX1:LP.SYS
DX0:DIR.SAV    to DX1:DIR.SAV
DX0:DUP.SAV    to DX1:DUP.SAV
DX0:ABC.MAC    to DX1:ABC.MAC
DX0:AAF.MAC    to DX1:AAF.MAC
DX0:CT.SYS     to DX1:CT.SYS
DX0:PIP.SAV    to DX1:PIP.SAV
DX0:MT.SYS     to DX1:MT.SYS
DX0:MM.SYS     to DX1:MM.SYS
DX0:COMB.DAT   to DX1:COMB.DAT
DX0:RT11FB.SYS to DX1:RT11FB.SYS

.COPY/BOOT DX1:RT11FB.SYS DX1:
```

The device names you can use for the optional argument *val* are PD, DD, DX, and DY. The following example creates a bootable system diskette for a PDT while the current system is on a PDP-11:

```
.COPY/BOOT:PD  DX0:RT11SJ.SYS  DX0:
```

The following example creates a bootable system diskette for a PDP-11 while the current system is on a PDT-11/150:

```
.COPY/BOOT:DX  PD1:RT11SJ.SYS  PD1:
```

Note that the monitor file cannot reside on a block that contains a bad sector error (BSE) if you are doing bad block replacement. If this condition occurs, a boot error results when you bootstrap the system. In this case, move the monitor so that it does not reside on a block with a BSE error.

/CONCATENATE Use this option to combine several input files into a single output file. This option is particularly useful to combine several object modules into a single file for use by the linker or librarian. The following command combines all the .FOR files on DX1: into a file called MERGE.FOR on DX0:

```
.COPY/CONCATENATE DX1:*.FOR DX0:MERGE.FOR
 Files copied:
DX1:A.FOR       to DX0:MERGE.FOR
DX1:B.FOR       to DX0:MERGE.FOR
DX1:C.FOR       to DX0:MERGE.FOR
```

Wildcards are illegal in the output file specification.

/DEVICE This option copies block for block the image of one device to another, and copies all data from one disk to another without changing the file structure or the location of the files on the device. This is convenient because the bootstrap blocks also remain unchanged. You can copy disks that are not in RT-11 format if they have no bad blocks. When copying RT-11 disks, you can ensure the integrity of the results by making sure the disk being copied contains no bad blocks. If the system encounters a bad block during the COPY/DEVICE operation it prints an error message. When copying any disk using COPY/DEVICE, make sure the output device contains no bad blocks because this operation will write over bad blocks on the output device.

If one device is smaller than the other, the system copies only as many blocks as the smaller device contains. For example, if you copy a large volume to a smaller one, you copy the entire directory of the input volume, but not every file in the input volume. It is possible to copy blocks between disk and magtape, even though magtape is not a random-access device. The data is stored on tape formatted in 1K-word blocks. Because magtape is not file-structured, there is room for only one disk image on a magtape. The following command copies an image of DX0: to DX1:.

```
.COPY/DEVICE DX0: DX1:
DX1:/Copy?  Are you sure?  Y
```

Respond to the query message by typing Y and a carriage return. Any response not beginning with Y cancels the command and the COPY operation does not proceed.

NOTE

The COPY command does not copy track 0 of diskettes. However, this restriction has no impact on any copy operations if your diskette was supplied by DIGITAL.

/DOS Use this option to transfer files between RSTS/E or DOS–11 format and RT–11 format. The option must appear in the command line after the file to which it applies. Valid input devices are DECtape and RK05; the only valid output device is DECtape. The only other options allowed with /DOS are /ASCII, /BINARY, /IMAGE, and /OWNER:[nnn,nnn]. The following command transfers a BASIC source file from a DOS–11 disk to an RT–11 disk.

```
.COPY RK:PROG.BAS/DOS/OWNER:[200,200] SY:*.*
```

The next command copies a memory image file from an RT–11 disk to a RSTS/E format DECtape.

```
.COPY DUMP.SAV DT:*.*/DOS
```

/END:n Use with /START:n and /DEVICE to specify the last block of the volume you are copying. The /END:n notation must follow the input file specification. The argument *n* represents a decimal block number. The following example copies blocks 0 to 500 from RK0: to RK1:, starting at block 501, in a file named ADAM.MAC:

```
.COPY RK0:/START:0/END:500  RK1:ADAM.MAC/START:501
```

/EXCLUDE This option copies all the files on a device except the ones you specify. The following command copies all files from DX0: to DX1: except .OBJ and .SAV files.

```
.COPY/EXCLUDE DX0:(*.OBJ,*.SAV) DX1:*.*
```

/FILES Use with /DEVICE to copy a volume to a file on another volume or vice versa. If you use a magtape or cassette for the input volume, you must specify a file name with the input volume. This operation is useful if you wish to make several copies of a volume that is on a slow device. You can copy the volume as a file onto a volume that is on a faster device, and then proceed to make copies. Note that when you copy a file to a volume, the bootstrap and directory of the output volume are replaced by the equivalent blocks of the input file.

The following example copies diskette DX0: to DL1: as file FLOPPY.BAK:

```
.COPY/DEVICE/FILES DX0:  DL1:FLOPPY.BAK
```

The following example copies file DECTAP.BAK to DD0:

```
.COPY/DEVICE/FILES  DECTAP.BAK  DD0:
```

/IGNORE Use this option to ignore errors during a copy operation. /IGNORE forces a single-block data transfer, which you can invoke at any other time with the /SLOWLY option. Use /IGNORE if an input error

occurred when you tried to perform a normal copy operation. This procedure can sometimes recover a file that is otherwise unreadable. If there is still an error, an error message prints on the terminal, but the copy operation continues. This option is invalid with /DOS, /TOPS, and /INTERCHANGE.

/IMAGE If you enter a command line without an option, or if you use the /IMAGE option, the copy operation proceeds in image mode. Use this method to transfer memory image files and any files other than ASCII or formatted binary. Note that you cannot transfer memory image files reliably to or from paper tape, or to the line printer or console terminal. You can image-copy ASCII and binary data with the following restrictions:

1. For ASCII data, there is no check for nulls.
2. For binary data, there is no checksum consideration.

This command copies a text file to a DECtape for storage:

```
.COPY LETTER.TXT DT0:*.*
```

The primary advantage to using /IMAGE is that it is faster than /ASCII and /BINARY.

/INTERCHANGE[:size] This option transfers data in interchange format between RT-11 block-replaceable devices and interchange diskettes that are compatible with IBM 3741 format. The option must appear in the command line after the file to which it applies. If the output file is to be in interchange format, you can specify the length of each record. The argument *size* represents the record length in characters (the default record length is 80 bytes). The following command transfers the RT-11 file WAIT.MAC from device DK: to device DX1: in interchange format, giving it the name WAIT.MA. The record length is set to 128 (decimal) bytes.

```
.COPY WAIT.MAC DX1:*.*/INTERCHANGE:128.
```

/LOG This option lists on the terminal the names of the files that were copied by the current command. Normally, the system prints a log only if there is a wildcard in the file specification. If you specify /QUERY, the system prints the name of each file and asks you for confirmation before the operation proceeds. In this case, the query messages replace the log, unless you specifically type /LOG/QUERY in the command line. The following example shows a copy command line and the resulting log.

```
.COPY/LOG DX1:FILE.MAC  DX0:FILE.MAC
 Files copied:
DX1:FILE.MAC    to DX0:FILE.MAC
```

/NOLOG This option prevents a list of the files copied from appearing on the terminal.

/NEWFILES Use this option in the command line if you want to copy only those files that have the current date. The following example shows a convenient way to back up all new files after a session at the computer.

```
.COPY/NEWFILES *.* DX1:*.*
 Files copied:
DK:A.FOR        to DX1:A.FOR
DK:B.FOR        to DX1:B.FOR
DK:C.FOR        to DX1:C.FOR
```

/OWNER:[nnn,nnn] Use this option with /DOS to represent a DOS–11 user identification code (UIC) for a DOS–11 input device. Note that the square brackets are part of the UIC; you must type them. The initial default for the UIC is [1,1].

/PACKED This option copies files in DECsystem–10, DOS, or interchange mode. You can use /PACKED on an input file specification with the /TOPS, /DOS, or /INTERCHANGE option to transfer files to RT–11 format. This option transfers DECsystem–10 files created by MACY11, MACX11, or LNKX11 with the /P option.

/POSITION:n Use this option when you copy files to or from magtape or cassette. The /POSITION:n option lets you direct the tape operation; you can move the tape and perform an operation at the point you specify. For all operations, omitting the argument *n* has the same effect as setting *n* equal to 0 (*n* is interpreted as a decimal number). Since this option applies to the device and not to the files, you can specify one /POSITION:n option for the output file and one for the input files.

For magtape read (copy from tape) operations, the /POSITION:n option initiates these procedures:

1. If *n* is 0:
   The tape rewinds and the handler searches for the file you specify. If you specify more than one file, the tape rewinds before each search. If the file specification contains a wildcard, the tape rewinds only once and then the handler copies all the appropriate files.

2. If *n* is a positive integer:
   The handler looks for the file at file sequence number *n*. If the file it finds there is the one you specify, the handler copies it. Otherwise, the it prints an error message. If you use a wildcard in the file specification, the handler goes to file sequence number *n* and then begins to look for the appropriate files.

3. If *n* is –1:
   The handler starts its search at the current position. Note that if the current position is not the beginning of the tape, it is possible that the file you specify will not be found, even though it does exist on the tape.

For magtape write (copy to tape) operations, the /POSITION:n option has this effect:

1. If *n* is 0:
   The tape rewinds before the handler copies each file. A warning message prints on the terminal if the handler finds another file on the tape with the same name and file type, and the handler does not copy the file.

2. If *n* is a positive integer:
   The handler goes to file sequence number *n* or to the logical end of tape,

whichever comes first. Then it enters the file you specify. If you specify more than one file, or if you use a wildcard in the file specification, the tape does not rewind before the handler writes each file, and the handler does not check for duplicate file names. If the handler finds the sequence number *n*, it creates a new logical end of tape. If there are any files with a sequence number greater than *n*, they are lost.

3. If *n* is –1:
   The handler goes to the logical end-of-tape and enters the file you specify. It does not rewind, and it does not check for duplicate file names.

4. If *n* is –2:
   The tape rewinds between each copy operation. The handler enters the file you specify at logical end-of-tape or at the first occurrence of a duplicate file name (but if the handler enters the file over the duplicate file, you lose everything after that file).

The handler also has special procedures for handling cassettes. For cassette read (copy from tape) operations, the /POSITION:n option initiates these procedures:

1. If *n* is 0:
   The cassette rewinds and the handler searches for the file you specify. If you specify more than one file, or if you use a wildcard in the file specification, the cassette rewinds before each search.

2. If *n* is a positive integer:
   The handler starts from the cassette's present position and searches for the file you specify. If the handler does not find the file you specify before it reaches the *n*th file from its starting position, it reads the *n*th file. Note that if the starting position is not the beginning of the tape, it is possible that the handler will not find the file you specify, even though it does exist on the tape.

3. If *n* is a negative integer:
   The cassette rewinds, then the handler follows the procedure outlined in step 2 above.

For cassette write (copy to tape) operations, the /POSITION:n has this effect:

1. If *n* is 0:
   The cassette rewinds and the handler writes the file you specify at the logical end-of-tape. The handler automatically deletes any file it finds that has the same name and file type as the file you specify.

2. If *n* is a positive integer:
   The handler starts from the cassette's present position and searches *n* files ahead, deleting along the way any file it finds that has the same name and file type as the file you specify. If the handler does not reach the logical end-of-tape before it reaches the *n*th file from its starting position, it enters the file you specify over the *n*th file and deletes any files beyond it on the tape. If the handler reaches the logical end-of-tape

before it reaches the *n*th file, it writes the file you specify at the end-of-tape position.

3. If *n* is a negative integer:
   The cassette rewinds, then the handler follows the same procedure outlined in step 2 above.

Chapter 7, Section 7.2.1, contains more detailed information about operations involving magtape and cassette.

/PREDELETE This option deletes a file on the output device that has the same name as a file you copy to that device. The system deletes the file on the output device before the copy occurs. Normally, the system deletes a file of the same name after the copy operation successfully completes. This option is useful for operations involving devices that have limited space, such as diskette. Be careful when you use the /PREDELETE option; if for any reason the input file is unreadable, the output file will already have been deleted and you are left with no usable version of the file. Cassette devices are valid for input files but not for output.

/QUERY If you use this option, the system requests confirmation from you before it performs the operation. /QUERY is particularly useful on operations that involve wildcards, when you may not be sure which files the system selected for an operation. The /QUERY option is valid on the COPY command only if both input and output are in RT-11 format. Note that if you specify /QUERY in a copy command line that also contains a wildcard in the file specification, the confirmation messages that print on the terminal replace the log messages that would normally appear. You must respond to a query message by typing Y (or anything that begins with a Y) and a carriage return. The system interprets any other response to mean NO, and it does not copy the file. The following example copies three of the four FOR files stored on DK: to DX1:.

```
.COPY/QUERY DK:*.FOR DX1:*.*
 Files copied:
DK:A.FOR        to DX1:A.FOR      ? Y
DK:B.FOR        to DX1:B.FOR      ? Y
DK:C.FOR        to DX1:C.FOR      ? N
DK:DEMOF1.FOR   to DX1:DEMOF1.FOR? Y
```

/NOQUERY This option suppresses the confirmation message that the system prints for some operations, such as COPY/DEVICE. It also suppresses logging of file names if the command line contains a wildcard. You must explicitly type /LOG to obtain a list of the files copied.

/REPLACE This is the default mode of operation for the COPY command. If a file exists on the output device with the same name as the file you specify for output, the system deletes the duplicate file after the copy operation successfully completes.

/NOREPLACE This option prevents execution of the copy operation if a file with the same name as the output file you specify already exists on the output device. /NOREPLACE is valid only if both the input and output are in RT–11 format.

/SETDATE This option causes the system to put the current date on all files it transfers, unless the current system date is zero. Normally, the system preserves the existing file creation date when it copies a file block for block. This option is invalid for operations involving magtape and cassette, because the system always uses the current date for tape files.

/SLOWLY This option transfers files one block at a time. On some devices, a single-block transfer increases the chances of an error-free transfer. Use this option if a previous copy operation failed because of a read or write error.

/START[:n] Use with the /DEVICE option to specify the starting block and, with /END:n, to specify the last block of the disk you are copying. The /START:n notation must follow the input or output file specification. The argument *n* with both /START and /END represents a decimal block number.

You can use /START:n with the output file specification to specify the starting block number for the write operation on the output volume.

The following example copies blocks 500 to 550 of RK0: to RK1: starting at block 100:

```
.COPY RK0:/START:500/END:550  RK1:/START:100
```

If you do not supply a value with /START, the system assumes the first block on the volume. If you do not specify a value with /END, the system assumes the last block on the volume. Note that the first block of a file or volume is block 0.

/SYSTEM Use this option if you need to copy system (.SYS) files. If you omit this option, the .SYS files are excluded from all operations and a message is printed on the terminal to remind you.

/TOPS This option transfers files on DECsystem-10 DECtape to RT-11 format. The option must follow the input file specification. Note that DECtape is the only valid input device. You cannot perform this copy operation while a foreground job is running. Use /PACKED with /TOPS to convert from TOPS-10 7-bit ASCII format to standard PDP-11 byte ASCII format. The following command copies in ASCII format all the files named MODULE from the DECsystem-10 DECtape DT0: to RT-11 device RK0:.

```
.COPY/ASCII DT0:MODULE.*/TOPS RK0:*.*
```

/VERIFY Use this option with /DEVICE to verify that the output matches the input after a copy operation. This option cannot be used for file copy operations, only for device copy operations.

/WAIT Use this option on systems that have only a single-disk drive, or on systems that have dual drive and the system volume is neither the input nor output volume. When you use this option, the system initiates execution of a command but then pauses and prints the message *Continue?*. At this time, you can remove the system disk and mount the disk on which you want the operation to take place. When the new disk is loaded, type a Y followed by a carriage return to resume the operation. When the operation completes, the system prints the *Continue?* message again. Mount the system volume and type a Y followed by a carriage return. The system then prints the keyboard monitor prompt. Make sure PIP and DUP are on your system volume when

you use the /WAIT option. The /WAIT option is valid with /DEVICE. However, the /WAIT option cannot be used when you copy to or from non-RT-11 formatted volumes. Therefore, /WAIT cannot be used with the following options: /DOS, /INTERCHANGE, /TOPS.

Single-Volume Operation

If you want to transfer a file between two storage volumes, and you have only one drive for that type of storage volume, follow the procedure below.

1. Enter a command string according to this general syntax:

   COPY/WAIT input-filespec output-filespec

   where *output-filespec* represents the destination device and file specification, and *input-filespec* represents the source device and file specification.

2. The system responds by printing the following message at the terminal.

   ```
   Mount input volume in <device>?  Continue?
   ```

   <device> represents the device into which you are to mount your input volume. Type a Y followed by a carriage return after you have mounted your input volume.

3. The system continues the copy procedure and prints the following message on the terminal:

   ```
   Mount output volume in <device>?  Continue?
   ```

4. After you have removed your input volume from the device, mount your output volume, then type Y followed by a carriage return.

5. Depending on the size of the file, the system may repeat the transfer cycle (steps 2 and 3) several times before the transfer is complete. When the transfer is complete, the system prints the following prompt at the terminal:

   ```
   Mount system volume in <device>?  Continue?
   ```

   When you mount your system volume and type a Y followed by a carriage return in response to the last instruction, you terminate the copy operation.

Double-Volume Operation

If you have a small disk system, you can use the /WAIT option for transferring files between two non-system volumes. The procedure for transferring files this way follows.

1. With your system volume mounted, enter a command according to the following general syntax:

   COPY/WAIT input-filespec output-filespec

   where *output-filespec* represents the destination device and file specification, and *input-filespec* represents the source device and file specification.

2. After you have entered the last command string, the system responds with the following prompt:

   ```
   Mount input volume in <device>?  Continue?
   ```

Type a Y followed by a carriage return when you have mounted the input volume.

3. The system then prints the next instruction for you to mount the output volume:

```
Mount output volume in <device>;  Continue?
```

Type a Y followed by a carriage return in response to the last message after you have mounted the output volume.

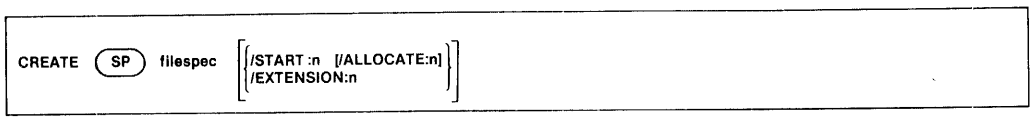
4. Unlike the single-volume transfer, the double-volume transfer involves only one cycle of mounting the input and output volumes. When the file transfer is complete, PIP prints the following instruction:

```
Mount system volume in <device>;  Continue?
```

When you mount your system volume and type a Y followed by a carriage return in response to the last instruction, you terminate the copy operation.

CREATE

The CREATE command creates or extends a file with a specific name, location, and size on the block replaceable volume that you specify.



In the command syntax illustrated above, *filespec* represents the device and file specifications of the file you wish to create or extend. If you are using the CREATE command to create a file, this command only creates a directory entry for the file. This command does not store any data in a file. You must specify both the file name and type of the file you wish to create or extend.

If you type a carriage return after typing CREATE, the system prompts *File?*.

The following sections describe the options you can use with the CREATE command.

/ALLOCATE:n Use this option following the file specification to allocate *n* blocks for the file you are creating, where *n* represents a decimal number. A value of -1 for *n* indicates a file of the maximum size available on the volume. If you do not use /ALLOCATE, the system assumes one block.

/EXTENSION:n Use this option to extend an existing file you specify by *n* blocks, where *n* is a decimal number. When you use this option following the file specification, make sure that there is enough unused space on the volume for the size you specify (use the DIRECTORY/FULL command to do this). If you do not supply a value with /EXTENSION, the system assumes one block.

The following example illustrates the procedure for extending a file with the CREATE command. In this example, BUILD.MAC is extended by 20 blocks. First, a DIRECTORY/FULL command determines whether there is available space adjacent to BUILD.MAC.

```
.DIRECTORY/FULL DX0:
 05-DEC-79
MYPROG.MAC      36P  19-NOV-79        TM      .MAC      25  27-NOV-79
VTMAC .MAC       7   19-NOV-79        SYSMAC.MAC        41  19-NOV-79
< UNUSED >      25                    RT11SJ.SYS        67  19-NOV-79
TT     .SYS      2   19-NOV-79        DX      .SYS       3  19-NOV-79
LELA   .LBM      1   05-DEC-79        BUILD  .MAC       80  19-NOV-79
< UNUSED >     199
 9 Files, 262 Blocks
 224 Free blocks
```

Next the CREATE command extends BUILD.MAC by 20 blocks.

```
.CREATE DX0:BUILD.MAC/EXTENSION:20
```

/START:n Use this option to specify the starting block number of the file you are creating. The argument *n* represents a decimal block number. If you do not use /START, the system uses the first available space on the volume.

The following example illustrates the procedure for creating a file with the CREATE command. In this example, SWAP.SYS is restored after having been previously deleted. First, a DIRECTORY/DELETED command establishes the starting block numbers of the deleted files on DX0:

```
.DIRECTORY/DELETED DX0:
 05-DEC-79
SWAP   .SYS     25  19-NOV-79    117   EMPTY.FIL    179  31-OCT-79    315
 0 Files, 0 Blocks
 204 Free blocks
```

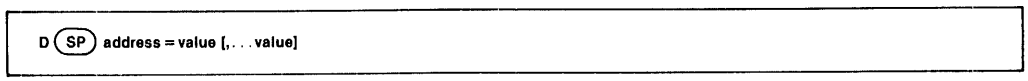
Next, the CREATE command restores SWAP.SYS, starting at block 117, and using the /ALLOCATE:n option to allocate 25 blocks.

```
.CREATE DX0:SWAP.SYS/START:117/ALLOCATE:25
```

See the *RT–11 Software Support Manual* for a detailed description of the RT–11 file structure.

D

The D (Deposit) command deposits values in memory, beginning at the location you specify.



In the command syntax illustrated above, *address* represents an octal address that, when added to the relocation base value from the Base command (if you used one), provides the actual address where the system must deposit the values. The argument *value* represents the new contents of the address. If you do not specify a value, the system assumes a value of 0. If you specify more than one value and separate the values by commas, the system deposits the values in sequential locations, beginning at the location you specify.

The Deposit command accepts both word and byte addresses, but it always executes the command as though you specified a word address. (If you specify an odd address, the system decreases it by one to make it even.) The Deposit command stores all values as word quantities.

Use commas to separate multiple values in the command line. Two or more adjacent commas cause the system to deposit zeroes at the location you specify and at the following locations, if indicated.

Note that you cannot specify an address that references a location outside the area of the background job. You can use the D command with GET and START to temporarily alter a program's execution. Use the SAVE command before START to make the alteration permanent.

The following command deposits zeroes into locations 300, 302, 304, and 306.

```
.D 300=,,,
```

The next command sets the base address to 0.

```
.B
```

The following command deposits 3705 into location 1000.

```
.D 1000=3705
```

The next command sets the relocation base to 1000.

```
.B 1000
```

The next command puts 2503 into location 1500 (offset of 500 from the last B command) and 22 into location 1502.

```
.D 500=2503,22
```

DATE

Use the DATE command to set or to inspect the current system date.

```
DATE [(SP) dd-mmm-yy]
```

In the command syntax shown above, *dd* represents the day (a decimal number from 1 to 31); *mmm* represents the first three characters of the name of the month; and *yy* represents the year (a decimal number from 73 to 99).

To enter a date into the system, as soon as you bootstrap the system specify the date in the format described above. The system uses this date for newly created files, for files that you transfer to magtape or cassette, and for listing files. The following example enters the current date.

```
.DATE 18-MAY-77
```

To display the current system date, type the DATE command without an argument, as this example shows.

```
.DATE
18-May-77
```

The FB and XM monitors automatically increment the date at midnight each day. The SJ monitor increments the date only if you select timer support as a system generation special feature. Note that you can also select automatic end-of-month date advancement through system generation.

DEASSIGN

The DEASSIGN command disassociates a logical device name from a physical device name.

```
DEASSIGN [(SP) logical-device-name]
```

In the command syntax illustrated above, *logical-device-name* represents an alphanumeric name, from one to three characters long, that is assigned to a particular device. Note that spaces and tabs are not permitted in the logical device name.

To remove the assignment of a particular logical device name to a physical device, specify that logical device name in the command line. The following example disassociates the logical name INP: from the physical device to which it is assigned.

```
.DEASSIGN INP:
```

If you specify a logical name that is not currently assigned, the system prints an error message, as this example shows.

```
.DEASSIGN INP:
?KMON-F-Logical name not found
```

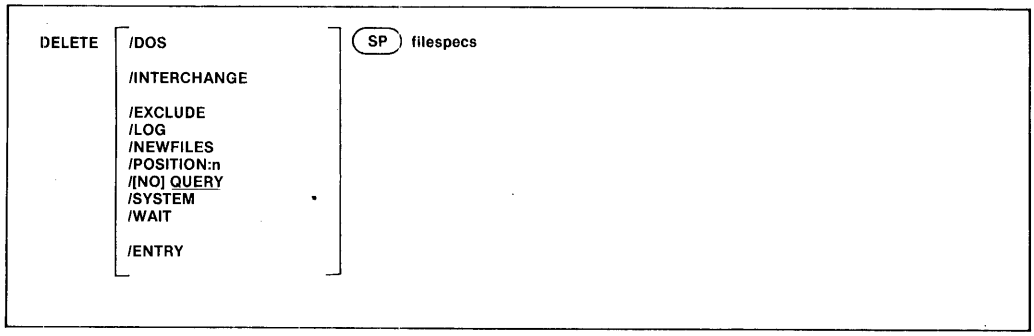
To disassociate all logical names from physical devices, type the DEASSIGN command without an argument. The following example disassociates all logical device names (except SY:) from physical devices and resets the logical names DK: and SY: to represent the system volume.

```
.DEASSIGN
```

If DK: is assigned to a non-system device (such as DX1:, for example), the following command disassociates DK: from DX1: and restores the default association of DK: to SY:, the system device.

```
.DEASSIGN DK:
```

The DELETE command deletes the files you specify.



In the command syntax shown above, *filespecs* represents the files to be deleted. You can specify up to six files; separate them with commas. You can enter the DELETE command as one line, or you can rely on the system to prompt you for information. If you omit the file specification, the DELETE command prompts *Files?*. If you delete a file accidentally, it may be possible to recover the file if you act immediately (see CREATE). A procedure for doing this is described in Chapter 8.

The system has a special way of handling system (.SYS) files and files that cover bad blocks (.BAD files). So that you do not delete system files by accident when you use a wildcard in the file specification, the system requires you to use the /SYSTEM option when you need to delete system files. To delete a .BAD file, you must specify it by explicitly giving its file name and file type. Since .BAD files cover bad blocks on a device, you do not need to copy, delete, or otherwise manipulate these files. To delete a protected file (a "P" next to the block size of a file's directory entry denotes protection) use the RENAME/NOPROTECTION command.

Another feature of the DELETE command is that the system unless using /LOG or /NOQUERY requests confirmation from you before it deletes a file. You must respond to the query message by typing Y followed by a carriage return in order to execute the command.

The following sections describe the options you can use with the DELETE command.

/DOS Use this option to delete a file that is in DOS–11 or RSTS/E format. The valid devices for this type of file are disks or DECtapes. You cannot use any other option in combination with /DOS.

/ENTRY Use this option to delete a job from the queue. Use /ENTRY when QUEUE is running as a foreground or system job (see Chapter 20, Queue Package).

When you use /ENTRY, you do not have to specify the input files in the job, only the job name. If you have not specified a job name, the system uses the first file name in the job as the job name. The following example deletes MILLER from the queue:

```
.DELETE/ENTRY MILLER
```

If QUEUE is printing a job when you delete that job, QUEUE immediately stops processing that job.

/EXCLUDE This option deletes all the files on a device except the ones you specify. The following command, for example, deletes all files from DX0: except .SAV files.

```
.DELETE/EXCLUDE DX0:*.SAV
?PIP-W-No .SYS action
 Files deleted:
DX0:ABC.OLD    ? Y
DX0:AAF.OLD    ? Y
DX0:COMB.      ? Y
DX0:MERGE.OLD  ? Y
```

/INTERCHANGE Use this option to delete from a diskette a file that is in interchange format. You cannot use any other option with /INTERCHANGE.

/LOG This option lists on the terminal a log of the files that are deleted by the current command. Note that if you specify /LOG, the system does not ask you for confirmation before execution proceeds (that is, /LOG implies /NOQUERY). Use both /LOG and /QUERY to invoke logging and querying.

/NEWFILES Use this option to delete only the files that have the current system date. This is a convenient way to remove all the files that you just created in a session at the computer. The following example deletes the files created today.

```
.DELETE/NEWFILES DX1:*.BAK
 Files deleted:
DX1:MERGE.BAK ? Y
```

/POSITION[:n] You can use this option when you delete files from cassette. It permits you to move the tape and perform an operation at the point you specify. Omitting the argument *n* has the same effect as setting *n* equal to 0 (*n* is interpreted as a decimal number). The /POSITION:n option has the following effect:

1. If *n* is 0:
   The cassette rewinds and the system searches for the file you specify. If you specify more than one file, or if you use a wildcard in the file specification, the cassette rewinds before each search.

2. If *n* is a positive integer:
   The system starts from the cassette's present position and searches for the file you specify. If the system does not find the file you specify before it reaches the *n*th file from its starting position, it deletes the *n*th file.

Note that if the starting position is not the beginning of the tape, it is possible that the system will not find the file you specify, even though it does exist on the tape.

3. If *n* is a negative integer:
   The cassette rewinds, then the system follows the procedure outlined in step 2 above.

/QUERY Use this option to request a confirmation message from the system before it deletes each file. This option is particularly useful on operations that involve wildcards, when you may not be completely sure which files the system selected for the operation. This is the default mode of operation. Note that specifying /LOG eliminates the automatic query; you must specify /QUERY with /LOG to retain the query function. You must respond to a query message by typing Y (or anything that begins with a Y) and a carriage return to initiate execution of a particular operation. The system interprets any other response as NO; it does not perform the operation. The following example shows querying. Only one file is deleted.

```
.DELETE/QUERY DX1:*.*
 Files deleted:
DX1:ABC.MAC    ? N
DX1:AAF.MAC    ? Y
DX1:MERGE.FOR  ? N
```

/NOQUERY This option suppresses the confirmation message the system prints before it deletes each file.

/SYSTEM Use this option if you need to delete system (.SYS) files. If you omit this option, the system files are excluded from the DELETE operation, and a message is printed on the terminal. (Note that the system prints this message only when system files might otherwise be included in the operation.)

/WAIT This option is useful if you have a single-disk system. When you use this option, the system initiates the DELETE operation but then pauses for you to mount the volume on which you want the operation to take place. When the system pauses, it prints *Mount input volume in <device>; Continue?*, where *<device>* represents the device into which you mount the volume. When the volume is mounted, type Y followed by a carriage return.

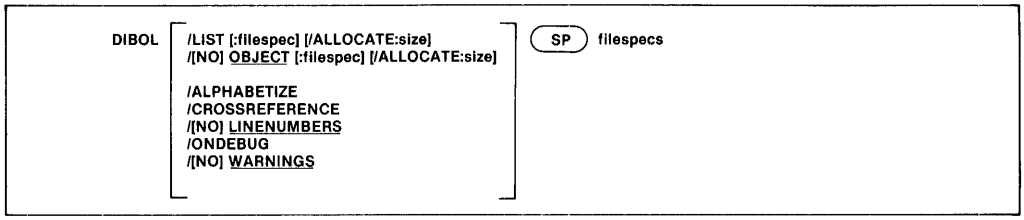
The following example deletes FILE.MAC from an RK05 disk:

```
.DELETE/WAIT RK0:FILE.MAC
Mount input volume in RK0:; Continue? Y
RK0:FILE.MAC? Y
Mount system volume in RK0:; Continue? Y
```

This option is invalid with /INTERCHANGE and /DOS.

DIBOL

The DIBOL command invokes the DIBOL compiler to compile one or more source programs.



In the command syntax illustrated above, *filespecs* represents one or more files to be included in the compilation. If you omit a file type for an input file, the system assumes .DBL. Output default file types are .LST for listing files and .OBJ for object files. To compile multiple source files into a single object file, separate the files by plus (+) signs in the command line. Unless you specify otherwise, the system creates an object file with the same name as the first input file and gives it an .OBJ file type. To compile multiple files in independent compilations, separate the files by commas (,) in the command line. This generates a corresponding object file for each set of input files.

Language options are position-dependent — that is, they have different meanings depending on where you place them in the command line. Options that qualify a command name apply across the entire command string. Options that follow a file specification apply only to the file (or group of files separated by plus signs) they follow in the command string.

You can enter the DIBOL command as one line, or you can rely on the system to prompt you for information. The DIBOL command prompt is *Files?* for the input specification.

The *DIBOL–11 Language Reference Manual* contains more detailed information about using DIBOL. The following sections describe the options you can use with the DIBOL command.

/ALLOCATE:size Use this option with /LIST or /OBJECT to reserve space on the device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of –1 is a special case that creates the largest file possible on the device.

/ALPHABETIZE Use this option to alphabetize entries in the symbol and label tables. This is useful for program maintenance and debugging.

/CROSSREFERENCE This option generates a symbol cross-reference section in the listing to which it adds as many as four separate sections to the listing. These sections are: (1) symbol cross-reference table, (2) label cross-reference table, (3) external subroutine cross-reference table, (4) COMMON cross-reference table. Note that the system does not generate a

listing by default. You must also specify /LIST in the command line to get a cross-reference listing.

/LINENUMBERS This option generates line numbers for the program during compilation. These line numbers are referenced by the symbol table segment, label table segment, and the cross-reference listing; they are especially useful in debugging DIBOL programs. This is the default operation.

/NOLINENUMBERS This option suppresses the generation of line numbers during compilation, which produces a smaller program and optimizes execution speed. Use this option to compile only programs that are already debugged; otherwise the DIBOL error messages are difficult to interpret.

/LIST[:filespec] You must specify this option to produce a DIBOL compilation listing. The /LIST option has different meanings depending on where you place it in the command line.

The /LIST option produces a listing on the line printer when /LIST follows the command name. For example, the following command line produces a line printer listing after compiling a DIBOL source file:

```
.DIBOL/LIST MYPROG<RET>
```

When the /LIST option follows the file specification, it produces a listing file. For example, the following command line produces the listing file DK:MYPROG.LST after compiling a DIBOL source file:

```
.DIBOL MYPROG/LIST<RET>
```

If you specify /LIST in the list of options that immediately follows the command name, but omit a file specification, the DIBOL compiler generates a listing that prints on the line printer. If you follow /LIST with a device name, the system creates a listing file on that device. If the device is a file-structured device, the system stores the listing file on that device, assigning it the same name as the input file with a .LST file type. The following command produces a listing on the terminal.

```
.DIBOL/LIST:TT: A
```

The next command creates on RK3: a listing file called A.LST.

```
.DIBOL/LIST:RK3: A
```

If the /LIST option contains a name and file type to override the default of .LST, the system generates a listing file with that name. The following command, for example, compiles A.DBL and B.DBL together, producing on device DK: files A.OBJ and FILE1.OUT:

```
.DIBOL/LIST:FILE1.OUT A+B
```

Another way to specify /LIST is to type it after the file specification to which it applies. To produce a listing file with the same name as a particular input file, you can use a command similar to this one:

```
.DIBOL A+B/LIST:RK3:
```

The command shown above compiles A.DBL and B.DBL together, producing files DK:A.OBJ and RK3:B.LST. If you specify a file name on a /LIST option following a file specification in the command line, it has the same meaning as when it follows the command. The following two commands have the same results:

```
.DIBOL A/LIST:B

.DIBOL/LIST:B A
```

Both commands generate as output files A.OBJ and B.LST.

Remember that file options apply only to the file (or group of files that are separated by plus signs) they follow in the command string. For example:

```
.DIBOL A/LIST,B
```

This command compiles A.DBL, producing A.OBJ and A.LST. It also compiles B.DBL, producing B.OBJ. However, it does not produce any listing file for the compilation of B.DBL.

/OBJECT[:filespec] Use this option to specify a file name or device for the object file. Because DIBOL creates object files by default, the following two commands have the same meaning:

```
.DIBOL A

.DIBOL/OBJECT A
```

Both commands compile A.DBL and produce A.OBJ as output. The /OBJECT option functions like the /LIST option; it can be either a command or a file qualifier.

As a command option, /OBJECT applies across the entire command string. The following command, for example, compiles A.DBL and B.DBL separately, creating object files A.OBJ and B.OBJ on RK1:.

```
.DIBOL/OBJECT:RK1: A,B
```

Use /OBJECT as a file option to create an object file with a specific name or destination. The following command compiles A.DBL and B.DBL together, creating files B.LST and B.OBJ.

```
.DIBOL A+B/LIST/OBJECT
```

/NOOBJECT Use this option to suppress creation of an object file. As a command option, /NOOBJECT suppresses all object files; as a file option, it suppresses only the object file produced by the related input files. In this command, for example, the system compiles A.DBL and B.DBL together, producing files A.OBJ and B.LST. It also compiles C.DBL and produces C.LST, but does not produce C.OBJ.

```
.DIBOL A+B/LIST, C/NOOBJECT/LIST
```

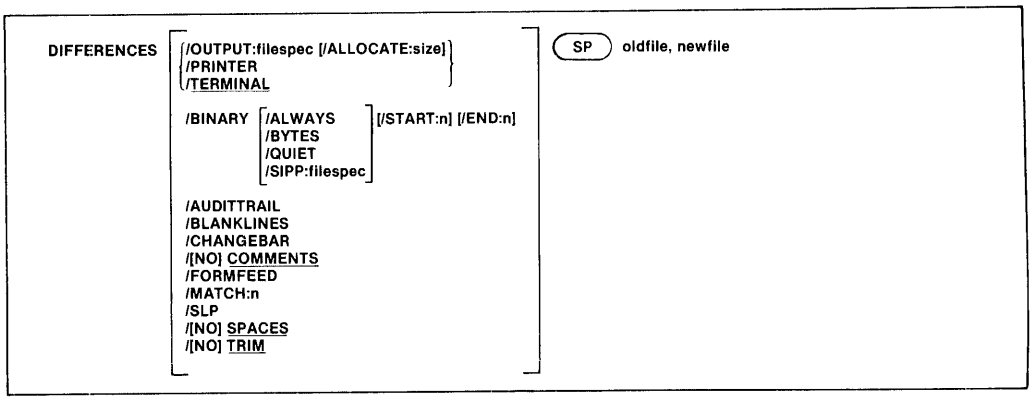
/ONDEBUG This option includes a symbol table in the object file. You can then use a debugging program to find and correct errors in the object file.

/WARNINGS Use this option to include warning messages in DIBOL compiler diagnostic error messages. These messages call certain conditions to your attention, but they do not interfere with the compilation. This is the default operation.

/NOWARNINGS Use this option to suppress warning messages during compilation.

DIFFERENCES

The DIFFERENCES command compares two files and lists the differences between them.



In the command syntax shown above, *oldfile* represents the first file to be compared and *newfile* represents the second. The default output device is the console terminal. The default file type for input files is .MAC; for output files it is .DIF. You can specify the entire command on one line, or you can rely on the system to prompt you for information. The DIFFERENCES command prompts are *File 1?* and *File 2?*.

The DIFFERENCES command is particularly useful when you want to compare two similar versions of a source or binary program, typically, an updated version against a backup version. A file comparison listing highlights the changes made to a program during an editing session. The following sections describe the various options you can use with the DIFFERENCES command. Following the descriptions of the options is a sample listing and an explanation of how to interpret it.

The DIFFERENCES command is also useful for creating command files that can install patches to backup versions of programs so they match the updated versions. The /SLP and /SIPP;filespec options are designed especially for this purpose.

/ALLOCATE:size Use this option with /OUTPUT and /SIPP to reserve space on the device for the output listing file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of -1 is a special case that creates the largest file possible on the device.

/ALWAYS When you use this option with /BINARY or /SIPP:filespec, the system creates an output file regardless of whether there are any differences between the two input files. This option is useful when running BATCH streams to prevent job step failures due to the absence of a DIFFERENCES output file.

The /ALWAYS option is position dependent. That is, you must use it immediately after the output file to which you want it to apply. If you use it at the end of the command string, it applies to all output files.

/AUDITTRAIL Use this option with /SLP to specify an audit trail. The /SLP option, described below, creates a command file which, when run with the source language patch program (SLP), can patch *oldfile* so it matches *newfile*. When you use SLP to modify a file, it creates an output file that has audit trails. An audit trail is a string of characters that appears in the right margin of each line that has been changed by the modification procedure. The audit trail keeps track of the patches you make to the patched source file.

By default, SLP uses the following characters for the audit trail:

```
;**NEW**
```

When you use the /AUDITTRAIL option, the system prints the following prompt at the terminal.

```
Audit trail?
```

Enter a string of up to 12 ASCII characters that you want to use in place of the default audit trail. Do not use the slash (/) in the audit trail.

/BINARY When you use this option, the system compares two binary files and lists the differences between them. This option is useful for comparing and relocatable image files (that is, machine runnable programs and object files) and provides a quick way of telling whether two files are identical. For example, you can use /BINARY to tell whether two versions of a program produce identical output.

When you use /BINARY and do not specify an output file, the system prints output at the terminal according to the following general syntax:

```
bbbbbb  ooo/  ffffff  ssssss  xxxxxx
```

where:

bbbbbb	represents the octal block number of the block that contains the difference
ooo	represents the octal offset within the block that contains the difference
ffffff	represents the value in the first file you are comparing
ssssss	represents the value in the second file you are comparing
xxxxxx	represents the logical exclusive OR of the two values in the input files

If you use the /OUTPUT:filespec option with /BINARY, the system stores the differences listing in the file you specify (if there are any differences found), instead of printing the differences at the terminal.

DIFFERENCES

/BLANKLINES Use this option to include blank lines in the file comparison. Normally, the system disregards blank lines.

/BYTES When you use this option with /BINARY, the system lists the differences byte-by-byte.

/CHANGEBAR Use this option to create an output file that contains *newfile* with a changebar character next to the lines in *newfile* that differ from *oldfile*. The system inserts a vertical bar next to each line that has been added to *newfile*, and a bullet (lower-case letter o) next to each line that has been deleted.

The output defaults to the terminal. Use the /PRINTER option to list the output to the line printer. Specify an output file with the /OUTPUT:filespec option.

The sample that follows creates a listing of RTLIB.MAC with a changebar character at the left margin of each line that is different from RTLIB.BAK:

```
DIFFERENCES/CHANGEBAR RTLIB.BAK,RTLIB.MAC
```

/COMMENTS When you use this option, the system includes in the file comparison all assembly language comments it finds in the two files. (Comments are preceded by a semicolon on the same line.) This is the default operation.

/NOCOMMENTS Use this option to exclude comments from the comparison. (Comments are preceded by a semicolon on the same line.) This is useful if you are comparing two MACRO source programs with similar content but different format.

/END[:n] Use this option with /BINARY to specify the ending block number of the file comparison, where *n* is an octal number that represents the ending block number. If you do not supply a value with /END, the system defaults to the last block of the file or volume.

/FORMFEED Use this option to include form feeds in the output listing. Normally, the system compares form feeds but does not include them in the output listing.

/MATCH[:n] Use this option to specify the number of lines from each file that must agree to constitute a match. The value *n* is an integer in the range 1-200. The default value for *n* is 3.

/OUTPUT:filespec Use this option to specify a device and file name for the output listing file. Normally, the listing appears on the console terminal. If you omit the file type for the listing file, the system uses .DIF. Note that the system creates this file only if there are any differences found. Use the /ALWAYS option if you want the system to create an output file regardless of whether any differences are found.

/PRINTER Use this option to print a listing of differences on the printer. Normally, the listing appears on the console terminal.

/QUIET When you use this option with /BINARY, the system suppresses printing the differences at the terminal and prints *?BINCOM-W-Files are different,* if applicable.

/SIPP:filespec Use this option with /BINARY to output a file that you can use as an input command file to the save image patch program SIPP, where *filespec* represents the name of the output file.

The file you create with /SIPP can patch *oldfile* so it matches *newfile.*

The example that follows creates an input command file which, when run with SIPP, patches DEMOF1.BAK so it matches DEMOF1.SAV.

```
DIFFERENCES/BINARY/SIPP:PATCH.COM  DEMOF1.BAK, DEMOF1.SAV
```

To execute the input command file created by /SIPP, see Chapter 22, Save Image Patch Program (SIPP).

/SLP Use this option with /OUTPUT:filespec to create a command file that, when run with the source language patch utility SLP, patches *oldfile* to match *newfile.* If you do not use the /OUTPUT:filespec option with /SLP, the system prints the command file at the terminal.

The sample that follows creates the command file PATCH.COM. PATCH.COM can be used as input to the program SLP to patch RTLIB.BAK so that it matches RTLIB.MAC.

```
.DIFFERENCES/SLP/OUTPUT:PATCH.COM RTLIB.BAK,RTLIB.MAC
```

To execute the command file you create with /SLP, see Chapter 24, Source Language Patch Program (SLP).

/SPACES This option includes spaces and tabs in the file comparison. This is the default operation and is particularly useful when you are comparing two text files and must pay careful attention to spacing.

/NOSPACES Use this option to exclude spaces and tabs from the file comparison. This is useful when you are comparing two source programs with similar contents but different formats.

/START[:n] Use this option with /BINARY to specify the starting block number of the file comparison, where *n* represents the octal starting block number. If you do not supply a value with /START, the system defaults to the first block in the file.

/TERMINAL Use this option to cause the list of differences to appear on the console terminal. This is the default operation.

To understand how to interpret the output listing, first look at the following two text files.

```
.TYPE FILE1.TXT
HERE'S A BOTTLE AND AN HONEST FRIEND!
  WHAT WAD YE WISH FOR MAIR, MAN?
WHA KENS, BEFORE HIS LIFE MAY END,
  WHAT HIS SHAME MAY BE O' CARE, MAN?
```

```
THEN CATCH THE MOMENTS AS THEY FLY,
  AND USE THEM AS YE OUGHT, MAN: --
BELIEVE ME, HAPPINESS IS SLY,
  AND COMES NOT AY WHEN SOUGHT, MAN.

                          --SCOTTISH SONG


.TYPE FILE2.TXT
HERE'S A BOTTLE AND AN HONEST FRIEND!
  WHAT WAD YE WISH FOR MAIR, MAN?
WHA KENS, BEFORE HIS LIFE MAY END,
  WHAT HIS SHARE MAY BE O' CARE, MAN?
THEN CATCH THE MOMENTS AS THEY FLY,
  AND USE THEM AS YE OUGHT, MAN: --
BELIEVE ME, HAPPINESS IS SHY,
  AND COMES NOT AY WHEN SOUGHT, MAN.

                          --SCOTTISH SONG
```

Notice that in the fourth line of FILE1.TXT, *shame* should be *share*; in the seventh line, *sly* should be *shy*.

The following command compares the two files, creating a listing file called DIFF.TXT.

```
.DIFFERENCES/MATCH:1/OUTPUT:DIFF.TXT FILE1.TXT,FILE2.TXT
?SRCCOM-W-Files are different
```

The following listing shows file DIFF.TXT.

```
.TYPE DIFF.TXT
1) DK:FILE1.TXT
2) DK:FILE2.TXT
**********
1)1         WHAT HIS SHAME MAY BE O' CARE, MAN?
1)        THEN CATCH THE MOMENTS AS THEY FLY,
****
2)1         WHAT HIS SHARE MAY BE O' CARE, MAN?
2)        THEN CATCH THE MOMENTS AS THEY FLY,
**********
1)1       BELIEVE ME, HAPPINESS IS SLY,
1)          AND COMES NOT AY WHEN SOUGHT, MAN.
****
2)1       BELIEVE ME, HAPPINESS IS SHY,
2)          AND COMES NOT AY WHEN SOUGHT, MAN.
**********
```

If the files are different, the system always prints the file name of each file as identification:

```
1) DK:FILE1.TXT
2) DK:FILE2.TXT
```

The numbers at the left margin have the form *n)m*, where *n* represents the source file (either 1 or 2) and *m* represents the page of that file on which the specific line is located.

The system next prints ten asterisks and then lists the differences between the two files. The /MATCH:n option was used in this example to set to 1 the number of lines that must agree to constitute a match.

The first three lines of the song are the same in both files, so they do not appear in the listing. The fourth line contains the first discrepancy. The system prints the fourth line from the first file, followed by the next matching line as a reference.

```
1)1          WHAT HIS SHAME MAY BE O' CARE, MAN?
1)         THEN CATCH THE MOMENTS AS THEY FLY,
****
```

The four asterisks terminate the differences section from the first file.

The system then prints the fourth line from the second file, again followed by the next matching line as a reference:

```
2)1          WHAT HIS SHARE MAY BE O' CARE, MAN?
2)         THEN CATCH THE MOMENTS AS THEY FLY,
**********
```

The ten asterisks terminate the listing for a particular difference section.

The system scans the remaining lines in the files in the same manner. When it reaches the end of each file, it prints *?SRCCOM-W-Files are different* on the terminal.

If you compare two files that are identical, the system does not create an output listing, but prints:

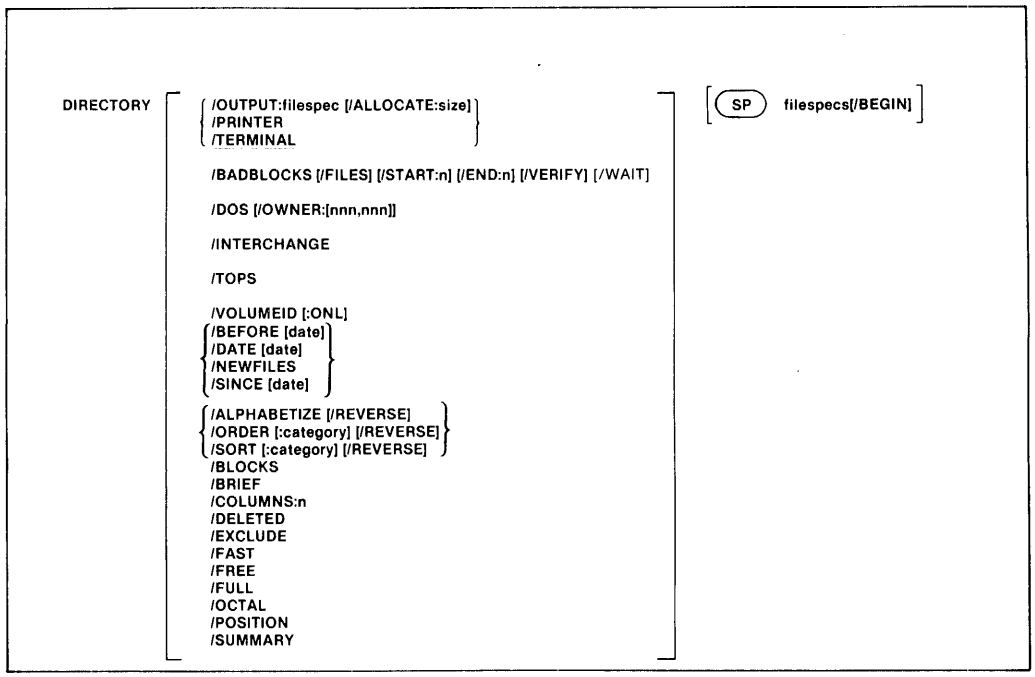
```
?SRCCOM-I-No differences found
```

/TRIM Use the /TRIM option with /SLP to ignore tabs and spaces that appear at the ends of source lines. This is the default setting.

/NOTRIM Use /NOTRIM with /SLP to include in the comparison spaces and tabs that appear at the ends of source lines. /TRIM is the default setting.

DIRECTORY

The DIRECTORY command lists information you request about a device, a file, or a group of files.



In the command syntax shown above, *filespecs* represents the device, file, or group of files whose directory information you request. The DIRECTORY command can list directory information about a specific device, such as the number of files stored on the device, their names, and their creation dates. It can list details about certain files including their names, their file types, and their size in blocks. You can specify up to six files explicitly, but you can obtain directory information about many files by using wildcards in the file specification. The DIRECTORY command can also print a device directory summary, organized in several ways, such as alphabetical or chronological.

Normally, the DIRECTORY command prints listings in two columns on the terminal. Read these listings as you would read a book; read across the columns, moving from left to right, one row at a time. Directory listings that are sorted (with /ALPHABETIZE, /ORDER, or /SORT) are an exception to this. Read these listings as you would a telephone directory, by reading the left column from top to bottom, then reading the right column from top to bottom.

The DIRECTORY command does not prompt you for any information. If you omit the file specification, the system lists directory information about device DK:, as this example shows.

```
.DIRECTORY
 27-Nov-79
RT11SJ.SYS     67P 03-Jul-79       RT11FB.SYS     80P 13-Aug-79
RT11BL.SYS     63P 15-Mar-79       DX     .SYS     3P 13-Aug-79
SWAP   .SYS    25P 13-Aug-79       TT     .SYS     2P 13-Aug-79
DP     .SYS     3P 13-Aug-79       DY     .SYS     4P 13-Aug-79
LP     .SYS     2P 27-Nov-79       PIP    .SAV    16  25-Jul-79
DUP    .SAV    41  26-Mar-79       RESORC.SAV    15  13-Aug-79
EDIT   .SAV    19  13-Aug-79       STARTS.COM     1  27-Aug-79
SIPP   .SAV    14  13-Aug-79
 15 Files, 413 Blocks
 73 Free blocks
```

A "P" next to the block size number of a file's directory entry indicates that the file is protected from deletion (see RENAME/PROTECTION).

If you specify only a device in the file specification, the system lists directory information about all the files on that device. If you specify a file name, the system lists information about just that file, as this example shows.

```
.DIRECTORY DX0:RT11FB.SYS
 10-Dec-79
RT11FB.SYS     80P 13-Aug-79
 1 File, 80 Blocks
 4 Free blocks
```

The following sections describe the options you can use with the DIRECTORY command and provide sample directory listings. Some of the options accept a date or part of a date as an argument. The syntax for specifying the date is:

[:dd][:mmm][:yy]

where:

dd	represents the day (a decimal integer in the range 1–31)
mmm	represents the first three characters of the name of the month
yy	represents the year (a decimal integer in the range 73–99)

The default value for the date is the current system date. If you specify just the day, the system interprets it as the given day of the current month and year. If you specify just the month, the system interprets it to be the first day of the given month in the current year. If you specify only the year, the system interprets it as the start of that year. If the current system date is not set, it is considered 0 (the same as for an undated file in a directory listing).

If you have selected timer support through the system generation process, but have not selected automatic end-of-month date advancement, make sure that you set the date of the beginning of each month with the DATE command. If you fail to set the date at the beginning of each month, the system prints *-BAD-* in the creation date column of each file created beyond the end-

of-month. (Note that you can eliminate *-BAD-* by using the RENAME/SETDATE command after you set the date.)

/ALLOCATE:size Use this option with /OUTPUT to reserve space on the device for the output listing file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of -1 is a special case that creates the largest file possible on the device.

/ALPHABETIZE This option lists the directory of the device you specify in alphabetical order by file name and file type. It has the same effect as the /ORDER:NAME option. Note that this option sorts numbers after letters.

/BADBLOCKS Sometimes devices (disks and DECtapes) have bad blocks, or they develop bad blocks as a result of use and age. Use the /BADBLOCKS option to scan a device and locate bad blocks on it. The system prints the absolute block number of these blocks on the devices that return hardware errors when the system tries to read them. This procedure does not destroy data that is already stored on the device. Remember that block numbers are listed in both octal and decimal and the first block on a device is block 0. If a device has no bad blocks, an informational message prints on the terminal.

```
.DIRECTORY/BADBLOCKS DX1:
?DUP-I-No bad blocks detected
```

If /BADBLOCKS is the only option in the command line, the volume being scanned does not need a valid RT–11 directory structure.

/BEFORE[date] This option prints a directory of files created before the date you specify. The following command lists on the terminal all files stored on device DX1: that were created before December 1979.

```
.DIRECTORY/BEFORE:DEC DX1:
 14-Dec-79
MYPROG.MAC     36P  19-Nov-79      TM     .MAC     25  27-Nov-79
VTMAC .MAC      7   19-Nov-79      SYSMAC.MAC      41  19-Nov-79
RT11SJ.SYS      0   19-Nov-79      RT11SJ.SYS      67  19-Nov-79
TT    .SYS      2   19-Nov-79      DX     .SYS      3  19-Nov-79
BUILD .MAC    100   19-Nov-79
 9 Files, 281 Blocks
 180 Free blocks
```

/BEGIN This option lists the directory of the device you specify, beginning with the file you name and including all the files that follow it in the directory. The occurrence of file names in the listing is the same as the order of the files on the device.

The following example lists the file VTMAC.MAC on device DX0: and all the files that follow it in the directory.

```
.DIRECTORY DX0:VTMAC.MAC/BEGIN
 10-Dec-79
VTMAC .MAC     15  10-Aug-79       DIR    .SAV     17  03-Aug-79
RK    .SYS      3  13-Aug-79       EDIT   .SAV     19  03-Aug-79
STARTS.COM      1  27-Aug-79       DD     .SYS      5  19-Aug-79
SRCCOM.SAV     13  13-Aug-79       BINCOM.SAV     11  05-Oct-79
SLP   .SAV      9  13-Aug-79       SIPP   .SAV     14  05-Oct-79
 10 Files, 107 Blocks
 73 Free blocks
```

/BLOCKS This option prints a directory of the device you specify and includes the starting block number in decimal (or in octal if you use /OCTAL) of all the files listed. The following example lists the directory of DX0:, including the starting block numbers of files.

```
.DIRECTORY/BLOCKS DX0:
 14-Dec-79
FSM   .MAC     31P 19-Nov-79  2955   BATCH .MAC   102P 19-Nov-79  2986
ELCOPY.MAC      8P 19-Nov-79  3088   ELINIT.MAC    15P 19-Nov-79  3096
ELTASK.MAC     15P 19-Nov-79  3111   ERROUT.MAC    48P 19-Nov-79  3126
ERRTXT.MAC      9P 19-Nov-79  3174   SYCND .BL      3P 19-Nov-79  3183
SYSTBL.BL       4P 19-Nov-79  3186   SYCND .DIS     5P 19-Nov-79  3190
SYSTBL.DIS      4P 19-Nov-79  3195   SYCND .HD      5P 19-Nov-79  3199
ABSLOD.SAV     48  15-Mar-76  3204   CHESS .SAV    40  17-Aug-75  3252
PETAL .SAV     36  11-Sep-75  3292   LAMP  .SAV    29  16-Mar-79  3328
WUMPUS.SAV     30  16-Mar-79  3357
 17 Files, 348 Blocks
 138 Free blocks
```

/BRIEF This option lists only file names and file types, omitting file lengths and associated dates. It produces a five-column listing, as the following example shows.

```
.DIRECTORY/BRIEF RK1:
 14-Dec-79
SWAP  .SYS    RT11SJ.SYS    RT11FB.SYS    RT11BL.SYS    TT    .SYS
DT    .SYS    DP    .SYS    DX    .SYS    DY    .SYS    RF    .SYS
RK    .SYS    DL    .SYS    DM    .SYS    DS    .SYS    DD    .SYS
LP    .SYS    LS    .SYS    CR    .SYS    MS    .SYS    MTHD  .SYS
DISMT1.COM    MMHD  .SYS    NUMBER.PAS    WLOCK .SAV    MYPROG.MAC
PROG  .MAP    ANTONY.BAK    MSHD  .SYS    NL    .SYS    PC    .SYS
PD    .SYS    CT    .SYS    BA    .SYS    MYPROG.SAV    ODT   .SAV
 35 Files, 408 Blocks
 78 Free blocks
```

/COLUMNS:n Use this option to list a directory in a specific number of columns. The value *n* represents an integer in the range 1-9. Normally, the system uses two columns for regular listings and five columns for brief listings. The following example lists the directory information for device DX1: in one column.

DIRECTORY

```
.DIRECTORY/COLUMNS:1 DX1:
 29-Nov-79
SWAP   .SYS     25P 19-Nov-79
RT11SJ.SYS      67P 19-Nov-79
RT11FB.SYS      80P 19-Nov-79
RT11BL.SYS      64P 19-Nov-79
TT     .SYS      2P 19-Nov-79
DT     .SYS      3P 19-Nov-79
DP     .SYS      3P 19-Nov-79
 7 Files, 244 Blocks
 242 Free blocks
```

/DATE[date] Use this option to include in the directory listing only those files with a certain date. The following command lists all the files on device DX0: that were created on 13 August 1979.

```
.DIRECTORY/DATE:13:AUG:79 DX0:
 15-Sep-79
RT11SJ.SYS      67P 13-Aug-79       RT11FB.SYS      80P 13-Aug-79
RT11BL.SYS      63P 13-Aug-79       DX     .SYS      3P 13-Aug-79
SWAP   .SYS     25P 13-Aug-79       TT     .SYS      2P 13-Aug-79
DP     .SYS      3P 13-Aug-79       DY     .SYS      4P 13-Aug-79
LP     .SYS      2P 13-Aug-79       PIP    .SAV     16  13-Aug-79
DUP    .SAV     41  13-Aug-79       RESORC.SAV     15  13-Aug-79
DIR    .SAV     17  13-Aug-79       RK     .SYS      3  13-Aug-79
EDIT   .SAV     19  13-Aug-79       DD     .SYS      5  13-Aug-79
SRCCOM.SAV      13  13-Aug-79       BINCOM.SAV     11  13-Aug-79
SLP    .SAV      9  13-Aug-79       SIPP   .SAV     14  13-Aug-79
 20 Files, 412 Blocks
 73 Free blocks
```

/DELETED This option lists a directory of files that have been deleted from a specific device, but whose file name information has not been destroyed. The listing includes the file names, types, sizes, creation dates, and starting block numbers in decimal of the files. The file names that print also represent tentative files. The listing can be useful in recovering files that have been accidentally deleted. Once you identify the file name and location, you can use the CREATE command or DUP to rename the area (see Section 8.2.1 for this procedure). The following command lists files on device DX0: that have been deleted.

```
.DIRECTORY/DELETED DX0:
 14-Dec-79
SYSGEN.CND     11  19-Nov-79  1403  TS     .MAC      2  27-Nov-79  2895
TM     .MAC    26  19-Nov-79  2926  MT     .SYS     32  27-Nov-79  3415
468DAT.DIR      1  14-Dec-79  3701  468DEL.DIR     527  14-Dec-79  3704
NUM2   .MAC     4  21-Nov-79  4231  NUM2   .LST     565  06-Sep-79  4235
 0 Files, 0 Blocks
 1164 Free blocks
```

Note in the example shown above that, since a deleted file does not really exist, the total number of files and blocks is 0.

/DOS Use this option to list the directory of a device that is in RSTS/E or DOS format. The only other options valid with /DOS are /BRIEF, /FAST, and /OWNER. The valid devices are DECtape for RSTS/E and DOS, and RK05 for DOS.

/END:n Use with /START:n and /BADBLOCKS to specify the last block number of a bad block scan. If you do not specify /END:n, the system scans to the last block on the volume.

/EXCLUDE This option lists a directory of all the files on a device except those files you specify. The following example lists all files on DX0: except the .SAV and .SYS files.

```
.DIRECTORY/EXCLUDE DX0:(*.SAV,*.SYS)
 29-Oct-79
RT11SJ.MAC     67P 06-Sep-79       RT11FB.MAC     80P 06-Sep-79
RT11BL.MAC     63P 06-Sep-79       DX     .MAC     3P 06-Sep-79
SWAP   .MAC    25P 06-Sep-79       TT     .MAC     2P 06-Sep-79
DP     .MAC     3P 06-Sep-79       DY     .MAC     4P 06-Sep-79
LP     .MAC     2P 06-Sep-79       RK     .MAC     3  06-Sep-79
STARTS.COM      1  27-Aug-79       DD     .MAC     5  06-Sep-79
 12 Files, 258 Blocks
 73 Free blocks
```

/FAST This option lists only file names and file types, omitting file lengths and associated dates. This is the same as /BRIEF.

/FILES Use this option with /BADBLOCKS to print the file names of bad blocks. If the system does not find any bad blocks, it prints only the heading, as the following example shows. Do not use this option if the volume is not a standard RT–11 directory-structured volume or if the volume does not contain an RT–11 directory.

```
.DIRECTORY/BADBLOCKS/FILES DT1:
?DUP-I-No bad blocks detected DT1:
```

/FREE Use this option to print a directory of unused areas and the size of each. This example lists the unused areas on device DK:.

```
.DIRECTORY/FREE
 14-Dec-79
< UNUSED >     11                  < UNUSED >      2
< UNUSED >     26                  < UNUSED >     32
< UNUSED >      1                  < UNUSED >    525
< UNUSED >      0                  < UNUSED >    565
 0 Files, 0 Blocks
 1162 Free blocks
```

/FULL This option lists the entire directory, including unused areas and their sizes in blocks (decimal). The following example lists the entire directory for device DX0:.

```
.DIRECTORY/FULL DX0:
 14-Dec-79
SWAP   .SYS     25P 23-Oct-79       RT11SJ.SYS     67P 23-Oct-79
RT11FB.SYS      80P 19-Nov-79       RT11BL.SYS     64P 19-Nov-79
TT     .SYS      2P 19-Nov-79       DT     .SYS     3P 19-Nov-79
DP     .SYS      3P 23-Oct-79       DX     .SYS     3P 19-Nov-79
DY     .SYS      4P 19-Nov-79       RF     .SYS     3P 19-Nov-79
RK     .SYS      3P 19-Nov-79       DL     .SYS     4P 23-Oct-79
DM     .SYS      5P 23-Oct-79       DS     .SYS     3P 19-Nov-79
DD     .SYS      5P 23-Oct-79       LP     .SYS     2P 23-Oct-79
LS     .SYS      2P 19-Nov-79       CR     .SYS     3P 19-Nov-79
MS     .SYS      9P 27-Nov-79       MTHD   .SYS     3P 23-Oct-79
DISMT1.COM       9P 27-Nov-79       MMHD   .SYS     4P 19-Nov-79
NUMBER.PAS       1  11-Dec-79       TONY   .AGP    14 17-Aug-79
NUM3   .LST       1  13-Dec-79       < UNUSED >    565
 25 Files, 322 Blocks
 164 Free blocks
```

/INTERCHANGE Use this option to list the directory of a diskette that is in interchange format. The only other options valid with /INTERCHANGE are /BRIEF and /FAST.

/NEWFILES This option includes in the directory listing only those files that were created on the current day. This is a convenient way to list the files you created in one session at the computer. The following command lists the new files on 19 May 1979.

```
.DIRECTORY/NEWFILES DT0:
 19-May-79
FILE1 .TXT      1  19-May-79        FILE2 .TXT    1  19-May-79
 2 Files, 2 Blocks
 328 Free blocks
```

/OCTAL This option lists the sizes (and starting block numbers if you also use /BLOCKS) in octal. If the device you specify is a magtape or cassette, the system prints the sequence numbers in octal. The following example shows an octal listing of device DX0:.

```
.DIRECTORY/OCTAL DX0:
 14-Dec-79 Octal
MYPROG.MAC      44P 12-Nov-79       TM     .MAC     31  27-Nov-79
VTMAC .MAC       7  18-Oct-79       SYSMAC.MAC      51  19-Nov-79
SWAP  .SYS      31  05-Sep-79       ANTON .MAC       4  19-Nov-79
RT11SJ.SYS     103  19-Nov-79       TT     .SYS      2  19-Nov-79
DX     .SYS      3  29-Aug-79       BUILD .MAC     144  19-Nov-79
 10 Files,    462 Blocks
    264 Free blocks
```

/ORDER[:category] This option sorts the directory of a device according to the category you specify. Table 4–3 summarizes the categories and their functions.

Table 4–3: Sort Categories

Category	Function
DATE	Sorts the directory chronologically by creation date. Files that have the same date are sorted alphabetically by file name and file type.
NAME	Sorts the directory alphabetically by file name. Files that have the same file name are sorted alphabetically by file type (this has the same effect as the /ALPHABETIZE option).
POSITION	Lists the files according to their position on the device (this is the same as using /ORDER with no category).
SIZE	Sorts the directory based on file size in blocks. Files that are the same size are sorted alphabetically by file name and file type.
TYPE	Sorts the directory alphabetically by file type. Files that have the same file type are sorted alphabetically by file name.

The following examples list the directory of device DX0:, according to each of the categories.

```
.DIRECTORY/ORDER:DATE DX0:
 14-Dec-79
BUILD .MAC    100  06-Sep-79       SYSMAC.MAC     41  19-Nov-79
DX    .SYS      3  06-Sep-79       TT     .SYS     2  19-Nov-79
MYPROG.MAC     36P 12-Oct-79       VTMAC  .MAC     7  19-Nov-79
RFUNCT.MAC      4  19-Nov-79       TM     .MAC    25  27-Nov-79
RT11SJ.SYS     67  19-Nov-79       SWAP   .SYS    25  05-Dec-79
 10 Files, 306 Blocks
 180 Free blocks


.DIRECTORY/ORDER:NAME DX0:
 14-Dec-79
BUILD .MAC    100  06-Sep-79       SWAP   .SYS    25  05-Dec-79
DX    .SYS      3  06-Sep-79       SYSMAC.MAC     41  19-Nov-79
MYPROG.MAC     36P 12-Oct-79       TM     .MAC    25  27-Nov-79
RFUNCT.SYS      4  19-Nov-79       TT     .SYS     2  19-Nov-79
RT11SJ.SYS     67  19-Nov-79       VTMAC  .MAC     7  19-Nov-79
 10 Files, 306 Blocks
 180 Free blocks


.DIRECTORY/ORDER:POSITION DX0:
 14-Dec-79
RT11SJ.SYS     67  19-Nov-79       BUILD .MAC    100  06-Sep-79
DX    .SYS      3  06-Sep-79       SYSMAC.MAC     41  19-Nov-79
MYPROG.MAC     36P 12-Oct-79       TM     .MAC    25  27-Nov-79
SWAP   .SYS    25  05-Dec-79       VTMAC  .MAC     7  19-Nov-79
RFUNCT.SYS      4  19-Nov-79       TT     .SYS     2  19-Nov-79
 10 Files, 306 Blocks
 180 Free blocks
```

DIRECTORY

```
.DIRECTORY/ORDER:SIZE DX0:
 14-Dec-79
TT     .SYS      2  19-Nov-79       SWAP   .SYS     25  05-Dec-79
DX     .SYS      3  06-Sep-79       MYPROG.MAC     36P 12-Oct-79
RFUNCT.SYS       4  19-Nov-79       SYSMAC.MAC     41  19-Nov-79
VTMAC  .MAC      7  19-Nov-79       RT11SJ.SYS     67  19-Nov-79
TM     .MAC     25  27-Nov-79       BUILD  .MAC    100  06-Sep-79
 10 Files, 306 Blocks
 180 Free blocks


.DIRECTORY/ORDER:TYPE DX0:
 14-Dec-79
BUILD  .MAC    100  06-Sep-79       DX     .SYS      3  06-Sep-79
MYPROG.MAC      36P 12-Oct-79       RFUNCT.SYS       4  19-Nov-79
SYSMAC.MAC      41  19-Nov-79       RT11SJ.SYS      67  19-Nov-79
TM     .MAC     25  27-Nov-79       SWAP   .SYS     25  05-Dec-79
VTMAC  .MAC      7  19-Nov-79       TT     .SYS      2  19-Nov-79
 10 Files, 306 Blocks
 180 Free blocks
```

/OUTPUT:filespec Use this option to specify a device and file name for the output listing file. Normally, the directory listing appears on the console terminal. If you omit the file type for the listing file, the system uses .DIR.

/OWNER:[nnn,nnn] Use this option with /DOS to specify a user identification code (UIC). Note that the square brackets are part of the UIC; you must type them.

/POSITION Use this option to list the file sequence numbers of files stored on a magtape. See /COLUMNS:n for a sample listing.

/PRINTER Use this option to print the directory listing on the line printer. The default output device is the terminal. Note that the /PRINTER option does not use the QUEUE program to queue the directory listing.

/REVERSE This option lists a directory in the reverse order of the sort you specify with /ALPHABETIZE, /ORDER, or /SORT. The following example sorts the directory of DX0: and lists it in reverse order by size.

```
.DIRECTORY/ORDER:SIZE/REVERSE DX0:
 14-Dec-79
BUILD  .MAC    100  06-Sep-79       TM     .MAC     25  27-Nov-79
RT11SJ.SYS      67  19-Nov-79       VTMAC  .MAC      7  19-Nov-79
SYSMAC.MAC      41  19-Nov-79       RFUNCT.SYS       4  19-Nov-79
MYPROG.MAC      36P 12-Oct-79       DX     .SYS      3  06-Sep-79
SWAP   .SYS     25  05-Dec-79       TT     .SYS      2  19-Nov-79
 10 Files, 306 Blocks
 180 Free blocks
```

/SINCE[date] This option lists a directory of all files on a specified device that were created on or after a specified date. The following command lists only those files on DK: that were created on or after 13 August 1977.

```
.DIRECTORY/SINCE:13:AUG:79
 14-Dec-79
RT11SJ.SYS     67P 14-Aug-79       RT11FB.SYS     80P 02-Sep-79
RT11BL.SYS     63P 19-Aug-79       DX     .SYS     3P 10-Sep-79
SWAP   .SYS    25P 02-Sep-79       TT     .SYS     2P 15-Sep-79
SIPP   .SAV    14  02-Sep-79
 7 Files, 154 Blocks
 332 Free blocks
```

/SORT[:category] This option sorts the directory of a device according to the category you specify. It is the same as /ORDER[:category].

/START[:n] Use this option with the /BADBLOCKS option to specify the starting block, and optionally the last block if you use /END:n, of the bad block scan. The argument *n* represents a block number in decimal. If you do not supply a value with /START, the system scans from the first block on the volume. If you do not specify /END:n, the system scans to the end of the volume.

/SUMMARY This option lists a summary of the device directory. The summary lists the number of files in each segment and the number of segments in use on the volume you specify. The /SUMMARY option does not list the segments in numerical order, only the order in which they are linked on the volume. The following example lists the summary of the directory for device DK:.

```
.DIRECTORY/SUMMARY
 14-Nov-79

    44 Files in segment 1

    46 Files in segment 4

    37 Files in segment 2

    34 Files in segment 5

    38 Files in segment 3

    16 Available segments, 5 in use

199 Files, 3647 Blocks
1115 Free blocks
```

/TERMINAL This option lists directory information on the console terminal. This is the default operation.

/TOPS Use this option to list the directory of a DECtape that is in DECsystem–10 format. The only other options valid with /TOPS are /BRIEF and /FAST.

/VERIFY Use this option with the /BADBLOCKS option to read a bad block, write to it, and read it again. If the system can not read the block, it reports a hard error. If the block recovers, it reports a soft error. This procedure does not destroy data already on the volume.

Use this option only when necessary; DIGITAL does not guarantee the integrity of the data recovered from a soft bad block error.

/VOLUMEID[:ONLY] Use /VOLUMEID to print the volume ID and owner name along with the directory listing of the storage volume. If you include the optional argument, ONLY, the system prints only the volume ID and owner name.

The following example displays the volume ID of volume DK1:

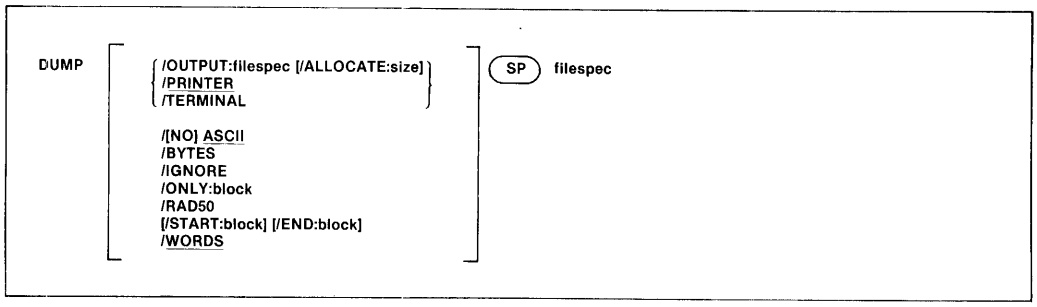
```
.DIRECTORY/VOLUMEID DX1:
 14-Dec-79
 Volume ID: BACKUP2
 Owner    : Marcy
SWAP   .SYS     25P 19-Nov-79       RT11SJ.SYS     67P 19-Nov-79
RT11FB.SYS     80P 19-Nov-79       RT11BL.SYS     64P 19-Nov-79
TT     .SYS      2P 19-Nov-79       DT     .SYS      3P 19-Nov-79
DP     .SYS      3P 19-Nov-79       DX     .SYS      3P 19-Nov-79
DY     .SYS      4P 19-Nov-79       RF     .SYS      3P 19-Nov-79
RK     .SYS      3P 19-Nov-79       DL     .SYS      4P 19-Nov-79
 12 Files, 271 Blocks
 215 Free blocks
```

/WAIT Use with the /BADBLOCKS option when you want the system to initiate a bad block scan but to pause for you to mount the input volume. This option is particularly useful if you have a single-disk system. When you use this option, and the system volume is mounted, the system initiates the operation you specify, then prints *Mount input volume in <device>; Continue?*. The prompt *<device>* represents the device into which you mount the volume. Mount your input volume and type Y, followed by a carriage return.

The following sample performs a bad block scan on an RK05 disk.

```
DIRECTORY/WAIT/BADBLOCKS RK0:
Mount input volume in RK0: Continue? Y
?DUP-I-No bad blocks detected RK0:
Mount system volume in RK0: Continue? Y
```

The DUMP command can print on the terminal or line printer, or write to a file all or any part of a file in octal words, octal bytes, ASCII characters, and/or Radix–50 characters. It is particularly useful for examining directories and files that contain binary data.



In the command syntax shown above, *filespec* represents the device or file you want to examine. If you do not specify an output file, the listing prints on the line printer. If you do not specify a file type for an output file, the system uses .DMP. You can specify the entire command on one line, or you can rely on the system to prompt you for information. The DUMP command prompt is *Device or file?*.

Notice that some of the options (/ONLY, /START, and /END) accept a block number as an argument. Remember that all block numbers are in octal, and that the first block of a device or file is block 0. To specify a decimal block number, follow the number with a decimal point. If you are dumping a file, the block numbers you specify are relative to the beginning of that file. If you are dumping a device, the block numbers are the absolute (physical) block numbers on that device.

The system handles operations involving magtape and cassette differently from operations involving random-access devices. If you dump an RT–11 file-structured tape and specify only a device name in the file specification, the system reads only as far as the logical end-of-tape. Logical end-of-tape is indicated by an end-of-file label (EOF1) followed by two tape marks. For non-file-structured tape, logical end-of-tape is indicated by two consecutive tape marks. If you dump a cassette and specify only the device name in the file specification, the results are unpredictable. For magtape dumps, tape mark messages appear in the output listing as the system encounters them on the tape.

NOTE

The DUMP operation does not print data from track 0 of diskettes.

The following sections describe the options you can use with the DUMP command. Following the options are some sample listings and an explanation of how to interpret them.

/ALLOCATE:size Use this option with /OUTPUT to reserve space on the device for the output listing file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of -1 is a special case that creates the largest file possible on the device.

/ASCII This option prints the ASCII equivalent of each octal word or byte that is dumped. A dot (.) represents characters that are not printable. This is the default operation.

/NOASCII Use this option to suppress the ASCII output, which appears in the right hand column of the listing (or below the bytes if you have specified /BYTES). This allows the listing to fit in 72 columns.

/BYTES Use this option to display information in octal bytes. The system does not display words unless you also use /WORDS.

/END:block Use this option to specify an ending block number for the dump. The system dumps the device or file you specify, beginning with block 0 (unless you use /START) and continuing until it dumps the block you specify with /END.

/FOREIGN Use this option to dump a magtape that is not RT–11 file-structured.

/IGNORE Use this option to ignore errors that occur during a dump operation. Use /IGNORE if an input or output error occurred when you tried to perform a normal dump operation.

/ONLY:block Use this option to dump only the block number you specify.

/OUTPUT:filespec Use this option to specify a device and file name for the output listing file. Normally, the listing appears on the line printer. If you omit the file type for the listing file, the system uses .DMP.

/PRINTER This option causes the output listing to appear on the line printer. This is the default operation.

/RAD50 This option prints the Radix–50 equivalent of each octal word that is dumped.

/START:block Use this option to specify a starting block number for the dump. The system dumps the device or file, beginning at the block number you specify with /START and continuing to the end of the device or file (unless you use /END).

/TERMINAL This option causes the output listing to appear on the console terminal. Normally, the listing appears on the line printer.

/WORDS This option displays information in octal words. This is the default operation.

The following command dumps block 1 of the file SYSMAC.MAC. The output listing, which shows octal bytes and their ASCII equivalent, is stored in file MACLIB.DMP. The PRINT command prints the contents of the file on the line printer.

```
.DUMP/OUTPUT:MACLIB/BYTES/ONLY:1 SYSMAC.MAC

.PRINT MACLIB.DMP


SY:SYSMAC.MAC
BLOCK NUMBER  00001
000/ 120 040 117 106 040 040 124 110 105 040 040 123 117 106 124 127
     P       O   F           T   H   E           S   O   F   T   W
020/ 101 122 105 040 040 111 123 040 040 110 105 122 105 102 131 015
     A   R   E           I   S           H   E   R   E   B   Y   .
040/ 012 073 040 124 122 101 116 123 106 105 122 122 105 104 056 015
     .   ;       T   R   A   N   S   F   E   R   R   E   D   .   .
060/ 012 073 015 012 073 040 124 110 105 040 111 116 106 117 122 115
     .   ;   .   .   ;       T   H   E       I   N   F   O   R   M
100/ 101 124 111 117 116 040 111 116 040 124 110 111 123 040 123 117
     A   T   I   O   N       I   N       T   H   I   S       S   O
120/ 106 124 127 101 122 105 040 111 123 040 123 125 102 112 105 103
     F   T   W   A   R   E       I   S       S   U   B   J   E   C
140/ 124 040 124 117 040 103 110 101 116 107 105 040 040 127 111 124
     T       T   O       C   H   A   N   G   E           W   I   I
160/ 110 117 125 124 040 040 116 117 124 111 103 105 015 012 073 040
     H   O   U   T           N   O   T   I   C   E   .   .   ;
200/ 101 116 104 040 040 123 110 117 125 114 104 040 040 116 117 124
     A   N   D           S   H   O   U   L   D           N   O   T
220/ 040 040 102 105 040 040 103 117 116 123 124 122 125 105 104 040
         B   E           C   O   N   S   T   R   O   E   D
240/ 040 101 123 040 040 101 040 103 117 115 115 111 124 115 105 116
         A   S           A       C   O   M   M   I   T   M   E   N
260/ 124 040 102 131 040 104 111 107 111 124 101 114 040 105 121 125
     T       B   Y       D   I   G   I   T   A   L       E   Q   U
300/ 111 120 115 105 116 124 015 012 073 040 103 117 122 120 117 122
     I   P   M   E   N   T   .   .   ;       C   O   R   P   O   R
320/ 101 124 111 117 116 056 015 012 073 015 012 073 040 104 111 107
     A   T   I   O   N   .   .   .   ;   .   .   ;       D   I   G
340/ 111 124 101 114 040 101 123 123 125 115 105 123 040 116 117 040
     I   T   A   L       A   S   S   U   M   E   S       N   O
360/ 122 105 123 120 117 116 123 111 102 111 114 111 124 131 040 106
     R   E   S   P   O   N   S   I   B   I   L   I   T   Y       F
400/ 117 122 040 124 110 105 040 125 123 105 040 117 122 040 040 122
     O   R       T   H   E       U   S   E       O   R           R
420/ 105 114 111 101 102 111 114 111 124 131 040 040 117 106 040 040
     E   L   I   A   B   I   L   I   T   Y           O   F
440/ 111 124 123 015 012 073 040 123 117 106 124 127 101 122 105 040
     I   T   S   .   .   ;       S   O   F   T   W   A   R   E
460/ 117 116 040 105 121 125 111 120 115 105 116 124 040 127 110 111
     O   N       E   Q   U   I   P   M   E   N   T       W   H   I
500/ 103 110 040 111 123 040 116 117 124 040 123 125 120 120 114 111
     C   H       I   S       N   O   T       S   U   P   P   L   I
520/ 105 104 040 102 131 040 104 111 107 111 124 101 114 056 015 012
     E   D       B   Y       D   I   G   I   T   A   L   .   .   .
540/ 014 056 115 101 103 122 117 040 056 056 126 061 056 056 015 012
     .   .   M   A   C   R   O       .   .   V   1   .   .   .   .
560/ 056 115 103 101 114 114 011 056 056 056 103 115 060 054 056 056
     .   M   C   A   L   L   .   .   .   .   C   M   0   ,   .   .
600/ 056 103 115 061 054 056 056 056 103 115 062 054 056 056 056 103
     .   C   M   1   ,   .   .   .   C   M   2   ,   .   .   .   C
620/ 115 063 054 056 056 056 103 115 064 054 056 056 056 103 115 065
     M   3   ,   .   .   .   C   M   4   ,   .   .   .   C   M   5
640/ 054 056 056 056 103 115 066 015 012 056 056 056 126 061 075 061
     ,   .   .   .   C   M   6   .   .   .   .   .   V   1   =   1
660/ 015 012 056 105 116 104 115 015 012 015 012 056 115 101 103 122
     .   .   .   E   N   D   M   .   .   .   .   .   M   A   C   R
700/ 117 040 056 056 126 062 056 056 015 012 056 115 103 101 114 114
     O       .   .   V   2   .   .   .   .   .   M   C   A   L   L
720/ 011 056 056 056 103 115 060 054 056 056 056 103 115 061 054 056
     .   .   .   .   C   M   0   ,   .   .   .   C   M   1   ,   .
740/ 056 056 103 115 062 054 056 056 056 103 115 063 054 056 056 056
     .   .   C   M   2   ,   .   .   .   C   M   3   ,   .   .   .
760/ 103 115 064 054 056 056 056 103 115 065 054 056 056 056 103 115
     C   M   4   ,   .   .   .   C   M   5   ,   .   .   .   C   M
```

In the printout above, the heading shows which file was dumped and which block of the file follows. The numbers in the leftmost column indicate the byte offset from the beginning of the block. Remember that these are all octal values, and that there are two bytes per word. The octal bytes that were dumped appear in the next eight columns. The ASCII equivalent of each octal byte appears underneath the byte. The system substitutes a dot (.) for nonprinting codes, such as those for control characters.

The last example shows block 6 (the directory) of device RK0:. The output is in octal words with Radix–50 equivalents below each word.

```
.DUMP/NOASCII/RAD50/ONLY:6 RK0:


RK0:/N/A/O:6
BLOCK NUMBER  00006
000/ 000020 000002 000004 000000 000046 002000 075131 062000
       P      B      D               &      YX     SWA     P
020/ 075273 000031 000000 027147 002000 071677 142302 075273
     SYS      Y             GP9     YX     RT1    1SJ    SYS
040/ 000103 000000 027147 002000 071677 141262 075273 000120
       A$            GP9     YX     RT1    1FB    SYS      B
060/ 000000 027147 002000 071677 141034 075273 000100 000000
            GP9     YX     RT1    1BL    SYS     AX
100/ 027147 002000 100040 000000 075273 000002 000000 027147
     GP9     YX     TT             SYS      B             GP9
120/ 002000 016040 000000 075273 000003 000000 027147 002000
       YX    DT             SYS      C             GP9     YX
140/ 015600 000000 075273 000003 000000 027147 002000 016300
     DP             SYS      C             GP9     YX    DX
160/ 000000 075273 000003 000000 027147 002000 016350 000000
            SYS      C             GP9     YX    DY
200/ 075273 000004 000000 027147 002000 070560 000000 075273
     SYS      D             GP9     YX     RF             SYS
220/ 000003 000000 027147 002000 071070 000000 075273 000003
       C            GP9     YX     RK             SYS      C
240/ 000000 027147 002000 015340 000000 075273 000004 000000
            GP9     YX     DL             SYS      D
260/ 027147 002000 015410 000000 075273 000005 000000 027147
     GP9     YX     DM             SYS      E             GP9
300/ 002000 015770 000000 075273 000003 000000 027147 002000
       YX    DS             SYS      C             GP9     YX
320/ 014640 000000 075273 000005 000000 027147 002000 046600
     DD             SYS      E             GP9     YX    LP
340/ 000000 075273 000002 000000 027147 002000 046770 000000
            SYS      B             GP9     YX    LS
360/ 075273 000002 000000 027147 002000 012620 000000 075273
     SYS      B             GP9     YX     CR             SYS
400/ 000003 000000 027147 002000 052070 000000 075273 000011
       C            GP9     YX     MS             SYS      I
420/ 000000 027547 002000 052150 014400 075273 000003 000000
            GW0     YX     MTH    D      SYS      C
440/ 027147 002000 015173 052177 012445 000011 000000 027547
     GP9     YX     DIS    MT1    COM      I             GW0
460/ 002000 051520 014400 075273 000004 000000 027147 002000
       YX    MMH    D      SYS      D             GP9     YX
500/ 015173 052200 012445 000010 000000 027547 002000 052100
     DIS    MT2    COM      H             GW0     YX    MSH
520/ 014400 075273 000004 000000 027147 002000 054540 000000
     D      SYS      D             GP9     YX    ML
540/ 075273 000002 000000 027147 002000 062170 000000 075273
     SYS      B             GP9     YX     PC             SYS
560/ 000002 000000 027147 002000 062240 000000 075273 000003
       B            GP9     YX     PD             SYS      C
600/ 000000 027147 002000 012740 000000 075273 000005 000000
            GP9     YX     CT             SYS      E
620/ 027147 002000 006250 000000 075273 000007 000000 027147
     GP9     YX     BA             SYS      G             GP9
```

```
640/ 002000 016130 000000 073376 000051 000000 027147 002000
       YX     DUP               SAV      AA             GP9      YX
660/ 023752 050574 073376 000023 000000 027147 002000 070533
     FOR     MAT     SAV       S             GP9      YX     RES
700/ 060223 073376 000017 000000 027147 002000 015172 000000
     ORC     SAV       O             GP9      YX     DIR
720/ 073376 000021 000000 027147 002000 075273 050553 074324
     SAV       Q             GP9      YX     SYS     MAC     SML
740/ 000052 000000 027147 002000 017751 076400 073376 000023
       AB            GP9      YX     EDI     T       SAV       S
760/ 000000 027147 002000 042614 000000 073376 000073 000000
            GP9      YX     KED             SAV      AS
```

E

The E (Examine) command prints in octal the contents of an address on the console terminal.

E (SP) address [-address]

In the command syntax illustrated above, *address* represents an octal address that, when added to the relocation base value from the Base command, provides the actual address that the system examines. This command permits you to open specific locations in memory and inspect their contents. It is most frequently used after a GET command to examine locations in a program.

The Examine command accepts both word and byte addresses, but it always executes the command as though you specified a word address. (If you specify an odd address, the system decreases it by one.)

If you specify more than one address (in the form *address1-address2)*, the sys- tem prints the contents of *address1* through *address2*, inclusive. The second address (*address2*) must always be greater than the first address. If you do not specify an address, the system prints the contents of relative location 0.

Note that you cannot examine addresses outside the background.

The following example prints the contents of location 1000, assuming the relocation base is 0.

```
.E 1000

127401
```

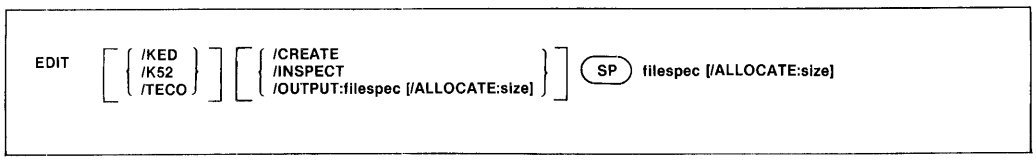
The next command sets the relocation base to 1000.

```
.B 1000
```

The following command prints the contents of locations 2000 (offset of 1000 from last B command) through 2005.

```
.E 1001-1005

127401 007624 127400
```

The EDIT command invokes the text editor.



The text editor, EDIT, is a program that creates or modifies ASCII files for use as input to programs such as the MACRO assembler or the FORTRAN compiler. The editor reads ASCII files from any input device, makes specified changes, and writes the file on an output device. It also allows efficient use of VT11 or VS60 display hardware, if this is part of the system configuration (except in multi-terminal systems).

You can also use the Keypad Editor (KED for VT100 terminals, K52 for VT52 terminals) as an alternative to EDIT. The Keypad Editor is restricted to the VT100 and VT52 terminals, however. You can invoke the Keypad Editor with the /KED or /K52 options described below. For more information on the Keypad Editor, see the *PDP–11 Keypad Editor User's Guide.*

NOTE

You can use the SET EDIT command to set a default editor (EDIT, KED, or K52) so that when you issue the EDIT command, you invoke that editor. The system defaults to the EDIT editor each time you bootstrap, however. For more details, see the SET EDIT command description.

EDIT considers a file to be divided into logical units called pages. A page of text is generally 50–60 lines long (delimited by form feed characters) and corresponds approximately to a physical page of a program listing. EDIT reads one page of text at a time from the input file into its internal buffers where the page becomes available for editing. You can then use editing commands to:

- Locate text to be changed
- Execute and verify the changes
- List an edited page on the console terminal
- Output a page of text to the output file

In the command syntax illustrated above, *filespec* represents the file you wish to edit. You can enter the EDIT command on one line, or you can rely on the system to prompt you for information. If you do not supply a file specification for the file to edit, the system prompts *File?*. If you do not specify any option with the EDIT command, the text editor performs the edit backup operation. To do this, it changes the name of the original file, giving it a file

type of .BAK when you finish making your editing changes. The actual file renaming occurs when you successfully exit from the editor.

When you want to edit an existing file, the editor does not perform any I/O operation as a result of your command. You must issue the R command to the editor to read the first page of text and make it available for you to work on. The following example invokes EDIT, opens an existing file, and reads the first page of text:

```
.EDIT MYFILE.TXT
*R$$
```

When you issue an EDIT command, the system invokes the text editor. (You can use the SET EDIT command to set the default editor. If you do not use the SET EDIT command, the system assumes EDIT.SAV each time you issue the EDIT command. See the SET EDIT command for more information.) It is possible to receive an error or warning message as a result of the EDIT command. If, for example, the file you need to edit with EDIT does not exist on device DK:, the editor issues an error message and remains in control. For example:

```
.EDIT/INSPECT EXAMP3.TXT
?EDIT-F-File not found
*^C$$
```

When a situation like this occurs, you can either issue another command directly to the text editor or enter CTRL/C followed by two ESCAPEs to return control to the monitor.

NOTE

To perform any edit operations on a protected file, you must disable the file's protected status (see the RENAME command description).

The following sections describe the options you can use with the EDIT command. A complete description of EDIT is contained in Chapter 5.

/ALLOCATE:size Use this option with /OUTPUT or after the file specification to reserve space on the device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of -1 is a special case that creates the largest file possible on the device.

/CREATE Use this option to build a new file. With EDIT you can also create a new file while you are working with the text editor by using the EDIT Write (EW) command, described in Chapter 5. The following example creates a file called NEWFIL.TXT on device DK:, inserts one line of text, and then closes the file.

```
.EDIT/CREATE NEWFIL.TXT
*ITHIS IS A NEW FILE.
$$
*EX$$
```

To create a file with /KED or /K52, see the *PDP–11 Keypad Editor User's Guide.*

/EXECUTE:filespec Use this option with /TECO to execute the TECO commands contained in the file you specify with /EXECUTE.

/INSPECT Use this option to open a file for reading. This option does not create any new output files. You can also open a file for inspection while you are working with the EDIT by using the Edit Read (ER) command, which is explained in Chapter 5.

The following command opens an existing file for inspection, lists its contents, and then exits.

```
.EDIT/INSPECT NEWFIL.TXT
*R$$
*/L$$
THIS IS A NEW FILE.
*^C$$
```

/KED This option invokes the Keypad Editor (KED). For more information on the Keypad Editor, see the *PDP–11 Keypad Editor User's Guide.* Use /KED only if you are using a VT100 terminal.

/K52 This option invokes the Keypad Editor. Use /K52 only if you are using a VT52 terminal. For more information on the Keypad Editor, see the *PDP–11 Keypad Editor User's Guide.*

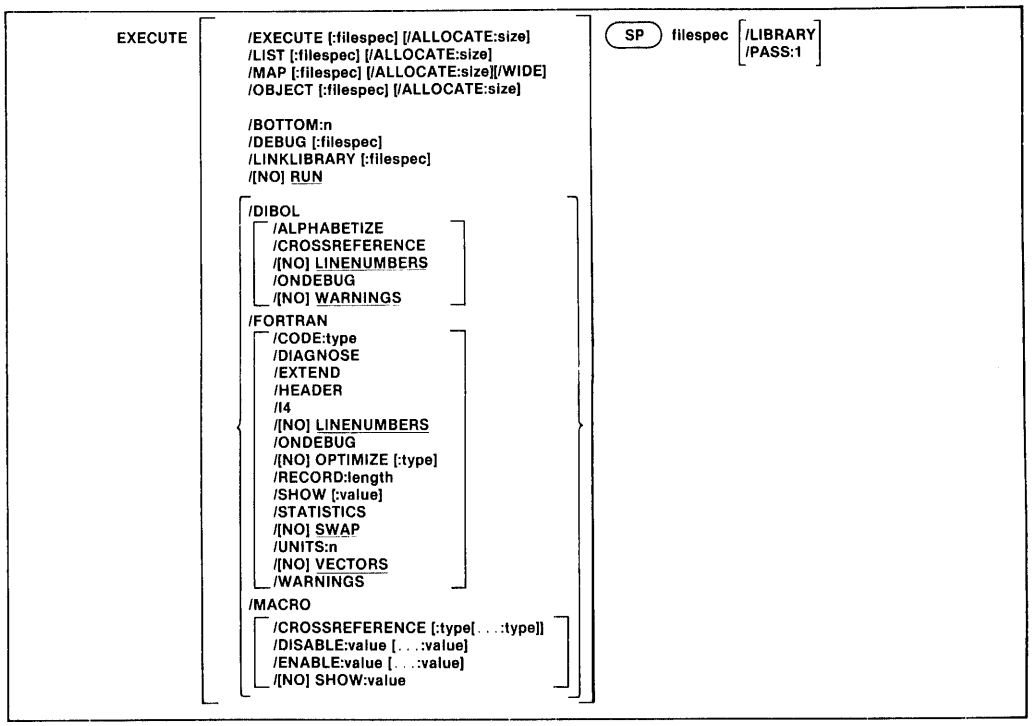
/OUTPUT:filespec This option directs the text you edit to the file you specify, leaving the input file unchanged. You can also write text to an output file while you are working with EDIT by using the Edit Write (EW) command, explained in Chapter 5. The following command reads file ORIG.TXT, and writes the edited text to file CHANGE.TXT.

```
.EDIT/OUTPUT:CHANGE.TXT ORIG.TXT
*
```

/TECO This option invokes the TECO editor. (TECO is not supported by DIGITAL. It is distributed on the RT–11 kit for the convenience of those customers who normally order TECO from the DECUS Program Library) For more information on TECO see the *PDP–11 TECO User's Guide.*

EXECUTE

The EXECUTE command invokes one or more language processors to assemble or compile the files you specify. It also links object modules and initiates execution of the resultant program.



In the command line shown above, *filespecs* represents one or more files to be included in the assembly. The default file types for the output files are .LST for listing files, .MAP for load map files, .OBJ for object files, and .SAV for memory image files. The defaults for input files depend on the language processor involved. These defaults include .MAC for MACRO files, .FOR for FORTRAN files, and .DBL for DIBOL files.

To compile (or assemble) multiple source files into a single object file, separate the files by plus (+) signs in the command line. Unless you specify otherwise, the system creates an object file with the same name as the first input file and gives it an .OBJ file type.

To compile multiple files in independent compilations, separate the files by commas (,) in the command line. This generates a corresponding object file for each set of input files. The system then links together all the object files and creates a single executable file. You can combine up to six files for a compilation producing a single object file. You can specify the entire EXECUTE command as one line, or you can rely on the system to prompt you for information. The EXECUTE command prompt is *Files?*.

There are several ways to establish which language processor the EXECUTE command invokes:

1. Specify a language-name option, such as /MACRO, to invoke the MACRO assembler.

2. Omit the language-name option and explicitly specify the file type for the source files. The EXECUTE command then invokes the language processor that corresponds to that file type. Specifying the file SOURCE.MAC, for example, invokes the MACRO assembler.

3. Let the system choose a file type of .MAC, .DBL, or .FOR for the source file you name. The handler for the device you specify must be loaded. If you specify DX1:A, and the DX handler is loaded, the system searches for source files A.MAC and A.DBL, in that order. If it finds one of these files, the system invokes the corresponding language processor. If it cannot find one of these files, or if the device handler associated with the input file is not resident, the system assumes a file type of .FOR and invokes the FORTRAN compiler.

   If the language processor selected as a result of this procedure described above is not on the system device (SY:), the system issues an error message.

Language options are position-dependent. That is, they have different meanings depending on where you place them in the command line. Options that qualify a command name apply across the entire command string. Options that follow a file specification apply only to the file (or group of files separated by plus signs) that they follow in the command string.

The following sections describe the options you can use with the EXECUTE command.

/ALLOCATE:size Use this option with /EXECUTE, /LIST, /MAP, or /OBJECT to reserve space on the device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of -1 is a special case that creates the largest file possible on the device.

/ALPHABETIZE Use this option with /DIBOL to alphabetize the entries in the symbol table listing. This is useful for program maintenance and debugging.

/BOTTOM:n Use this option to specify the lowest address to be used by the relocatable code in the load module. The argument *n* represents a six-digit, unsigned, even octal number. If you do not use this option, the system positions the load module so that the lowest address is location 1000 (octal). This option is invalid for foreground links.

/CODE:type Use this option with /FORTRAN to produce object code that is designed for a particular hardware configuration. The argument *type* represents a three-letter abbreviation for the type of code to be produced. The valid values are: EAE, EIS, FIS, and THR. See the *RT–11/RSTS/E FOR-*

TRAN IV User's Guide for a complete description of the types of code and their function.

/CROSSREFERENCE[:type[...:type]] Use this option with /MACRO or /DIBOL to generate a symbol cross-reference section in the listing. This information is useful for program maintenance and debugging. Note that the system does not generate a listing by default. You must also specify /LIST in the command line to get a cross-reference listing.

With MACRO this option takes an optional argument. The argument *type* represents a one-character code that indicates which sections of the cross-reference listing the assembler should include. Table 4–10 summarizes the valid arguments and their meaning.

/DEBUG[:filespec] Use this option to link ODT (on-line debugging technique, described in Chapter 21) with your program to help you debug it. If you supply the name of another debugging program, the system links the debugger you specify with your program. The debugger is always linked low in memory relative to your program.

/DIAGNOSE Use this option with /FORTRAN to help analyze an internal compiler error. /DIAGNOSE expands the crash dump information to include internal compiler tables and buffers. Submit the diagnostic printout to DIGITAL with a software performance report (SPR) form. The information in the listing can help DIGITAL programmers locate the compiler error and correct it.

/DIBOL This option invokes the DIBOL language processor to compile the associated files.

/DISABLE:value[...:value] Use this option with /MACRO to specify a .DSABL directive. Table 4–11 summarizes the arguments and their meaning. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all legal values.

/ENABLE:value[...:value] Use this option with /MACRO to specify an .ENABL directive. Table 4–11 summarizes the arguments and their meaning. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all legal values.

/EXECUTE[:filespec] Use this option to specify a file name or device for the executable file. Because the EXECUTE command creates executable files by default, the following two commands have the same meaning:

```
.EXECUTE  MYPROG

.EXECUTE/EXECUTE  MYPROG
```

Both commands link MYPROG.OBJ and produce MYPROG.SAV as a result. The /EXECUTE option has different meanings when it follows the command and when it follows the file specification. The following command creates an executable file called PROG1.SAV on device RK1:.

```
.EXECUTE/EXECUTE:RK1:  PROG1,PROG2
```

The next command creates an executable file called MYPROG.SAV on device DK:.

```
*EXECUTE RTN1,RTN2,MYPROG/EXECUTE
```

/EXTEND Use this option with /FORTRAN to change the right margin for source input lines from column 72 to column 80.

/FORTRAN This option invokes the FORTRAN language processor to compile the associated files.

/HEADER Use this option with /FORTRAN to include in the printout a list of options currently in effect.

/I4 Use this option with /FORTRAN to allocate two words for the default integer data type (FORTRAN uses only one-word integers) so that it takes the same physical space as real variables.

/LIBRARY Use this option with /MACRO to identify the file the option qualifies as a macro library file. Use it only after a library file specification in the command line. The MACRO assembler looks first to the library associated with the most recent /LIBRARY option to satisfy references (made with the .MCALL directive) from MACRO programs. It then looks to any libraries you specified earlier in the command line, and it looks last to SYSMAC.SML.

In the example below, the two files A.FOR and B.FOR are compiled together, producing B.OBJ and B.LST. The MACRO assembler assembles C.MAC, satisfying .MCALL references from MYLIB.MAC and SYSMAC.SML. It produces C.OBJ and C.LST. The system then links B.OBJ and C.OBJ together, resolving undefined references from SYSLIB.OBJ, and produces the executable file B.SAV. Finally, the system loads and executes B.SAV.

```
*EXECUTE A+B/LIST/OBJECT,MYLIB/LIBRARY+C,MAC/LIST/OBJECT
```

/LINENUMBERS Use this option with /DIBOL or /FORTRAN to include internal sequence numbers in the executable program. These are especially useful in debugging programs. This is the default operation.

/NOLINENUMBERS Use this option with /DIBOL or /FORTRAN to suppress the generation of internal sequence numbers in the executable program. This produces a smaller program and optimizes execution speed. Use this option to compile only those programs that are already debugged; otherwise the line numbers in DIBOL or FORTRAN error messages are difficult to interpret.

/LINKLIBRARY:filespec Use this option to include the library file name you specify as an object module library during the linking operation. Repeat the option if you need to specify more than one library file.

/LIST[:filespec] You must specify this option to produce a compilation or assembly listing. The /LIST option has different meanings depending on where you put it in the command line.

If you specify /LIST without *filespec* in the list of options that immediately follows the command name, the system generates a listing that prints on the line printer. If you follow /LIST with a device name, the system creates a listing file on that device. If the device is a file-structured device, the system stores the listing file on that device, assigning it the same name as the input file with a .LST file type. The following command produces a listing on the terminal:

```
.EXECUTE/LIST:TT A.FOR
```

The next command creates a listing file called A.LST on RK3:.

```
.EXECUTE/LIST:RK3: A.MAC
```

If the /LIST option contains a name and file type to override the default of .LST, the system generates a listing file with that name. The following command, for example, compiles A.FOR and B.FOR together, producing files A.OBJ and FILE1.OUT on device DK:. It then links A.OBJ (using SYSLIB.OBJ as needed) and produces A.SAV.

```
.EXECUTE/NORUN/FORTRAN/LIST:FILE1.OUT A+B
```

Another way to specify /LIST is to type it after the file specification to which it applies. To produce a listing file with the same name as a particular input file, you can use a command similar to this one:

```
.EXECUTE/DIBOL A+B/LIST:RK3:
```

The command shown above compiles A.DBL and B.DBL together, producing files DK:A.OBJ and RK3:B.LST. It then links A.OBJ (using SYSLIB.OBJ as needed) and produces DK:A.SAV. If you specify a file name on a /LIST option following a file specification in the command line, it has the same meaning as when it follows the command. The following two commands have the same results:

```
.EXECUTE/MACRO A/LIST:B

.EXECUTE/MACRO/LIST:B A
```

Remember that file options apply only to the file (or group of files that are separated by plus signs) that they follow in the command string. For example:

```
.EXECUTE/NORUN A.MAC/LIST,B.FOR
```

This command compiles A.MAC, producing A.OBJ and A.LST. It also compiles B.FOR, producing B.OBJ. However, it does not produce any listing file for the compilation of B.FOR. Finally, the system links A.OBJ and B.OBJ together, producing A.SAV.

/MACRO This option invokes the MACRO assembler to assemble associated files.

/MAP[:filespec] You must specify this option to produce a load map after a link operation. The /MAP option has different meanings depending on

where you put it in the command line. It follows the same general rules outlined above for /LIST.

/OBJECT[;filespec] Use this option to specify a file name or device for the object file. Because the EXECUTE command creates object files by default, the following two commands have the same meaning:

```
.EXECUTE/FORTRAN A

.EXECUTE/FORTRAN/OBJECT A
```

Both commands compile A.FOR and produce A.OBJ as output. The /OBJECT option functions like the /LIST option; it can be either a command or a file qualifier.

As a command option, /OBJECT applies across the entire command string. The following command, for example, assembles A.MAC and B.MAC separately, creating object files A.OBJ and B.OBJ on RK1:.

```
.EXECUTE/OBJECT:RK1: A.MAC,B.MAC
```

Use /OBJECT as a file option to create an object file with a specific name or destination. The following command compiles A.DBL and B.DBL together, creating files B.LST, B.OBJ, and B.SAV.

```
.EXECUTE/DIBOL A+B/LIST/OBJECT/EXECUTE
```

/ONDEBUG Use this option with /DIBOL to include a symbol table in the object file. You can then use a debugging program to find and correct errors in the object file.

Use /ONDEBUG with /FORTRAN to include debug lines (those that have a D in column one) in the compilation. You do not, therefore, have to edit the file to include these lines in the compilation or to logically remove them. You can include messages, flags, and conditional branches to help you trace program execution and find an error.

/OPTIMIZE:type Use this option with /FORTRAN to enable certain options that optimize object code for various conditions. The argument *type* represents the three-letter code for the type of optimization to be enabled. Table 4–4 summarizes the codes and their meaning. This option is not available with version 2.5 of the FORTRAN compiler.

/NOOPTIMIZE:type Use this option with /FORTRAN to disable certain options that optimize object code for various conditions. The argument *type* represents the three-letter code for the type of optimization to be disabled. Table 4–4 summarizes the codes and their meaning. This option is not available with version 2.5 of the FORTRAN compiler.

/PASS:1 Use this option with /MACRO on a prefix macro file to process that file only during pass 1 of the assembly. This option is useful when you assemble a source program together with a prefix file that contains only macro definitions, since these do not need to be redefined in pass 2 of the

assembly. The following command assembles a prefix file and a source file together, producing files PROG1.OBJ, PROG1.LST, and PROG1.SAV.

```
.EXECUTE/NORUN/MACRO PREFIX/PASS:1+PROG1/LIST/OBJECT/EXECUTE
```

/RECORD:length Use this option with /FORTRAN to override the default record length of 132 characters for ASCII sequential formatted input and output. The meaningful range for length is from 4 to 4095.

/RUN Use this option to initiate execution of your program if there are no errors in the compilation or the link. This is the default operation. Do not use /RUN with any option that requires a response from the terminal.

/NORUN Use this option to suppress execution of your program. The system performs only the compilation and the link.

/SHOW[:value] Use this option with /FORTRAN to control FORTRAN listing format. The argument *value* represents a code that indicates which listings the compiler is to produce. Table 4–5 summarizes the codes and their meaning.

Use this option with /MACRO to specify any MACRO .LIST directive. Table 4–12 summarizes the valid arguments and their meaning. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives.

/NOSHOW:value Use this option with /MACRO to specify any MACRO .NLIST directive. Table 4–12 summarizes the valid arguments and their meaning. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives.

/STATISTICS Use this option with /FORTRAN to include compilation statistics, such as amount of memory used, amount of time elapsed, and length of the symbol table.

/SWAP Use this option with /FORTRAN to permit the USR (user service routine) to swap over the FORTRAN program in memory. This is the default operation.

/NOSWAP Use this option with FORTRAN to keep the USR resident during execution of a FORTRAN program. This may be necessary if the FORTRAN program uses some of the RT–11 System subroutine library calls (see the *RT–11 Programmer's Reference Manual*). If the program frequently updates or creates a large number of different files, making the USR resident can improve program execution. However, the cost for making the USR resident is 2K words of memory.

/UNITS:n Use this option with /FORTRAN to override the default number of logical units (6) to be open at one time. The maximum value you can specify for *n* is 16.

/VECTORS This option directs FORTRAN to use tables to access multidimensional arrays. This is the default mode of operation.

/NOVECTORS This option directs FORTRAN to use multiplication operations to access multidimensional arrays.

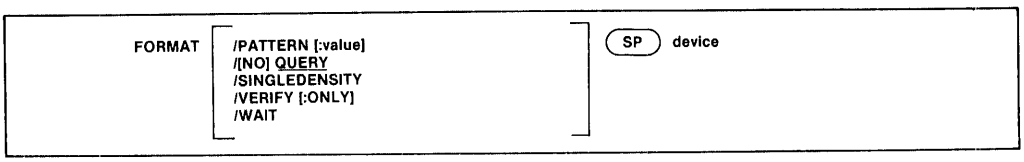
/WARNINGS Use this option to include warning messages in DIBOL or FORTRAN compiler diagnostic error messages. These messages call certain conditions to your attention but do not interfere with the compilation. This is the default operation for DIBOL.

/NOWARNINGS Use this option with /DIBOL to suppress warning messages during compilation. These messages are for your information only; they do not affect the compilation. This is the default operation for FORTRAN.

/WIDE Use this option with /MAP to produce a wide load map listing. Normally, the listing is wide enough for three Global Value columns, which is suitable for a page with 72 or 80 columns. The /WIDE option produces a listing that is six Global Value columns wide, or 132 columns.

FORMAT

The FORMAT command formats disks and diskettes, and verifies any disk, diskette, DECtape, or DECtape II.



In the command syntax described above, *device* represents the storage volume you wish to format and/or verify. Although you can verify any disk or DECtape, the formatting process is valid only for the disks and diskettes listed below.

RX01-RX02
RK05
RP02-RP03
RK06-RK07

When the system formats a volume, it writes headers for each block in the volume. The header of a block contains data the device controller must use to transfer data to and from that block. Using the FORMAT command to format a storage volume makes that volume usable to the RT–11 operating system. Formatting is advisable under the following circumstances:

- when you receive a new RK05 disk from DIGITAL
- when you wish to format an RX02 double density diskette to single density, and vice versa
- when you wish to eliminate bad blocks (though formatting does not guarantee the elimination of every bad block, formatting can reduce the number of bad blocks)

When the system verifies a volume, it writes a 16-bit pattern on each block in the volume, and then reads each pattern. When the system is unable to write and read a pattern, it reports a bad block. The verification process is similar to the bad block scan (see INITIALIZE), except that verification is a data-destructive process. That is, whereas bad block scanning only reads data from each block on a volume, verifying both writes and reads data, destroying any data previously existing on the volume. Because the verification process reads and writes data, it can be more effective than a bad block scan in establishing the validity of data contained in a block. Verifying also makes sure that the previous formatting operation was successful.

NOTE

You can format a diskette (RX01 or RX02) only when you have mounted the diskette in a double density diskette

drive unit (DY). Unless you use the /SINGLEDENSITY option, the system will format both single and double density diskettes in double density format. If you attempt to format a diskette in a single density drive unit (DX), the system will print an error message.

When you format an RK06 or RK07 disk, the system lists the block numbers of all the bad blocks in the manufacturer's bad block table and in the software bad block table.

The options you can use with the FORMAT command follow.

/PATTERN[:value] Use this option with /VERIFY[:ONLY] to specify which 16-bit patterns you want the system to use when it verifies the volume. The optional argument *value* represents an octal integer in the range 0 to 377 that denotes which of eight patterns you want used. The /P:n option in Chapter 18 provides a complete description of the patterns you can specify with /PATTERN[:value]. As the system uses each pattern, it prints at the terminal which pattern it is using.

The command line that follows verifies an RK05 disk with the 16-bit patterns denoted by the value 25.

```
.FORMAT/VERIFY/PATTERN:25  RK0:
RK0:/FORMAT-Are you sure? Y
?FORMAT-I-Formatting complete
PATTERN #5
PATTERN #3
PATTERN #1
?FORMAT-I-Verification complete
```

If you do not supply a value with /PATTERN, the system uses pattern #8.

/QUERY Use this option when you want the system to print a confirmation message before it performs formatting or verification. This is the default setting.

/NOQUERY Use this option if you do not want the system to print a confirmation message before it performs formatting or verification. When you use this option in the FORMAT command line, the system prints only the pattern numbers it uses if it performs verification and the informational messages indicating the formatting or verification is complete. The default setting is /QUERY.

/SINGLEDENSITY Use this option to format an RX02 double density diskette in single density format. The following example uses the /SINGLEDENSITY option to format an RX02 in drive unit 1 as a single density diskette.

```
.FORMAT/SINGLEDENSITY DY1:
DY1:/FORMAT-Are you sure? Y
?FORMAT-I-Formatting complete
```

/VERIFY[:ONLY] Use this option when you want to verify a volume following formatting. Use the optional argument, :ONLY, when you want the

system to only verify a volume. (Note that although you can format only a limited variety of storage volumes, you can verify any disk, diskette, DECtape, or DECtape II.) When you use /VERIFY, the system first formats the specified volume, and then writes a bit pattern to each block on the volume. Next, the system reads each pattern. After the verification process is complete, the system prints at the terminal the block number of each bad block it found.

The example that follows uses /VERIFY to format and verify an RK05 disk in drive unit 2.

```
.FORMAT/VERIFY RK2:
RK2:/FORMAT-Are you sure? Y
?FORMAT-I-Formatting complete
PATTERN #8
?FORMAT-I-Verification complete
```

The next example uses /VERIFY:ONLY to only verify an RX01 diskette in drive unit 0.

```
.FORMAT/VERIFY:ONLY DX0:
DX0:/VERIFY-Are you sure? Y
PATTERN #8
?FORMAT-I-Verification complete
```

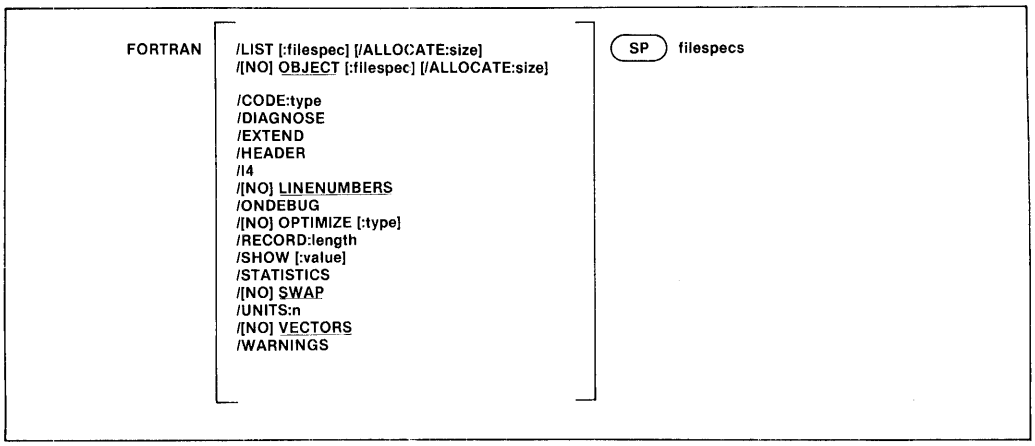
/WAIT Use this option to pause before formatting begins in order to substitute a second volume for the volume you specify in the command line. The /WAIT option is useful for single drive systems. After the system accepts your command line, it pauses while you exchange volumes. Type a Y followed by a carriage return when you have exchanged volumes and are ready for formatting to begin. When formatting completes, the system pauses again while you replace the system volume. Type a Y followed by a carriage return after you have remounted the system volume to terminate the formatting operation.

The following example uses the /WAIT option to format an RK05 disk.

```
.FORMAT/WAIT RK0:
RK0:/FORMAT-Are you sure? Y
Insert volume you wish to format.   CONTINUE(Y/N)?Y
/FORMAT-I-Formatting complete
replace original volume.            CONTINUE(Y/N)?Y
```

FORTRAN

The FORTRAN command invokes the FORTRAN IV compiler to compile one or more source programs.



You can enter the FORTRAN command as one line, or you can rely on the system to prompt you for information. The FORTRAN command prompt is *Files?* for the input specification.

In the command syntax illustrated above, *filespecs* represents one or more files to be included in the compilation. If you omit a file type for an input file, the system assumes .FOR. Output default file types are .LST for listing files and .OBJ for object files. To compile multiple source files into a single object file, separate the files with plus (+) signs in the command line. Unless you specify otherwise, the system creates an object file with the same name as the first input file and gives it an .OBJ file type. To compile multiple files in independent compilations, separate the files with commas (,) in the command line. This generates a corresponding object file for each set of input files.

Language options are position-dependent — that is, they have different meanings depending on where you place them in the command line. Options that qualify a command name apply across the entire command string. Options that follow a file specification apply only to the file (or group of files separated by plus signs) that they follow in the command string.

The *RT–11/RSTS/E FORTRAN IV User's Guide* contains detailed information about using FORTRAN. The following sections describe the options you can use with the FORTRAN command.

/ALLOCATE:size Use this option with /LIST or /OBJECT to reserve space on a device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of –1 is a special case that creates the largest file possible on the device.

FORTRAN

/CODE:type Use this option to produce object code that is designed for a particular hardware configuration. The argument *type* represents a three-letter abbreviation for the type of code to be produced. The legal values are: EAE, EIS, FIS, and THR. See the *RT–11/RSTS/E FORTRAN IV User's Guide* for a complete description of the types of code and their function.

/DIAGNOSE Use this option to help analyze an internal compiler error. /DIAGNOSE expands the crash dump information to include internal compiler tables and buffers. Submit the diagnostic printout to DIGITAL with a software performance report (SPR) form. The information in the listing can help DIGITAL programmers locate the compiler error and correct it.

/EXTEND Use this option to change the right margin for source input lines from column 72 to column 80.

/HEADER This option includes in the printout a list of options that are currently in effect.

/I4 Use this option to allocate two words for the default integer data type (FORTRAN uses one-word integers) so that it takes the same physical space as real variables.

/LINENUMBERS Use this option to include internal sequence numbers in the executable program. These are especially useful in debugging a FORTRAN program. They identify the FORTRAN statements that cause run-time diagnostic error messages. This is the default operation.

/NOLINENUMBERS This option suppresses the generation of internal sequence numbers in the executable program. This produces a smaller program and optimizes execution speed. Use this option to compile only those programs that are already debugged; otherwise the line numbers in FORTRAN error messages are replaced by question marks and the messages are difficult to interpret.

/LIST[:filespec] You must specify this option to produce a FORTRAN compilation listing. The /LIST option has different meanings depending on where you place it in the command line.

The /LIST option produces a listing on the line printer when /LIST follows the command name. For example, the following command line produces a line printer listing after compiling a FORTRAN source file:

```
.FORTRAN/LIST MYPROG<RET>
```

When the /LIST option follows the file specification, it produces a listing file. For example, the following command line produces the listing file DK:MYPROG.LST after compiling a FORTRAN source file:

```
.FORTRAN MYPROG/LIST<RET>
```

If you specify /LIST without a file specification in the list of options that immediately follows the command name, the FORTRAN compiler generates a listing that prints on the line printer. If you follow /LIST with a device name, the system creates a listing file on that device. If the device is a file-

structured device, the system stores the listing file on that device, assigning it the same name as the input file with a .LST file type. The following command produces a listing on the terminal:

```
.FORTRAN/LIST:TT: A
```

The next command creates a listing file called A.LST on RK3:.

```
.FORTRAN/LIST:RK3: A
```

If the /LIST option contains a name and file type to override the default of .LST, the system generates a listing file with that name. The following command, for example, compiles A.FOR and B.FOR together, producing files A.OBJ and FILE1.OUT on device DK:.

```
.FORTRAN/LIST:FILE1.OUT A+B
```

Another way to specify /LIST is to type it after the file specification to which it applies. To produce a listing file with the same name as a particular input file, you can use a command similar to this one:

```
.FORTRAN A+B/LIST:RK3:
```

The above command compiles A.FOR and B.FOR together, producing files DK:A.OBJ and RK3:B.LST. If you specify a file name on a /LIST option following a file specification in the command line, it has the same meaning as when it follows the command. The following two commands have the same results:

```
.FORTRAN A/LIST:B
```

```
.FORTRAN/LIST:B A
```

Both the above commands generate A.OBJ and B.LST as output files.

Remember that file options apply only to the file (or group of files that are separated by plus signs) that they follow in the command string. For example:

```
.FORTRAN A/LIST+B
```

This command compiles A.FOR, producing A.OBJ and A.LST. It also compiles B.FOR, producing B.OBJ. However, it does not produce any listing file for the compilation of B.FOR.

/OBJECT[:filespec] Use this option to specify a file name or device for the object file. Because FORTRAN creates object files by default, the following two commands have the same meaning:

```
.FORTRAN A
```

```
.FORTRAN/OBJECT A
```

Both commands compile A.FOR and produce A.OBJ as output. The /OBJECT option functions like the /LIST option; it can be either a command or a file qualifier.

As a command option, /OBJECT applies across the entire command string. The following command, for example, compiles A.FOR and B.FOR separately, creating object files A.OBJ and B.OBJ on RK1:.

```
.FORTRAN/OBJECT:RK1: A,B
```

Use /OBJECT as a file option to create an object file with a specific name or destination. The following command compiles A.FOR and B.FOR together, creating files B.LST and B.OBJ.

```
.FORTRAN A+B/LIST/OBJECT
```

/NOOBJECT Use this option to suppress creation of an object file. As a command option, /NOOBJECT suppresses all object files; as a file option, it suppresses only the object file produced by compilation of the related input files. In this command, for example, the system compiles A.FOR and B.FOR together, producing files A.OBJ and B.LST. It also compiles C.FOR and produces C.LST, but does not produce C.OBJ.

```
.FORTRAN A+B/LIST,C/NOOBJECT/LIST
```

/ONDEBUG Use this option to include debug lines (those that have a D in column one) in the compilation. You do not, therefore, have to edit the file to include these lines in the compilation or to logically remove them. This option is useful in debugging a program. You can include messages, flags, and conditional branches to help you trace program execution and find an error.

/OPTIMIZE:type Use this option to enable certain options that optimize object code for various conditions. The argument *type* represents the three-letter code for the type of optimization to be enabled. Table 4–4 summarizes the codes and their meaning.

Table 4–4: Optimization Codes

Code	Meaning
BND	Global register bindings for inline code generation
CSE	Common subexpression elimination
SPD	Optimization for speed of execution, as opposed to minimal program size
STR	Strength reduction optimization

This option is not available with version 2.5 of the FORTRAN compiler.

/NOOPTIMIZE:type Use this option to disable certain options that optimize object code for various conditions. The argument *type* represents the three-letter code for the type of optimization to be disabled. Table 4–4 summarizes the codes and their meaning. This option is not available with version 2.5 of the FORTRAN compiler.

/RECORD:length Use this option to override the default record length of, usually 132, characters for ASCII, sequentially formatted input and output. The meaningful range for length is from 4 to 4095.

/SHOW[:value] Use this option to control FORTRAN listing output. The argument *value* represents a code that indicates which listings the compiler is to produce. Table 4–5 summarizes the codes and their meaning. You can combine options by specifying the sum of their numeric codes. For example:

/SHOW:7

or

/SHOW:ALL

The two options shown above have the same meaning. If you specify no code, the default value is 3, a combination of SRC and MAP.

Table 4–5: FORTRAN Listing Codes

Code	Listing Content
0	Diagnostics only
1 or SRC	Source program and diagnostics
2 or MAP	Storage map and diagnostics
3	Diagnostics, source program, and storage map
4 or COD	Generated code and diagnostics
7 or ALL	Diagnostics, source program, storage map, and generated code

/STATISTICS Use this option to include compilation statistics in the listing, such as amount of memory used, amount of time elapsed, and length of the symbol table.

/SWAP Use this option to permit the USR (user service routine) to swap over the FORTRAN program in memory. This is the default operation.

/NOSWAP This option keeps the USR resident during execution of a FORTRAN program. This may be necessary if the FORTRAN program uses some of the RT–11 system subroutine library calls (see Chapter 4 of the *RT–11 Programmer's Reference Manual*). If the program frequently updates or creates a large number of different files, making the USR resident can improve program execution. However, the cost for making the USR resident is 2K words of memory.

/UNITS:n Use this option to override the default number of logical units (6) to be open at one time. The maximum value you can specify for *n* is 16.

/VECTORS This option directs FORTRAN to use tables to access multidimensional arrays. This is the default mode of operation.

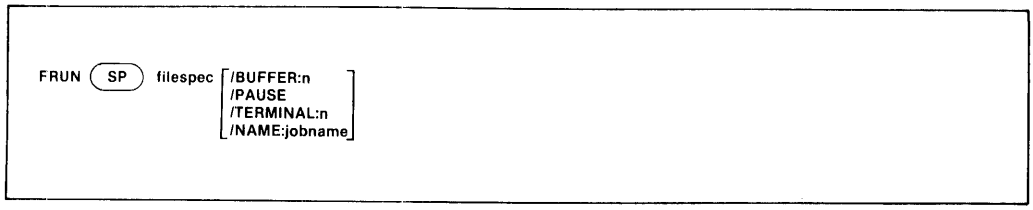
/NOVECTORS This option directs FORTRAN to use multiplication operations to access multidimensional arrays.

/WARNINGS Use this option to include warning messages in FORTRAN compiler diagnostic error messages. These messages call certain conditions to your attention, but do not interfere with the compilation. A warning mes-

sage prints, for example, if you change an index within a DO loop, or if you specify a variable name longer than six characters.

/NOWARNINGS Use this option to exclude warning messages in FORTRAN compiler diagnostic error messages. This is the default setting.

The FRUN command initiates foreground jobs. The default file type is .REL.



In the command syntax illustrated above, *filespec* represents the program to execute. Because this command runs a foreground job, it is valid for the FB and XM monitors only.

If a foreground job is active when you issue the FRUN command, an error message prints on the terminal. You can run only one foreground job at a time. If a terminated foreground job is occupying memory, the system reclaims that region for your program. Then, if the system finds your program and if your program fits in the available memory, execution begins.

The following sections describe the options you can use with FRUN. Note that the option must follow the file specification in the command line.

Note that you can use the FRUN command to run a virtual foreground job, and that you can use FRUN to run a virtual .SAV file in the foreground under the XM monitor.

/BUFFER:n Use this option to reserve more space in memory than the actual program size. The argument *n* represents, in octal, the number of words of memory to allocate. You must use this option to execute a FORTRAN foreground job. If you use /BUFFER for a virtual job linked with the /V option (or /XM), the system ignores /BUFFER because it has already provided a buffer in extended memory.

The following formula determines the space needed to run a FORTRAN program as a foreground job.

$$n = [1/2[504 + (33*N) + (R-136) + A*512]]$$

where:

A represents the maximum number of files open at any one time. Each file opened as double buffered should be counted as two files.

N represents the maximum number (decimal) of simultaneously open channels (logical unit numbers). This value is specified when the compiler is built, and can be overridden with the /UNITS option during main program compilation; the default value is 6. Make sure you use a decimal point with this number.

R represents the maximum formatted sequential record length. This value is specified when the compiler is built and can be overridden with the /RECORD option during main program compilation; the default value is 136.

This formula must be modified for certain System Subroutine Library (SYSLIB) functions.

The IQSET function requires the formula to include additional space for queue elements (qcount) as follows:

$$n = [1/2[504 + (33*N) + (R-136) + A*512]] + [10*qcount]$$

The ICDFN function requires the formula to include additional space for the integer number of channels (num) as follows:

$$n = [1/2[504 + (33*N) + (R-136) + A*512]] + [6*num]$$

The INTSET function requires the formula to include additional space for the number of INTSET calls issued in the program as follows:

$$n = [1/2[504 + (33*N) + (R-136) + A*512]] + [25*INTSET]$$

Any functions, including INTSET, that invoke completion routines must include 64(decimal) words plus the number of words needed to allocate the second record buffer (default is 68(decimal) words). The length of the record buffer is controlled by the /RECORD option to the FORTRAN compiler. If the /RECORD option is not used, the allocation in the formula must be 136(decimal) bytes, or the length that was set at FORTRAN installation time. This modifies the formula as follows:

$$n = [1/2[504 + (33*N) + (R-136) + A*512]] + [64 + R/2]$$

If the /BUFFER option does not allocate enough space in the foreground on the initial call to a completion routine, the following message appears:

```
?ERR 0, NON-FORTRAN error call
```

This message also appears if there is not enough free memory for the background job or if a completion routine in the single-job monitor is activated during another completion routine. In the latter case, the job aborts; you should use the FB monitor to run multiple active completion routines.

/PAUSE Use this option to help you debug a program. When you type the carriage return at the end of the command string, the system prints the load address of your program and waits. You can examine or modify the program (by using ODT, described in Chapter 21) before starting execution. You must use the RESUME command to start the foreground job. The following command loads the program DEMOSP.REL, prints the load address, and waits for a RESUME command to begin execution.

```
.FRUN DEMOSP/P
Loaded at 127276
.RESUME
```

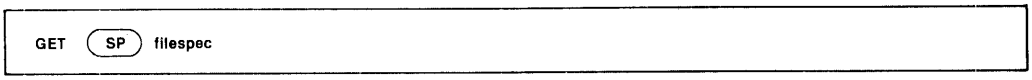
/TERMINAL:n This option is meaningful only in a multi-terminal system. Use it to assign a terminal to interact with the foreground job. The argument *n* represents a terminal logical unit number. If you do not use this option, the foreground job shares the console terminal with the background job. By assigning a different terminal to interact with the foreground job, you eliminate the need for the foreground and background jobs to share the console terminal. Note that the original console terminal still interacts with the back-

ground job and with the keyboard monitor, unless you use the SET TT:CONSOL command to change this.

/NAME:name Use this option to assign a logical name to the foreground job. This option is valid only on a monitor that has system job support, a special feature enabled by the system generation process.

GET

The GET command loads a memory image file into memory.



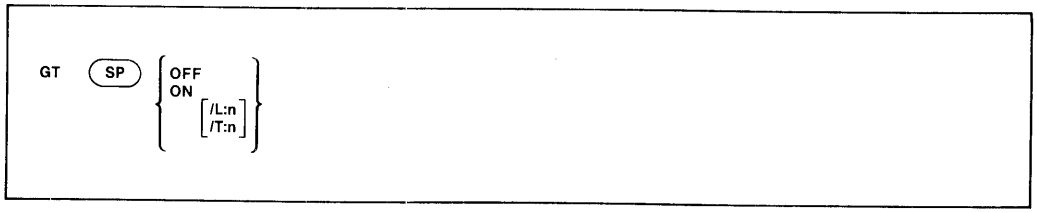
In the command syntax shown above, *filespec* represents the memory image file to be loaded. The default file type is .SAV. Note that magtape and cassette are not block-replaceable devices and therefore are not permitted with the GET command. Use the GET command for a background job only. You cannot use GET on a virtual program that executes under the XM monitor. The GET command is useful when you need to modify or debug a program. You can use GET with the Base, Deposit, Examine, and START commands to test changes. Use the SAVE command to make these changes permanent. You can combine programs by issuing multiple GET commands, as the following example shows. This example loads a program, DEMOSP.SAV, loads ODT.SAV (on-line debugging technique, described in Chapter 21), and starts the program using the address of ODT's entry point.

```
.GET DEMOSP

.GET ODT

.START

ODT V01.04
*
```

If more than one program requires the same locations in memory, the program you load later overlays the previous program. Note that you cannot use GET to load overlay segments of a program; it can load only the root. If the file you need to load resides on a device other than the system device, the system automatically loads that device handler into memory when you issue the GET command. This prevents problems that occur if you use the START command and your program is overlaid.

GT

The GT command enables or disables the VT11 or VS60 display hardware.



When you issue the GT OFF command, you disable the display hardware. The printing console terminal then becomes the device that prints output from the system.

When you issue the GT ON command, the display screen replaces the printing console terminal. The display screen offers some advantages over the printing terminal: (1) it is quieter than a printing terminal, (2) it is faster than a printing terminal, (3) it does not require a supply of paper, and (4) it is the device for which EDIT's immediate mode is intended. The display screen can speed up the editing process (see Chapter 5 for information on how to use the text editor). You can use CTRL/A, CTRL/S, CTRL/E, and CTRL/Q to control scrolling. These commands are explained in Chapter 3.

Note that RT–11 does not permit you to use display hardware (with GT ON) if you have multi-terminal support (enabled by a user-generated monitor) or if you have an 8K configuration. You cannot use GT ON or GT OFF when a foreground or system job is active; this causes the system to print an error message. Issue the GT ON command before you begin execution of the foreground job. ODT (on-line debugging technique, described in Chapter 21) is the only system program that cannot use the display screen. Its output always appears on the console terminal. You can use VDT, a variant of ODT, because it can interact with the display hardware.

NOTE

If an indirect command file issues a GT ON command, part of the command may echo on the terminal and the rest may echo on the graphics screen. Also, if you type the GT ON command, followed by CTRL/E, the initial line on the terminal overprints when you type GT OFF.

The following options control the number of lines that appear on the screen and position the first line vertically.

/L:n Use this option to change the number of lines of text that display on the screen. Table 4–6 shows the valid range for the argument *n* in decimal. If you do not use this option the system determines the screen size and automatically assigns the largest valid value.

GT

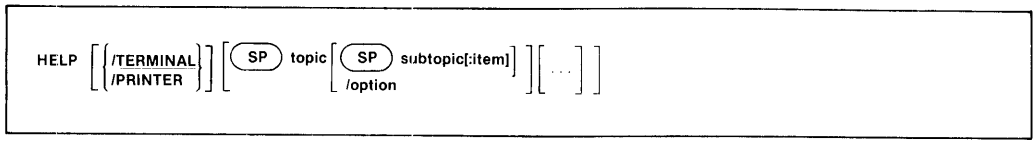
/T:n Use this option to change the top position of the scroll display. Table 4–6 shows the valid range for the argument *n* in decimal. If you do not use this option, the system determines the screen size and automatically assigns the largest valid value.

Table 4–6: Display Screen Values

Screen Size	Lines	Top Position
12-inch	1–31	1–744
17-inch	1–40	1–1000 (or larger)

HELP

The HELP command lists information related to RT–11 commands to help you remember command syntax, options, and so on, when you are at the console.



In the command syntax shown above, *topic* represents a subject about which you need information. In the help file supplied with RT–11, the topics are the keyboard monitor commands. The *subtopic* represents a category within a topic. In the RT–11 help file, the subtopics are SYNTAX, SEMANTICS, OPTIONS, and EXAMPLES. The *item* represents one member of the subtopic group. You can specify more than one item in the command line if you separate the items by colons (:). If you type HELP followed by a carriage return, the system lists information on the HELP command.

The HELP command permits you to access the HELP text file. The help file distributed with RT–11 contains information about the keyboard monitor commands and how to use them. However, the concept of the help file is a general one. That is, you can create your own help file to supply quick reference material on any subject. For information on how to change the HELP text file, see the *RT–11 Installation and System Generation Guide*. There are only two options you can use with the HELP command. They are /PRINTER and /TERMINAL.

/PRINTER Use this option to list information on the line printer.

/TERMINAL This option lists information on the console terminal. This is the default operation.

The following examples all make use of the standard RT–11 help file.

The following command lists all the topics for which assistance is available.

```
.HELP *

APL         Invokes the APL language interpreter
ASSIGN      Associates a logical device name with a physical device
B           Sets a relocation base
   .
   .
   .
```

The next command lists all the information about the DATE command.

```
.HELP DATE

DATE       Sets or displays the current system date

  SYNTAX
          DATE[ dd-mmm-yy]

  SEMANTICS
          All numeric values are decimal;  mmm is the first three
          characters of the name of the month

  OPTIONS
          None

  EXAMPLES
          DATE 12-MAY-79
```

The next command lists all the options that are valid with the DIRECTORY command.

```
.HELP DIRECTORY OPTIONS

  OPTIONS
    ALLOCATE:size
        Use with /OUTPUT to reserve space for the output  listing
        file
    ALPHABETIZE
        Sorts the directory in alphabetical order  by  file  name
        and type
      .
      .
      .
```

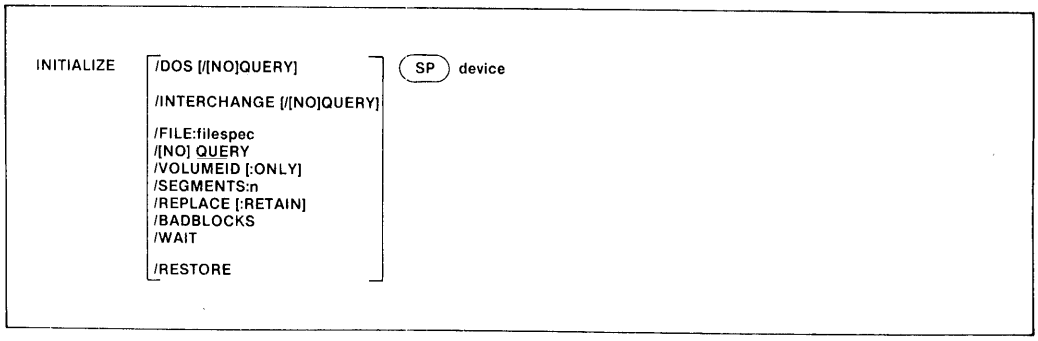
The next command lists information about the /BRIEF option for the DIRECTORY command.

```
.HELP DIRECTORY OPTIONS:BRIEF

    BRIEF
        Lists only file names and file types of files;   same  as
        /FAST
```

The following command lists information about the DIRECTORY command options that begin with B.

```
.HELP DIRECTORY/B

    BADBLOCKS

        Scans the device for bad blocks  and  types  their  octal
        number
    BEFORE[date]
        Lists the files created before the date you specify
    BEGIN
        Lists the directory, starting with the file you specify
    BLOCKS
        Lists the starting block numbers of the files
    BRIEF
        Lists only file names and file types of files;   same  as
        /FAST
```

Use the INITIALIZE command to clear and initialize a device directory.



In the command syntax illustrated above, *device* represents the volume you need to initialize. The initialize operation must always be the first operation you perform on a new volume after you receive it, formatted, from the manufacturer. If the volume is not formatted, use the FORMAT command (see the FORMAT command description) to format the volume. After you use the INITIALIZE command, there are no files in the directory. If you use the INITIALIZE command with no options, the system simply initializes the device directory. You can enter the INITIALIZE command as one line, or you can rely on the system to prompt you for the name of the device with *Device?*. The following sections describe the options you can use with INITIALIZE and give some examples of their use.

The default number of directory segments for RT-11 directory structured volumes is listed in Table 4-7. If any default is too small for your needs, see the *RT-11 Installation and System Generation Guide* for details on changing this default directory size.

If the volume you are initializing has protected files, the system always prints a confirmation message as in the following example.

```
.INIT RK0:
RK0:/Initialize; Are you sure? Y <RET>
Volume contains protected files; Are you sure? Y <RET>
```

/BADBLOCKS[:RET] Use this option to scan a volume (disk or DECtape) for bad blocks and write .BAD files over them. For each bad block the system encounters on the volume, it creates a file called FILE.BAD to cover it. After the volume is initialized and the scan completed, the directory consists of only FILE.BAD entries to cover the bad blocks. This procedure ensures that the system will not attempt to access these bad blocks during routine operations. If the system finds a bad block in either the boot block or the volume directory, it prints an error message and the volume is not usable. DIGITAL recommends that you use the DIRECTORY/BADBLOCKS command after using the INITIALIZE/BADBLOCKS command so that you can find out

where the bad blocks are, if any. The following command initializes volume RK1: and scans for bad blocks.

```
.INITIALIZE/BADBLOCKS RK1:
RK1:/Initialize;  Are your sure?  Y
```

If you use /BADBLOCKS:RET, the system will retain through initialization all files with a .BAD file type that it finds on the volume, giving them the name FILE.BAD. The system does not do a bad block scan. The advantage in using :RET is that initializing takes less time. Note that some volumes support bad block replacement; DIGITAL recommends you use the /REPLACE[:RET] option instead of /BADBLOCKS[:RET] for these volumes when scanning for bad blocks.

If you use INITIALIZE/BADBLOCKS with a volume that has been previously initialized with the INITIALIZE/REPLACE command, .BAD files will be written over all bad blocks and the bad block replacement table will be ignored by the system.

If the volume being initialized contains bad blocks, the system prints the locations of the bad blocks in octal and in decimal, as in the following example:

```
.INITIALIZE/BADBLOCKS RK0:
RK0:/Initialize;  Are you sure?  Y
    Block           Type
000120     80.  Hard
000471    313.  Hard
000521    337.  Hard
?DUP-I-Bad blocks detected 3.
```

The left column lists the locations in octal, and the middle column lists the locations in decimal. The right column indicates the type of bad block found: hard or soft.

/DOS Use this option to initialize a DECtape for DOS–11 format.

/FILE:filespec Use this option to initialize a magtape and create a bootable tape. For *filespec,* substitute *dev:MBOOT.BOT*. This file is distributed with RT–11 for this purpose only. Consult the *RT–11 Installation and System Generation Guide* for more information. The following example creates a bootable magtape.

```
.INITIALIZE/FILE:MBOOT.BOT  MT0:
```

/INTERCHANGE Use this option to initialize a diskette for interchange format. The following example initializes DX1: in interchange format.

```
.INITIALIZE/INTERCHANGE DX1:
DX1:/Init  Are you sure?  Y
```

NOTE

The directory of an initialized interchange diskette has a single file entry, DATA, that reserves the entire diskette. You must delete this file before you can write any new files on the diskette. Do this by using the following command:

```
DELETE/INTERCHANGE DX1:DATA
```

This is necessary for IBM compatibility.

/QUERY This option prompts you for confirmation before it initializes a device. Respond by typing a Y followed by a carriage return to initiate execution of the command. The system interprets a response beginning with any other character to mean NO. /QUERY is the default operation.

/NOQUERY Use this option to suppress the confirmation message the system prints before it proceeds with the initialization.

/REPLACE[:RET] If you have an RK06, RK07, RL01, or RL02 disk, use this option to scan a disk for bad blocks. If the system finds any bad blocks, it creates a replacement table so that routine operations access good blocks instead of bad ones. Thus, the disk appears to have only good blocks. Note, though, that accessing this replacement table slows response time for routine input and output operations. With /REPLACE, you have the option of deciding which bad blocks you want replaced if there is a replacement table overflow. The RK06s and RK07s support up to 32 bad blocks in the replacement table; the RL01s and RL02s support up to 10.

With an RK06 or RK07 disk, the system can replace only those bad blocks that generate a bad sector error (BSE). Of the blocks the system cannot replace, the system can report a bad block as being hard or soft. If you use /VERIFY with /REPLACE and the system still cannot use the block, the system reports a hard bad block. If the system can use the block, it reports a soft bad block.

With an RL01 or RL02 disk, the system can replace any kind of bad block.

When you use /REPLACE, the system prints a list of replaceable bad blocks as in the following sample:

```
.INITIALIZE/REPLACE DL0:

    Block          Type
030722   12754.  Replaceable
115046   39462.  Replaceable
133617   46991.  Replaceable
136175   48253.  Replaceable
136277   48319.  Replaceable
136401   48385.  Replaceable
140405   49413.  Replaceable
146252   52394.  Replaceable
DUP-I-Bad blocks detected 8.
```

If there is a replacement table overflow, the system prompts you to indicate which blocks you want replaced as follows:

```
?DUP-W-Replacement table overflow
Type <RET>, 0, or nnnnnn (<RET>)
Replace block #
```

nnnnnn represents the octal number of the block you want the system to replace.

After you enter a block number, the system responds by repeating the *Replace block #* prompt. If you type a 0 at any time you do not want any more blocks replaced, prompting ends and any blocks not placed in the replacement table are marked as FILE.BAD.

If you enter a carriage return at any time, the system places all bad blocks you have not entered into the replacement table, starting with the first on the disk, until the table is full. The system assigns the name FILE.BAD to any remaining bad blocks and prompting ends.

If you use /NOQUERY with /REPLACE, and there is a replacement table overflow, the effect will be as if you had entered a carriage return in response to the first *Replace block #* prompt.

If you use :RET with /REPLACE, the system initializes the volume and retains the bad block replacement table (and FILE.BAD files) created by the previous /REPLACE command.

Note that if the monitor file resides on a block that contains a bad sector error (BSE) and you are doing bad block replacement, a boot error results when you attempt to bootstrap the system. In this case, move the monitor. Use the DIRECTORY/BADBLOCKS/FILES command to determine which files reside on bad blocks.

/RESTORE Use this option to *uninitialize* a volume. That is, you can use this option to restore the directory and files that were present on the volume prior to the previous initialization. You can use /RESTORE only if no files have been transferred to the volume since the last time it was initialized.

The /RESTORE option does not restore the boot blocks; so if you use /RESTORE to restore a previously bootable volume, use the COPY/BOOT command to make the volume bootable again.

/SEGMENTS:n Use this option if you need to initialize a disk and also change the number of directory segments. The number of segments in the directory establishes the number of files that can be stored on a device. The system allows a maximum of 72 files per directory segment, and 31 directory segments per device. The argument *n* represents the number of directory segments. The valid range for *n* is from 1 to 31 (decimal). Table 4–7 shows the default values of *n* for standard RT–11 devices.

Table 4–7: Default Directory Sizes

Device	Size (decimal) of Directory in Segments
RK	16
DD	1
DT	1
RF	4
DS	4
DP	31
DX	1
DM	31
DY	4
DL	16
PD	1

/VOLUMEID[:ONLY] Use /VOLUMEID to write a volume identification on a device when you initialize it. This identification consists of a volume ID (up to 12 characters long for a block-replaceable device, up to 6 characters long for magtape) and an owner name (up to 12 characters long for a block-replaceable device, up to 10 characters long for magtape). The following example initializes device RK1: and writes a volume identification on it.

```
.INITIALIZE/VOLUMEID RK1:
RK1:/Initialize: Are you sure? Y
 Volume ID? FORTRAN VOL
 Owner?     Marcy
```

Use /VOLUMEID:ONLY to write a new volume identification on a device without reinitializing the device. You cannot change the volume ID of a magtape or cassette without initializing the entire tape.

/WAIT The /WAIT option is useful if you have a single-disk system. When you use this option to initialize a volume, the system begins the procedure but then pauses and waits for you to mount the volume you want to initialize. When the system pauses, it prints the following prompt at the terminal:

```
Mount input volume in <device>:  Continue?
```

<device> is the name of the device into which you mount the volume to be initialized. After you have mounted the input volume, type Y followed by a carriage return. After the system completes the initialization process, it prints the following message prompting you to mount the system volume:

```
Mount system volume in <device>?  Continue?
```

After you mount the system volume, type Y followed by a carriage return. When you use /WAIT, make sure that DUP is on the system volume.

INSTALL

The INSTALL command installs the device you specify into the system.

INSTALL (SP) device [,...device]

In the command syntax shown above, *device* represents the name of the device to be installed. The INSTALL command accepts no options. It allows you to install into the system tables a device that was not installed into the system when it was bootstrapped. (A device handler must exist in the system tables before you can use that device.) The device occupies the first available device slot. Using the INSTALL command does not change the monitor disk image; it only modifies the system tables of the monitor that is currently in memory.

You can enter the command on one line, or you can rely on the system to prompt you for information. The INSTALL command prompt is *Device?*.

When you specify a device name, the system searches the system device for the corresponding device handler file. For SJ and FB systems, if LP: is to be installed, the INSTALL command searches for the file SY:LP.SYS. For XM systems, INSTALL searches for SY:LPX.SYS. The INSTALL command does not allow a device handler built for a different configuration of the system to be installed in a given system. Note that you cannot install the device names SY, DK, or BA.

To permanently install a device, include the INSTALL command in the standard, system startup indirect command file. This file is invoked automatically when you boot the system. The INSTALL command also allows you to configure a special system for a single session without having to reconfigure to revert to the standard device configuration. If there are no free device slots (use the SHOW DEVICES command to ascertain this), you must remove an existing device (with the REMOVE command) before you can install a new device.

The following command installs the card reader into the system tables from the file CR.SYS. (The colon (:) that follows the device handler name is optional.)

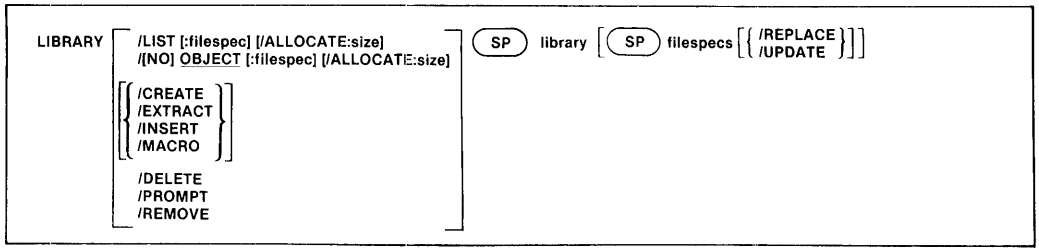
```
.INSTALL CR:
```

The next example installs the line printer, the card reader, and DECtape.

```
.INSTALL LP:,CR:,DT:
```

LIBRARY

The LIBRARY command lets you create, update, modify, list, and maintain library files.



In the command syntax illustrated above, *library* represents the library file name, and *filespecs* represents the input module file names. Separate the library file specification from the module file specifications with a space. Separate the module file specifications with commas. The system uses .LST as the default file type for library directory listing files. It uses .OBJ as the default file type for object libraries and object input files, and .MAC for macro libraries and macro input files. Object libraries contain machine-level object modules, and macro libraries contain MACRO source modules. You cannot combine object modules with MACRO modules. The default operation, if you do not specify an option, is /INSERT. If you do not specify a library file in the command line, the system prompts *Library?*. If you specify /CREATE, /INSERT, or /MACRO and omit the module file specification, the system prompts *Files?*. If you specify /EXTRACT, the system prompts *File?*. Note that no other option causes the *File?* or *Files?* prompt.

The LIBRARY command can perform all the functions listed above on object library files. It can also create macro library files for use with the MACRO–11 assembler. A library file is a direct-access file (a file that has a directory) that contains one or more modules of the same type. The system organizes library files so the linker and MACRO–11 assembler can access them rapidly. Each library is a file that contains a library header, library directory, and one or more object modules. The object modules in a library file can be routines that are repeatedly used in a program, routines that are used by more than one program, or routines that are related and simply gathered together for convenience. An example of a typical object library file is the default system library, SYSLIB.OBJ, used by the linker. An example of a macro library file is SYSMAC.SML.

You access object modules in a library file by making calls or references to their global symbols; you link the object modules with the program that uses them by using the LINK command to produce a single executable module. Each input file for an object library consists of one or more object modules, and is stored on a device under a specific file name and file type. Once you insert an object module into a library file, you no longer reference the module by the file name of which it was a part; reference it by its individual module name. For example, the input file FORT.OBJ may exist on DT2: and can

contain an object module called ABC. Once you insert the module into a library, reference only ABC, and not FORT.OBJ.

The input files normally do not contain main programs but only subprograms, functions, and subroutines. The library file must never contain a FORTRAN BLOCK DATA subprogram because there is no undefined global symbol to cause the linker to load it automatically.

The following sections describe the LIBRARY command options and explain how to use them. The last section under this command describes the LIBRARY prompting sequence and order of execution for commands that combine two or more LIBRARY options. Chapter 12 contains more detailed information on object and macro libraries.

/ALLOCATE:size Use this option only with /LIST or /OBJECT to reserve space on the device for the output file. The value *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of -1 is a special case that allocates the largest area available on the device.

The following example uses /ALLOCATE to create the object library MYLIB.OBJ from the object library MYFILE.OBJ. The argument, -1, is specified with /ALLOCATE.

```
LIBRARY/OBJECT:MYLIB/ALLOCATE:-1 MYFILE
```

/CREATE Use this option by itself to create an object library. Specify a library name followed by the file specifications for the modules that are to be included in that library. The following command creates a library called NEWLIB.OBJ from the modules contained in files FIRST.OBJ and SECOND.OBJ.

```
.LIBRARY/CREATE NEWLIB FIRST,SECOND
```

/DELETE Use this option to delete an object module and all its associated global symbols from a library file directory. Since the module is deleted only from the directory (and not from the object module itself), the module and all global symbols that were previously deleted are restored whenever you update that library, unless you use /DELETE again to delete them. Specify the library name in the command line. The system prompts you for the names of the modules to delete. The prompt is:

```
Module name?
```

Respond with the name of a module. (Be sure to specify a module name and not a global name.) Follow each module name with a carriage return. Enter a carriage return on a line by itself to terminate the list of module names. The following example deletes modules SGN and TAN from the library called NEWLIB.OBJ.

```
.LIBRARY/DELETE NEWLIB
Module name? SGN
Module name? TAN
Module name?
```

/EXTRACT Use this option to extract an object module from a library and store it in a file with the same name as the module and a file type of .OBJ.

You cannot combine this option with any other option. The system prompts you for the name of the object module to be extracted. The prompt is:

```
Global?
```

If you specify a global name, the system extracts the entire module of which that global is a part. Follow each global name with a carriage return. Enter a carriage return on a line by itself to terminate the list of global symbols. The following example shows how to extract the module ATAN from the library called NEWLIB.OBJ and store it in file ATAN.OBJ on DX1:.

```
.LIBRARY/EXTRACT
Library? NEWLIB
File   ? DX1:ATAN
Global ? ATAN
Global ?
```

/INSERT Use this option to insert an object module into an existing library. Although you can insert object modules that have duplicate names, this practice is not recommended because of the difficulty involved in replacing or updating these modules. Note that /INSERT is the default operation. If you do not specify any option, insertion takes place. The following example inserts the modules contained in the files THIRD.OBJ and FOURTH.OBJ into the library called OLDLIB.OBJ.

```
.LIBRARY/INSERT OLDLIB THIRD,FOURTH
```

/LIST[:filespec] Use this option to obtain a directory listing of an object library. The following example obtains a directory listing of OLDLIB.OBJ on the terminal (the line printer is the default device).

```
.LIBRARY/LIST:TT:  OLDLIB
```

The directory listing prints global symbol names. A plus sign (+) in the module column indicates a continued line. See Section 12.2.7 for a procedure to include module names in the directory listing.

You can also use /LIST with other options (except /MACRO) to obtain a directory listing of an object library after you create or modify it. The following command, for example, inserts the modules contained in the files THIRD.OBJ and FOURTH.OBJ into the library called OLDLIB.OBJ; it then prints a directory listing of the library on the terminal.

```
.LIBRARY/INSERT/LIST:TT:  OLDLIB THIRD,FOURTH
```

You cannot obtain a directory listing of a macro library.

Make sure when you use /LIST with LIBRARY that you use it on the command side of the command string, and not after the file specification.

/MACRO Use this option to create a macro library. Note that this is the only valid function for a macro library. You can create a macro library, but you cannot list or modify it. To update a macro library, simply edit the ASCII text file and then reprocess the file with the LIBRARY/MACRO com-

mand. The following example creates a macro library called NEWLIB.MAC from the ASCII input file SYSMAC.MAC.

```
.LIBRARY/MACRO/CREATE NEWLIB SYSMAC
```

When you use /MACRO with LIBRARY, use it on the command side of the command string, and not after the file specification.

/OBJECT[:filespec] The system creates object library files by default as a result of executing a LIBRARY command. When you modify an existing library, the system actually makes the changes to the library you specify, thus creating a new, updated library that it stores under the same name as the original library. Use this option to give a new name to the updated library file and preserve the original library. The following example creates a library called NEWLIB.OBJ, which consists of the library OLDLIB.OBJ plus the modules that are contained in files THIRD.OBJ and FOURTH.OBJ.

```
.LIBRARY/INSERT/OBJECT:NEWLIB OLDLIB THIRD,FOURTH
```

/NOOBJECT Use this option to suppress the creation of a new object library as a result of a LIBRARY command.

/PROMPT Use this option to specify more than one line of input file specifications in a LIBRARY command. This option is valid with all other library functions except the /EXTRACT option. You must specify // as the last input in order to terminate the input list. Note that the file specifications you enter after typing the /PROMPT option must conform to Command String Interpreter conventions. The following example creates a macro library called MACLIB.MAC from seven input files.

```
.LIBRARY/MACRO/PROMPT MACLIB A, B, C, D
*E,F,G
*//
```

/REMOVE This option permits you to delete a specific global symbol from a library file's directory. Since globals are deleted only from the directory (and not from the object module itself), all the globals that were previously deleted are restored whenever you update that library, unless you use /REMOVE again to delete them. This feature lets you recover a library if you have inadvertently deleted the wrong global. The system prompts you for the names of the global symbols to remove. The prompt is:

```
Global?
```

Respond with the name of a global symbol to be removed. Follow each global symbol with a carriage return. Enter a carriage return on a line by itself to terminate the list of global symbols. The following example deletes the globals GA, GB, GC, and GD from the library OLDLIB.OBJ.

```
.LIBRARY/REMOVE OLDLIB
Global? GA
Global? GB
Global? GC
Global? GD
Global?
```

/REPLACE Use this option to replace modules in an existing object library with modules of the same name contained in the files you specify. The following example replaces a module called SQRT in the library MATHLB.OBJ with a new module, also called SQRT, from the file called MFUNCT.OBJ.

```
.LIBRARY MATHLB MFUNCT/REPLACE
```

Note that the /REPLACE option must follow each file specification that contains a module to be inserted into the library. Note also that you can use /REPLACE only with a module(s), and never a library file(s).

/UPDATE This option combines the functions of /INSERT and /REPLACE. Specify it after each file specification to which it applies. If the modules in the input file already exist in the library, the system replaces those library modules. If the modules in the input file do not exist in the library, the system inserts them. The following example updates the library OLDLIB.OBJ.

```
.LIBRARY OLDLIB FIRST/UPDATE,SECOND/UPDATE
```

Note that the /UPDATE option must follow each file specification to which it applies, and that you can use this option only with modules, not files.

You can combine the LIBRARY options with the exceptions of /EXTRACT and /MACRO, which you cannot combine with most of the other functions. Table 4–8 lists the sequence in which the system executes the LIBRARY options and prompts you for additional information.

Table 4–8: Execution and Prompting Sequence of LIBRARY Options

Option	Prompt
/CREATE /DELETE /REMOVE /UPDATE /REPLACE /INSERT /LIST	Module name? Global?

The following example combines several options.

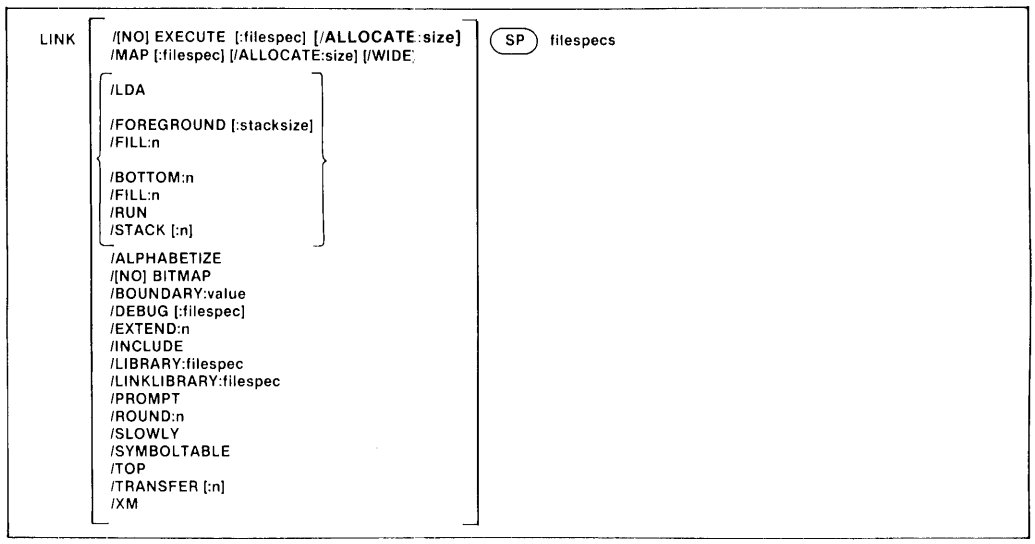
```
LIBRARY/LIST:TT:/REMOVE/INSERT NEWLIB LIB2/REPLACE,LIB3
Global? SQRT
Global?
RT-11 LIBRARIAN V03.10  FRI 15-JUL-79 00:08:37
NEWLIB                  FRI 15-JUL-79 00:08:35

MODULE          GLOBALS         GLOBALS         GLOBALS

                COS             SIN
                DATAN           DATAN2
                ATAN            ATAN2
                DCOS            DSIN
```

LIBRARY

The command executes in the following sequence:

1. Removes global SQRT from NEWLIB
2. Replaces any duplicates of the modules in the file LIB2.OBJ
3. Inserts the modules in the file LIB3.OBJ
4. Lists the directory of NEWLIB.OBJ on the terminal

The LINK command converts object modules into a format suitable for loading and execution.



The RT–11 system lets you separately assemble a main program and each of its subroutines without assigning an absolute load address at assembly time. The linker can then process the object modules of the main program and subroutines to relocate each object module and assign absolute addresses. It links the modules by correlating global symbols that are defined in one module and referenced in another, and it creates the initial control block for the linked program. The linker can also create an overlay structure (if you specify the /PROMPT option) and include the necessary run-time overlay handlers and tables. The linker searches libraries you specify to locate unresolved global symbols, and it automatically searches the default system subroutine library, SYSLIB.OBJ, to locate any remaining unresolved globals. Finally, the linker produces a load map (if you specify /MAP) that shows the layout of the executable module. See Chapter 11 for a more detailed explanation of the RT–11 linker. The linker also can produce and STB file.

In the command syntax illustrated above, *filespecs* represents the object modules to be linked. Each input module should be stored on a random-access device (disk, diskette, DECtape, or DECtape II); the output device for the load map file can be any RT–11 device. The output for an .LDA file (if you specify /LDA) can also be any RT–11 device, even those that are not block replaceable, such as paper tape.

The default file types are as follows:

Load Module	:	.SAV,.REL(/FOREGROUND), .LDA(/LDA)
Map Output	:	.MAP
Object Module	:	.OBJ

If you specify two or more files to be linked, separate the files by commas. The system creates an executable file with the same name as the first file in the input list (unless you use /EXECUTE to change it).

The LINK command options and explanations of how to use them follow. Table 4–9 summarizes LINK prompting sequence for commands that combine two or more LINK options.

Table 4–9: Prompting Sequence for LINK Options

Option	Prompt
/TRANSFER	Transfer symbol?
/STACK	Stack symbol?
/EXTEND:n	Extend section?
/BOUNDARY:value	Boundary section?
/ROUND:n	Round section?
/INCLUDE	Library search?

If you combine any of the options listed in Table 4–9, the system prompts you for information in the sequence shown in the table. Note that the *Library search?* prompt is always last. This is the only prompt that accepts more than one line as a response. For all the prompts, terminate your response with a carriage return. Terminate your list of responses to the *Library search?* prompt by typing an extra carriage return. Note that if the command lines are in an indirect file and the system encounters an end-of-file before all the prompting information has been supplied, it prints the prompt messages on the terminal.

/ALLOCATE:size Use this option with /EXECUTE or /MAP to reserve space on the device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of –1 is a special case that creates the largest file possible on the device. When used with /EXECUTE, /ALLOCATE is valid only when you are generating a .REL or .LDA file.

/ALPHABETIZE When you use this option, the linker lists in the load map your program's global symbols in alphabetical order.

/BITMAP Use this option if you want the linker to create a memory usage bitmap. This is the default setting.

/NOBITMAP Use this option if you do not want the linker to create a memory usage bitmap. This option is useful if you are preparing your program for ROM storage and its code lies between locations 360 and 377 inclusive. /BITMAP is the default setting.

/BOTTOM:n Use this option to specify the lowest address to be used by the relocatable code in the load module. The argument *n* represents a six-digit unsigned, even octal number. If you do not use this option, the linker

positions the load module so that the lowest address is location 1000 (octal). This option is invalid for foreground links.

/BOUNDARY:value Use the /BOUNDARY option to start a specific program section on a particular address boundary. The system generates a whole number multiple of the value you specify for the starting address of the program section. The argument *value* must be a power of 2. The system extends the size of the previous program section to accommodate the new starting address for the specific section. When you have entered the complete LINK command, the system prompts you for the name of the section whose starting address you need to modify. The prompt is:

```
Boundary section?
```

Respond with the appropriate program section name and terminate your response with a carriage return.

/DEBUG[:filespec] Use this option to link ODT (on-line debugging technique, described in Chapter 21) with your program to help you debug it. If you supply the name of another debugging program, the system links the debugger you specify with your program. The system links the debugger low in memory relative to your program.

/EXECUTE[:filespec] Use this option to specify a file name or device for the executable file. Because the LINK command creates executable files by default, the following two commands have the same meaning:

```
.LINK MYPROG

.LINK/EXECUTE MYPROG
```

Both commands link MYPROG.OBJ and produce MYPROG.SAV as a result. The /EXECUTE option has different meanings when it follows the command and when it follows the file specification. The following command creates an executable file called PROG1.SAV on device RK1:.

```
.LINK/EXECUTE:RK1:  PROG1,PROG2
```

The next command creates an executable file called MYPROG.SAV on device DK:.

```
LINK RTN1,RTN2,MYPROG/EXECUTE
```

/NOEXECUTE Use this option to suppress creation of an executable file.

/EXTEND:n This option allows you to extend a program section to a specific octal value *n*. The resultant program section size is equal to or greater than the value you specify, depending on the space the object code requires. When you have entered the complete LINK command, the system prompts you for the name of the program section you need to extend. The prompt is:

```
Extend section?
```

Respond with the appropriate program section name, and terminate your response with a carriage return.

/FILL:n Use this option to initialize unused locations in the load module and place a specific octal value *n* in those locations. Note that the linker automatically initializes to 0 unused locations in the load module; use this option to place another value in those locations. This option can be useful in eliminating random results that occur when a program references uninitialized memory by mistake. It can also help you to determine which locations have been modified by the program and which remain unchanged.

/FOREGROUND[:stacksize] This option produces an executable file in relocatable (.REL) format for use as a foreground job under the FB or XM monitor. You cannot use .REL files under the single-job system. This option assigns the default file type .REL to the executable file. The argument *stacksize* represents the number of bytes of stack space to allocate for the foreground job. The value you supply is interpreted as an octal number; specify an even number. Follow *n* with a decimal point (*n.*) to represent a decimal number. The default value is 128 (decimal) (or 200 octal) bytes of stack space. DIGITAL recommends that you allocate 256. bytes of stack space when linking a FORTRAN program to run in the foreground.

/INCLUDE This option lets you take global symbols from any library and include them in the linked memory image. When you have entered the complete LINK command, the system prompts you for a list of global symbols to include in the load module. The prompt is:

```
Library search?
```

Respond by typing the global symbols to be included in the load module. Type a carriage return after each global symbol. Type a carriage return on a line by itself to terminate the list. This option forces modules that are not called by other modules to be loaded from the library.

/LDA This option produces an executable file in LDA format. The LDA-format file can be output to any device, including those that are not block-replaceable, such as the paper tape punch or cassette. The default file type .LDA is assigned by /LDA to the executable file. This option is useful for files that you need to load with the Absolute Binary Loader.

/LIBRARY This option is the same as /LINKLIBRARY. It is included here only for system compatibility.

/LINKLIBRARY:filespec You can use this option to include the library file you specify as an object module library in the linking operation. Because the system automatically recognizes library files in the linking operation you do not normally need this option; it is provided for compatibility with the EXECUTE command.

/MAP[:filespec] You must specify this option to produce a load map listing. The /MAP option has different meanings depending on where you put it in the command line.

If you specify /MAP without a filespec in the list of options that immediately follows the command name, the system generates a listing that prints on the line printer. If you follow /MAP with a device name, the system creates a

map file on that device. If the device is a file-structured device, the system stores the listing file on that device, assigning it the same name as the first input file and a .MAP file type. The following command produces a load map on the terminal.

```
.LINK/MAP:TT:  MYPROG
```

The next command creates a map listing file called MYPROG.MAP on RK3:.

```
.LINK/MAP:RK3:  MYPROG
```

If the /MAP option contains a name and file type to override the default of .MAP, the system generates a listing with that name. The following command, for example, links PROG1 and PROG2, producing a map listing file called MAP.OUT on device DK:.

```
.LINK/MAP:MAP.OUT PROG1,PROG2
```

Another way to specify /MAP is to type it after the file specification to which it applies. To link a file and produce a map listing file with the same name, use a command similar to this one.

```
.LINK PROG1,PROG2/EXECUTE/MAP
```

The command shown above links PROG1 and PROG2, producing files PROG2.SAV and PROG2.MAP. If you specify a file name on a /MAP option following a file specification in the command line, it has the same meaning as when it follows the command.

/PROMPT Use this option to enter additional lines of input. The system continues to accept lines of linker input until you enter two slashes (//). Chapter 11 describes the commands you can enter directly to the linker. When you use the /PROMPT option, note that successive lines of input must conform to CSI conventions (see Chapter 6, Command String Interpreter). The example that follows uses the /PROMPT option to create an overlay structure for the program COSINE.MAC:

```
.LINK/PROMPT COSINE
*TAN/O:1
*COS1/O:1
*SIN3/O:2
*LML3/O:2//
```

The /PROMPT option also gives you a convenient way to create an overlaid program from an indirect file. The file ANTON.COM contains these lines:

```
A/PROMPT
SUB1/O:1
SUB2/O:1
SUB3,SUB4/O:1
//
```

The following command produces an executable file, DK:A.SAV, and a link map on the printer.

```
.LINK/MAP  @ANTON
```

/ROUND:n This option rounds up the section you specify so that the size of the root segment is a whole number of the value *n* you supply. The argument *n* must be a power of 2. When you have entered the complete LINK command, the system prompts you for the name of the section that you need to round. The prompt is:

```
Round section?
```

Respond with the appropriate program section name, and terminate your response with a carriage return.

/RUN Use this option to initiate execution of the resultant .SAV file. This option is valid for background jobs only. Do not use /RUN with any option that requires a response from the terminal.

/SLOWLY This option instructs the system to allow the largest possible memory area for the link symbol table at the expense of making the link process slower. Use this option only if an attempt to link a program failed because of symbol table overflow.

/STACK[:n] This option lets you modify the stack address, location 42, which is the address that contains the value for the stack pointer. When your program executes, the monitor sets the stack pointer (SP) to the contents of location 42. The argument *n* is an even, unsigned, six-digit, octal number that defines the stack address. When you have entered the complete LINK command, the system prints the following prompt message if you did not already specify a value for *n*:

```
Stack symbol?
```

Respond with the global symbol whose value is the stack address. You cannot specify a number at this point. Terminate your response with a carriage return. If you specify a nonexistent symbol, the system prints an error message. It then sets the stack address to 1000 (for memory image files) or to the bottom address if you used /BOTTOM.

/SYMBOLTABLE[:filespec] When you use this option, the linker creates a file that contains symbol definitions for all the global symbols in the load module. Enter the symbol table file specification as the third output specification in the LINK command line. If you do not specify a file name, the linker uses the name of the first input file and assigns a .STB file type. By default, the system does not create a symbol table file.

The following example creates the symbol table file BTAN.STB.

```
.LINK    BOBJ,BOBJ2       AOBJ,BOBJ/SYMBOLTABLE:BTAN
```

/TOP:value Use this option to specify the highest address to be used by the relocatable code in the load module. The argument *value* represents an unsigned, even octal number.

/TRANSFER[:n] The transfer address is the address at which a program starts when you initiate execution with R, RUN, FRUN, or SRUN. The /TRANSFER option lets you specify the start address of the load module.

The argument *n* is an even, unsigned, six-digit, octal number that defines the transfer address. When you have entered the complete LINK command, the system prints the following prompt message if you did not already specify a value for *n*:

```
Transfer symbol?
```

Respond with the global symbol whose value is the transfer address. You cannot specify a number at this point. Terminate your response with a carriage return. If you specify a nonexistent symbol, an error message prints and the linker sets the transfer address to 1 so that the system cannot execute the program. If the transfer address you specify is odd, the program does not execute after loading, and control returns to the monitor.

/WIDE Use this option with /MAP to produce a wide load map listing. Normally, the listing is wide enough for three Global Value columns, which is suitable for paper with 72 or 80 columns. The /WIDE option produces a listing that is six Global Value columns wide, which is equivalent to 132 columns.

Table 4–9 lists the sequence in which the system prompts you for additional information when you combine LINK options.

/XM When you use this option, you enable special .SETTOP and .LIMIT features provided in the XM monitor. This option allows a virtual job to map a scratch region in extended memory with the .SETTOP programmed request. See the *RT–11 Programmer's Reference Manual* and the *RT–11 Software Support Manual*, for more details on these special features. Do not use with /FOREGROUND.

If you want to create an extended memory overlay structure for your program, use the /PROMPT option. You can then specify on subsequent lines the overlay structure using the LINK /V option (see Chapter 11 of this manual). Note that when you use /V to create an overlay structure, the linker automatically enables the special .SETTOP and .LIMIT features.

LOAD

The LOAD command loads a device handler into memory for use with foreground, background, or system jobs, or BATCH.

```
LOAD (SP) device [=jobname] [,...device [=jobname]]
```

In the command syntax shown above, *device* represents the device handler to be made resident; *jobname* assigns the device handler to the background job if it has the value B, or to the foreground or system job if it has the value F. The *jobname* specification is invalid with the SJ monitor. Under a monitor that has system job support, *jobname* can be the logical job name of a system job.

The LOAD command helps control system execution by bringing a device handler into memory and optionally allocating the device to a job. The system allocates memory for the handler as needed. Before you use a device in a foreground program with the FB monitor, or any device at all with the XM monitor, you must first load the device handler. A device can be owned exclusively by either the foreground, background, or system job. (Note that BATCH, if running, is considered to be a background job under the FB and XM monitors.) This exclusive ownership prevents the input and output of two different jobs from being intermixed on the same non-file-structured device. In the following example, magtape belongs to the background job, while DECtape is available for use by either the background, foreground, or system job; the line printer is owned by the foreground job. All three handlers are made resident in memory.

```
.LOAD DT:,MT:=B,LP:=F
```

For a monitor with system job support, the following example reserves the line printer for the system job QUEUE.

```
.LOAD LP:=QUEUE
```

Different units of the same random-access device controller can be owned by different jobs. Thus, for example, DT1: can belong to the background job, while DT5: can belong to the foreground or system job. If no ownership is indicated, the device is available for public use.

NOTE

If you use the LOAD command to load a non-file-structured device handler, and assign ownership of that handler to a job, all units of that particular device become assigned to that job. This means no other job can use any unit of that particular device.

To change ownership of a device, use another LOAD command. It is not necessary to first unload the device. For example, if the line printer has been loaded into memory and assigned to the foreground job as in the example

above, the following command reassigns it to the background job without unloading the handler first.

```
.LOAD LP:=B
```

Note, however, that if you interrupt an operation that involves magtape or cassette, you must unload (with the UNLOAD command) then load the appropriate device handler (MM, MT, MS, or CT). When using the MT handler with the FB monitor, this restriction does not apply.

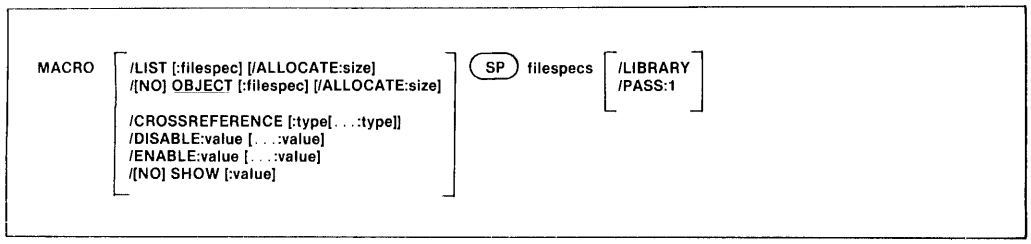
You cannot assign ownership of the system unit (the unit you bootstrapped) of a system device, and any attempt to do so is ignored. You can, however, assign ownership of other units of the same type as the system device. LOAD is valid for use with logical names. For example:

```
.ASSIGN RK: XY
.LOAD XY:=F
```

If you are using a diskette, loading the necessary device handlers into memory can improve system performance significantly, since no handlers need to be loaded dynamically from the diskette. Use the SHOW command to display on the terminal the status of device handlers and device ownership.

MACRO

The MACRO command invokes the MACRO assembler to assemble one or more source files.



In the command syntax shown above, *filespecs* represents one or more files to be included in the assembly. If you omit a file type for an input file, the system assumes .MAC. Output default file types are .LST for listing files and .OBJ for object files.

To assemble multiple source files into a single object file, separate the files by plus (+) signs in the command line. Unless you specify otherwise, the system creates an object file with the same name as the first input file and gives it an .OBJ file type. To assemble multiple files in independent assemblies, separate the files by commas (,) in the command line. This generates a corresponding object file for each set of input files.

Language options are position-dependent — that is, they have different meanings depending on where you place them in the command line. Options that qualify a command name apply across the entire command string. Options that follow a file specification apply only to the file (or group of files separated by plus signs) that they follow in the command string.

You can enter the MACRO command as one line, or you can rely on the system to prompt you for information. The MACRO command prompt is *Files?* for the input specification. The system prints on the terminal the number of errors MACRO detects during an assembly, as this printout shows:

```
.MACRO/CROSSREFERENCE PROG1+PROG2/LIST/OBJECT
ERRORS DETECTED:  0
```

Chapter 10 and the *PDP–11 MACRO Language Reference Manual* contain more detailed information about using MACRO. The options you can use with the MACRO command follow.

/ALLOCATE:size Use this option with /LIST or /OBJECT to reserve space on the device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 32767. A value of –1 is a special case that creates the largest file possible on the device.

/CROSSREFERENCE[:type[...:type] Use this option to generate a symbol cross-reference section in the listing. This information is useful for program maintenance and debugging. Note that the system does not generate a

listing by default. You must also specify /LIST in the command line to get a cross-reference listing. The argument *type* represents a one-character code that indicates which sections of the cross-reference listing the assembler should include. Table 4–10 summarizes the arguments and their meaning.

Table 4–10: Cross-reference Sections

Argument	Section Type
S	User-defined symbols
R	Register symbols
M	Macro symbolic names
P	Permanent symbols (instructions, directives)
C	Control sections (.CSECT symbolic names)
E	Error codes
no argument	Equivalent to :S:M:E

/DISABLE:value[...:value] Use this option to specify a MACRO .DSABL directive. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all legal values. Table 4–11 summarizes the arguments and their meaning.

Table 4–11: .DSABL and .ENABL Directive Summary

Argument	Default	Enables or Disables
ABS	disable	Absolute binary output
AMA	disable	Assembles all absolute addresses as relative addresses
CDR	disable	Treats source columns 73 and greater as comments
FPT	disable	Floating-point truncation
GBL	disable	Treats undefined symbols as globals
LC	disable	Accepts lower case ASCII input
LSB	disable	Local symbol block
PNC	enable	Binary output
REG	enable	Mnemonic definitions of registers

/ENABLE:value[...:value] Use this option to specify a MACRO .ENABL directive. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all legal values. Table 4–11 summarizes the arguments and their meaning.

/LIBRARY This option identifies the file it qualifies as a library file; use it only after a library file specification in the command line. The MACRO assembler looks first to the library file or files you specify and then to the system library, SYSMAC.SML, to satisfy references (made with the .MCALL directive) from MACRO programs. In the example below, the command string includes two user libraries.

```
.MACRO MYLIB1/LIBRARY+A+MYLIB2/LIBRARY+B
```

When MACRO assembles file A, it looks first to the library, MYLIB1.MAC, and then to SYSMAC.SML to satisfy .MCALL references. When it assem-

bles file B, MACRO searches MYLIB2.MAC, MYLIB1.MAC, and then SYSMAC.SML, in that order, to satisfy references.

/LIST[:filespec] You must specify this option to produce a MACRO assembly listing. The /LIST option has different meanings depending on where you place it in the command line.

The /LIST option produces a listing on the line printer when /LIST follows the command name. For example, the following command line produces a line printer listing after compiling a MACRO source file:

```
.MACRO/LIST MYPROG<RET>
```

When the /LIST option follows the file specification, it produces a listing file. For example, the following command line produces the listing file DK:MYPROG.LST after compiling a MACRO source file:

```
MACRO MYPROG/LIST<RET>
```

If you specify /LIST without a file specification in the list of options that immediately follows the command name, the MACRO assembler generates a listing that prints on the line printer. If you follow /LIST with a device name, the system creates a listing file on that device. If the device is a file-structured device, the system stores the listing file on that device, assigning it the same name as the input file and a .LST file type. The following command produces a listing on the terminal.

```
.MACRO/LIST:TT: A
```

The next command creates a listing file called A.LST on RK3:.

```
.MACRO/LIST:RK3: A
```

If the /LIST option contains a name and file type to override the default of .LST, the system generates a listing file with that name. The following command for example, assembles A.MAC and B.MAC together, producing files A.OBJ and FILE1.OUT on device DK:.

```
.MACRO/LIST:FILE1.OUT A+B
```

You cannot use a command like the next one. In this example, the second listing file would replace the first one and cause an error.

```
.MACRO/LIST:FILE2 A,B
```

Another way to specify /LIST is to type it after the file specification to which it applies. To produce a listing file with the same name as a particular input file, you can use a command similar to this one:

```
.MACRO A+B/LIST:RK3:
```

The above command assembles A.MAC and B.MAC, producing files DK:A.OBJ and RK3:B.LST. If you specify a file name on a /LIST option following a file specification in the command line, it has the same meaning as

when it follows the command. The following two commands have the same results:

```
.MACRO A/LIST:B

.MACRO/LIST:B A
```

Both commands generate output files A.OBJ and B.LST.

Remember that file options apply only to the file (or group of files that are separated by plus signs) they follow in the command string. For example:

```
.MACRO A/LIST,B
```

This command assembles A.MAC, producing A.OBJ and A.LST. It also assembles B.MAC, producing B.OBJ. However, it does not produce any listing file for the assembly of B.MAC.

/OBJECT[:filespec] Use this option to specify a file name or device for the object file. Because MACRO creates object files by default, the following two commands have the same meaning:

```
.MACRO A

.MACRO/OBJECT A
```

Both commands assemble A.MAC and produce A.OBJ as output. The /OBJECT option functions like the /LIST option; it can be either a command or a file qualifier.

As a command option, /OBJECT applies across the entire command string. The following command, for example, assembles A.MAC and B.MAC separately, creating object files A.OBJ and B.OBJ on RK1:.

```
.MACRO/OBJECT:RK1: A,B
```

Use /OBJECT as a file option to create an object file with a specific name or destination. The following command assembles A.MAC and B.MAC together, creating files B.LST and B.OBJ.

```
.MACRO A+B/LIST/OBJECT
```

/NOOBJECT Use this option to suppress creation of an object file. As a command option, /NOOBJECT suppresses all object files; as a file option, it suppresses only the object file produced by the related input files. In this command, for example, the system assembles A.MAC and B.MAC together, producing files A.OBJ and B.LST. It also assembles C.MAC and produces C.LST, but does not produce C.OBJ.

```
.MACRO A+B/LIST,C/NOOBJECT/LIST
```

/PASS:1 Use this option on a prefix macro file to process that file during pass 1 of the assembly only. This option is useful when you assemble a source program together with a prefix file that contains only macro definitions, since these definitions do not need to be redefined in pass 2 of the

assembly. The following command assembles a prefix file and a source file together, producing files PROG1.OBJ and PROG1.LST.

```
.MACRO PREFIX.MAC/PASS:1+PROG1/LIST/OBJECT
```

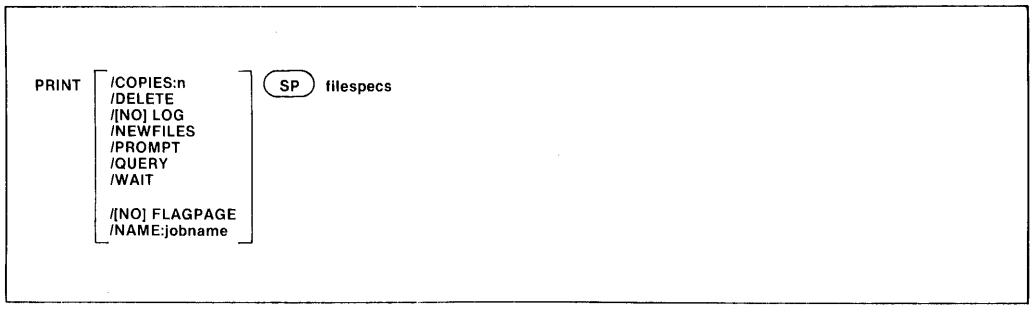
/SHOW:value Use this option to specify any MACRO .LIST directive. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives. Table 4–12 summarizes the arguments and their meaning. Note that you must explicitly request a listing file with the /LIST option.

Table 4–12: .LIST and .NLIST Directive Summary

Argument	Default	Controls
SEQ	list	Source line sequence numbers
LOC	list	Location counter
BIN	list	Generated binary code
BEX	list	Binary extensions
SRC	list	Source code
COM	list	Comments
MD	list	Macro definitions, repeat range expansions
MC	list	Macro calls, repeat range expansions
ME	nolist	Macro expansions
MEB	nolist	Macro expansions binary code
CND	list	Unsatisfied conditionals, .IF and .ENDC statements
LD	nolist	Listing directives with no arguments
TOC	list	Table of Contents
TTM	line printer mode	Output format
SYM	list	Symbol table

/NOSHOW:value Use this option to specify any MACRO .NLIST directive. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives. Table 4–12 summarizes the valid arguments and their meaning. Note that you must explicitly request a listing file with the /LIST option.

The PRINT command lists the contents of one or more files on the line printer.



In the command syntax illustrated above, *filespecs* represents the file or files to be printed. You can explicitly specify up to six files as input to the PRINT command. The system prints the files in the order in which you specify them in the command line. You can also use wildcards in the file specification. In this case, the system prints the files in the same order that they occur in the directory of the specified device. If you specify more than one file, separate the files by commas. If you omit the file type for a file specification, the system assumes .LST. You can specify the entire command on one line, or you can rely on the system to prompt you for information. The PRINT command prompt is *Files?*.

If you are running QUEUE as either a foreground or system job, many of the PRINT commands are executed by this program, therefore, the keyboard monitor may return the dot prompt immediately. See Chapter 20, Queue Package, for more information. If QUEUE is not running, some PRINT options are invalid (as noted). Likewise, some PRINT options are invalid if QUEUE is running. You should use the LOAD command to assign ownership of a non-file structured device to QUEUE so that another job and QUEUE will not intermix output on that device.

The PRINT command options follow; they include command examples.

/COPIES:n Use this option to print more than one copy of the file. The meaningful range of values for the decimal argument *n* is from 1 to 32 (1 is the default). This option must appear immediately after the PRINT command, and not after the file specification. The following command, for example, prints three copies of the file REPORT.LST on the line printer.

```
.PRINT/COPIES:3 REPORT
```

/DELETE Use this option to delete a file after it lists on the line printer. This option must appear following the command in the command line. The PRINT/DELETE operation does not ask you for confirmation before it executes. You must use /QUERY for this function. The following example prints PROG1.BAS on the line printer, then deletes it from DX1:.

```
.PRINT/DELETE DX1:PROG1.BAS
```

/FLAGPAGE:n Use this option if you want banner pages for each file being printed, where *n* represents the number of banner pages you want for each file. This option is valid only if you are running QUEUE. If you specify more than one file to be printed, QUEUE prints a banner page for each file.

The banner page that QUEUE creates consists of a page showing the file name in large, block letters. The banner page also includes a trailer that lists the job name, the date and time the job was output, the copy number and number of copies in the job, and the input file specification.

NOTE

If you use the PRINT command to output files, and QUEUE is running, you may get banner pages even when you do not specify /FLAGPAGE. This condition is due to a default value you can set when you run QUEMAN, the background job that serves as an interface between you and QUEUE. The QUEMAN /P option sets the default number of banner pages for output jobs, so that each time you output a job, you get banner pages. This condition remains in effect until you reset it with the QUEMAN /P option. For more information on QUEMAN and the /P option, see Chapter 20, Queue Package.

The following example prints three banner pages for each file in the command line.

```
.PRINT/FLAGPAGE:3 PROG1.MAC,PROG1.LST,PROG1.STB
```

/NOFLAGPAGE Use this option if you do not want any banner pages printed for each of the files in the job you want printed. Use this option only if you are running QUEUE. This option is useful if you have previously set QUEMAN's /P option to create banner pages each time a job is output (see note above). The default setting is /NOFLAGPAGE unless you specify otherwise with the QUEMAN /P option.

/LOG This option lists on the terminal the names of the files that are printed by the current command. Normally, the system prints a log only if there is a wildcard in the file specification. If you specify /QUERY, the query messages replace the log, unless you specifically type /LOG/QUERY in the command line. The following example shows a PRINT command and the resulting log.

```
.PRINT/LOG/DELETE REPORT
 Files copied/deleted:
DK:REPORT.LST to LP:
```

This option is invalid if QUEUE is running.

/NOLOG This option prevents a list of the files copied from typing out on the terminal. You can use this option to suppress the log when you use a wildcard in the file specification. This option is invalid if QUEUE is running.

/NAME:[dev:]jobname Use this option to specify a job name for the files you want printed. This option is valid only if you are running QUEUE. You can use up to six alphanumeric characters for the job name. If you do not use the /NAME option, the system uses the first input file name as the job name. If you specify a device with the job name, you can send the files to that device, permitting you to send files to any valid RT–11 device. If you send the files to a mass storage volume, the system uses the job name as the file name for the job, assigning a .JOB file type. Note that the handler for the output device must be loaded in memory (see the LOAD command description).

The following example sends JOB5, consisting of FILE1.LST, FILE2.LST, and FILE3.LST, to DX1:.

```
.PRINT/NAME:DX1:JOB5 FILE1,FILE2,FILE3
```

The files from this example reside on DX1: as JOB5.JOB.

/NEWFILES Use this option in the command line if you need to print only those files that have the current date. The following example shows a convenient way to print all new files after a session at the computer.

```
.PRINT/NEWFILES *.LST
 Files copied:
DK:OUTFIL.LST    to LP:
DK:REPORT.LST    to LP:
```

This option is invalid if QUEUE is running.

/PROMPT Use this option to continue a command string onto subsequent lines. This option is valid only if you are running QUEUE. When you use /PROMPT, you can enter file specifications on subsequent lines directly to QUEMAN, described in Chapter 20. Terminate the command with two slashes (//).

The following example uses /PROMPT to print FILE1,FILE2,FILE3,FILE4, and FILE5:

```
.PRINT/PROMPT FILE1
*FILE2, FILE3
*FILE4
*FILE5//
```

/QUERY If you use this option, the system requests confirmation from you before it performs the operation. /QUERY is particularly useful on operations that involve wildcards, when you may not be sure which files the system selected for an operation. Note that if you specify /QUERY in a PRINT command line that also contains a wildcard in the file specification, the confirmation messages that print on the terminal replace the log messages that would normally appear. You must respond to a query message by typing Y (or anything that begins with Y) and a carriage return to initiate execution of a particular operation. The system interprets any other

response to mean NO; it does not perform the specific operation. The following example uses /QUERY.

```
.PRINT/QUERY *.LST
 Files copied:
DK:OUTFIL.LST    to LP:? N
DK:REPORT.LST    to LP:? Y
```

This option is invalid if QUEUE is running.

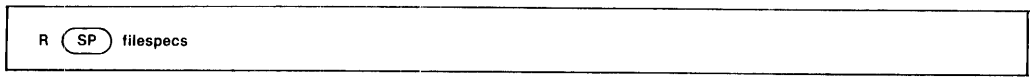
/WAIT This option is useful if you have a single-disk system. When you use this option, the system initiates the PRINT operation, but then pauses and waits for you to mount the volume from which you want the operation to take place. When the system pauses, it prints *Mount input volume in <device>; Continue?*. When the volume is mounted, type Y followed by a carriage return.

The following command line prints ERREX.MAC from RK0:

```
.PRINT/WAIT RK0:ERREX.MAC
Mount input volume in RK0:; Continue?Y
Mount system volume in RK0:; Continue?Y
```

In the case of PRINT, the system prints the file or files you specify before it prints *Mount system volume in <device>; Continue?*. Make sure when you use /WAIT that PIP is on the system volume. This option is invalid if QUEUE is running.

The R command loads a memory image file from the system device into memory and starts execution.



In the command syntax shown above, *filespec* represents the program to be executed. The default file type is .SAV. The only valid device is SY:. The R command is similar to the RUN command except that the file you specify in an R command string must be on the system device (SY:). Use the R command only with background jobs including privileged jobs in XM. (Use FRUN to execute a foreground job under the FB or XM monitor.) The following command loads and executes MYPROG.SAV from device SY:.

```
.R MYPROG
```

The R command is the only monitor command that can execute a background virtual job under the XM monitor. The R command creates a virtual memory partition for the job, creates a region 0 and window 0 definition block, and sets up the user mapping registers.

REENTER

The REENTER command starts the program at its reentry address (the start address minus 2).

```
REENTER
```

The REENTER command accepts no options or arguments. REENTER does not clear or reset any memory areas. Use it to avoid reloading the same program for subsequent execution. You can use REENTER to return to a system program or to any program that allows for a REENTER after the program terminates. You can also use REENTER after you have used two CTRL/Cs to interrupt those programs.

If you issue the REENTER command and it is not valid, the message *?KMON-F-Illegal command* is printed. You must start that program with an R or RUN command.

In the following example the directory program (DIR) lists the directory of DK: on the line printer. Two CTRL/Cs interrupt the listing and return to the monitor. REENTER starts DIR at its reentry address, and DIR prompts for a line of input.

```
.R DIR
*LP:=DK:*.*
^C
^
.
.REENTER
*
```

Note in the example above that using REENTER does not mean that the directory listing continues from where it was interrupted, only that the DIRECTORY program re-commences execution.

REMOVE

The REMOVE command removes a device name from the system tables.

```
REMOVE (SP) device [,...device]
```

In the command syntax shown above, *device* represents the device to be removed from the system tables. The REMOVE command accepts no options. You can enter the REMOVE command on one line, or you can rely on the system to prompt you for information. The REMOVE command prompt is *Device?*.

Using the REMOVE command does not change the monitor disk image; it only modifies the system tables of the monitor currently in core. This allows you to configure a special system for a single session at the computer without having to reconfigure to return to your standard device configuration. Bootstrapping the system device restores the original device configuration. To permanently REMOVE a device, include the REMOVE command in the standard system startup indirect command file.

You cannot remove SY: (the handler for the system device), BA: (the BATCH handler), or TT: (the terminal handler). If you attempt to REMOVE a device that does not exist in the running monitor's system table, the system prints an error message. You can use the INSTALL command to install a new device after using the REMOVE command to remove a device (thus creating a free device slot).

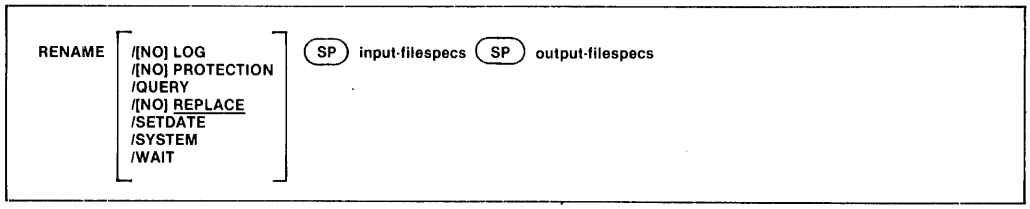
The following command removes the line printer handler and the card reader handler from the system. Note that the colons (:) are optional.

```
.REMOVE  LP:,CR:
```

Use the SHOW command to display on the terminal a list of devices that are currently available on your system.

RENAME

The RENAME command assigns a new name to an existing file.



In the command syntax illustrated above, *input-filespecs* represents the files to be renamed, and *output-filespec* represents the new name. You can specify up to six input files, but only one output file. Note that the device specification must be the same for input and output; you cannot rename a file from one device to another. If a file exists with the same name and file type as the output file you specify, the system deletes the existing file unless you use the /NOREPLACE option to prevent this.

So that you do not rename system (.SYS) files by accident when you use a wildcard in the file specification, the system requires you to use the /SYSTEM option when you need to rename system files. To rename files that cover bad blocks (.BAD files), you must explicitly give the file name and file type of the specified .BAD file. Since .BAD files cover bad blocks on a device, you usually do not need to rename or otherwise manipulate these files.

Note that because of the file protection feature, you cannot execute any RENAME operations that result in deleting a protected file. For example, you cannot rename a file to the name of a protected file that already exists on the same volume.

The options you can use with the RENAME command follow.

/LOG This option lists on the terminal the files that were renamed by the current command. Normally, the system prints a log only if there is a wildcard in the file specification. If you specify /QUERY, the query messages replace the log (unless you specifically type /LOG/QUERY in the command line).

This example demonstrates logging.

```
.RENAME DX0:(A*.MAC  *.FOR)
 Files renamed:
DX0:ABC.MAC     to DX0:ABC.FOR
DX0:AAF.MAC     to DX0:AAF.FOR
```

/NOLOG This option prevents a list of the files that are renamed from appearing on the terminal.

/NEWFILES Use this option in the command line if you want to rename only those files that have the current date. This is a convenient way to access all new files after a session at the computer.

/PROTECTION Use this option to give a file *protected* status so that it cannot be deleted until you disable that status. Note that if a file is protected, you cannot delete it implicitly. For example, you cannot perform any operations on a file that result in deleting a protected file. You can change a protected file's name, but not its protected status, unless you also use the /NOPROTECTION option.

/NOPROTECTION Use this option to enable a file for deletion. This option disables a file's *protected* status.

/QUERY If you use this option, the system requests confirmation from you before it performs the operation. /QUERY is particularly useful on operations that involve wildcards, when you may not be sure which files the system selected for the operation. Using the /QUERY option also provides a quick way of performing operations on several files. For example, renaming several files is easier if you use /QUERY. You can then specify Y for each file you want renamed, as the following example shows.

```
.RENAME/QUERY  *.BAK  *.MAC
 Files renamed:
DK:PROG1.BAK    to DK:PROG1.MAC ? Y
DK:PROG2.BAK    to DK:PROG2.MAC ? Y
DK:PROG6.BAK    to DK:PROG6.MAC ? Y
DK:LML8A.BAK    to DK:LML8A.MAC ?
DK:LML9 .BAK    to DK:LML9 .MAC ? Y
```

Note that if you specify /QUERY in a command line that also contains a wildcard in the file specification, the confirmation messages that print on the terminal replace the log messages that would normally appear. You must respond to a query message by typing Y (or anything that begins with Y) and a carriage return to initiate execution of a particular operation. The system interprets any other response to mean NO; it does not perform the specific operation. The following example demonstrates querying.

```
.RENAME/QUERY DX0:(PIP1.SAV PIP.SAV)
 Files renamed:
DX0:PIP1.SAV    to DX0:PIP.SAV    ? Y
```

/REPLACE This is the default mode of operation for the RENAME command. If a file exists with the same name as the file you specify for output, the system deletes that duplicate file when it performs the rename operation.

/NOREPLACE This option prevents execution of the rename operation if a file with the same name as the output file you specify already exists on the same device. The following example uses /NOREPLACE. In this case, the output file already existed and no action occurs.

```
.RENAME/NOREPLACE DX0:TEST.SAV DX0:DUP.SAV
?PIP-W-Output file found, no operation performed DX0:TEST.SAV
```

/SETDATE This option causes the system to put the current date on all files it renames, unless the current system date is not set. Normally, the sys-

tem preserves the existing file creation date when it renames a file. The following example renames files and changes their dates.

```
.RENAME/SETDATE DX0:(*.FOR *.OLD)
 Files renamed:
DX0:ABC.FOR     to DX0:ABC.OLD
DX0:AAF.FOR     to DX0:AAF.OLD
DX0:MERGE.FOR   to DX0:MERGE.OLD
```

/SYSTEM Use this option if you need to rename system (.SYS) files. If you omit this option, the system files are excluded from the rename operation and a message is printed on the terminal to remind you of this. This example renames MM.SYS to MX.SYS.

```
.RENAME/SYSTEM DX0:MM.SYS DX0:MX.SYS
```

/WAIT This option is useful if you have a single-disk system. When you use this option, the system initiates the RENAME operation, but then pauses and waits for you to mount the input volume on which the operation is to take place. When the system pauses, it prints *Mount input volume in <device>; Continue?* where *<device>* represents the device into which you mount the volume. When the volume is mounted, type Y followed by a carriage return.

The following command line renames PRIAM.TXT to NESTOR.TXT. PRIAM.TXT is on an RK05 disk.

```
.RENAME/WAIT/NOLOG RK0:PRIAM.TXT NESTOR.TXT
Mount input volume in RK0:; Continue?Y
Mount system volume in RK0:; Continue?Y
```

RESET

The RESET command resets several background system tables and does a general clean-up of the background area.

```
RESET
```

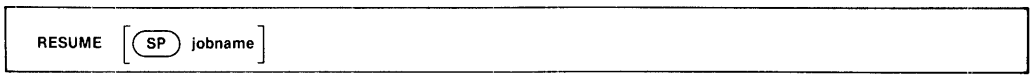
The RESET command accepts no options or arguments.

It causes the system to purge all open input/output channels, initialize the user program memory area, and release any device handlers that were not explicitly made resident with the LOAD command. It also disables CTRL/O, clears locations 40–53, and resets the KMON (keyboard monitor) stack pointer. Use RESET before you execute a program if a device or the monitor needs reinitialization, or when you need to discard the results of previously issued GET commands. The RESET command has no effect on the foreground or system job. The following example uses the RESET command before running a program.

```
.RESET
.R MYPROG
```

RESUME

The RESUME command continues execution of the foreground or system job from the point at which a SUSPEND command was issued.



If you have system job support enabled on your monitor, the RESUME command must be followed by the name of the foreground or system job you wish to resume. (The RESUME command accepts logical job names.) If you do not have system job support enabled on your monitor, do not include the name of the foreground job you wish to resume. When you issue the RESUME command, the foreground or system job enters any completion routines that were scheduled while the job was suspended. Note that RESUME is valid only with the FB and XM monitors. The following command resumes execution of the foreground job that is currently suspended.

```
.RESUME
```

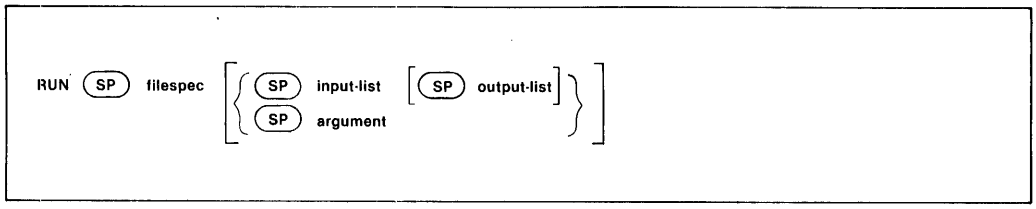
The next command resumes execution of the system job, QUEUE.SYS, that is currently suspended.

```
.RESUME QUEUE
```

You can also use the RESUME command to start a foreground job that you loaded with FRUN using /PAUSE. Likewise, you can use RESUME to start a system job that you loaded with SRUN using /PAUSE.

RUN

The RUN command loads a memory image file into memory and starts execution.



In the command syntax illustrated above, *filespec* represents the program to be executed. The system assumes a .SAV file type for the executable file, which can reside on any RT–11 block-replaceable device. The default device is DK:. The RUN command automatically loads the device handler for the device you specify if it is not already resident. This eliminates the need to explicitly load a device handler when you run an overlaid program from a device other than the system device. The RUN command executes only those programs that have been linked to run as background jobs. (Use FRUN to execute foreground jobs under the FB or XM monitor.)

RUN is a combination of the GET and START commands. First it loads a memory image file from a storage device into memory. Then it begins execution at the program's transfer address. You can use RUN to execute a privileged job under the XM monitor the same way you execute any other background job in FB or SJ. However, a virtual job in XM requires special preparation for execution. You must use the R command to execute a background virtual job. The R command creates a virtual memory partition for the job, creates a region 0 and window 0 definition block for the partition, and sets up the user mapping registers. The following command, for example, executes MYPROG.SAV, which is stored on device DX1:.

```
.RUN DX1:MYPROG
```

You can also pass an argument in the RUN command to the program, or specify a list of input and output. This allows you to specify a line of input for a user program or for a system utility program (which accepts file specifications in the special syntax described in Chapter 6). The system automatically converts the input list and the output list you specify into a format that the Command String Interpreter accepts. For example, to execute the directory program (DIR) and obtain a complete listing of the directory of DX1: on the printer, you can use the following command.

```
.RUN DIR DX1:*.* LP:/E
.
```

RUN

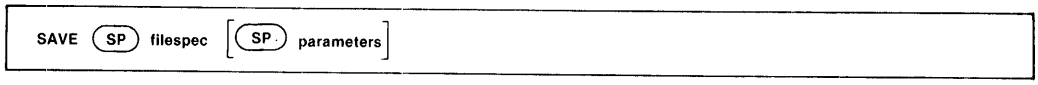
This command has the same effect as the following lines.

```
.RUN DIR
*LP:/E=DX1:*.*
*^C
.
```

Note that when you use either an argument or an input list and output list with RUN, control returns to the monitor when the program completes.

SAVE

The SAVE command writes memory areas in memory image format to the file and device that you specify.



In the command syntax shown above, *filespec* represents the file to be saved on a block-replaceable device. If you do not specify a file type, the system uses .SAV. The parameters represent memory locations to be saved.

Parameters are of the form:

address[-address(2)][,address(3)[-address(n)]]

where:

address is an octal value representing a specific block of memory locations to be saved. If you specify more than one address, each address must be higher than the previous one

RT–11 transfers memory in 256-word blocks, beginning on boundaries that are multiples of 256 (decimal). If the locations you specify make a block that is less than 256 words, the system saves additional words to make a 256-word block

The system saves memory from location 0 to the highest memory address specified by the parameter list or to the program high limit (location 50 in the system communication area). Initially, the system gives the start address and the Job Status Word the default value 0 and sets the stack to 1000. If you want to change these or any of the following addresses, you can use the Deposit command to alter them and the SAVE command to save the correct areas.

Area	Location
Start address	40
Stack	42
JSW	44
USR address	46
High address	50
Fill characters	56

If you change the values of the addresses, it is your responsibility to reset them to their default values. For more information concerning these addresses refer to the *RT–11 Programmer's Reference Manual.* Note that the SAVE command does not write the overlay segments of programs; it saves only the root segment. You cannot use the SAVE command for foreground or virtual jobs.

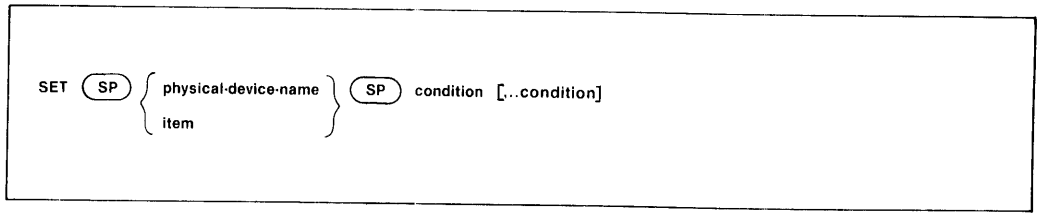
The following command saves locations 10000 through 11777, and 14000 through 14777. It stores the contents of these locations in the file FILE1.SAV on device DK:.

```
.SAVE FILE1 10000-11000,14000-14100
```

The next example sets the reenter bit in the JSW and saves locations 1000 through 5777 in file PRAM.SAV on device SY:.

```
.D 44=2000
.SAVE SY:PRAM 1000-5777
```

The SET command changes device handler characteristics and certain system configuration parameters.



In the command syntax illustrated above, *physical-device-name* represents the device handler whose characteristics you need to modify.

See Table 3–1 in this manual for a list of the standard RT–11 permanent device names. The argument *item* represents a system parameter that you need to modify. The system items you can change include error handling (SET ERROR) and wildcard handling (SET WILD). Table 4–13 lists the devices and items you can modify, as well as the valid conditions for these devices and items. If you set more than one condition for a device, separate the conditions with commas. With the exception of the SET TT, SET USR, and SET item commands, the SET command locates the file SY:device.SYS and permanently modifies it. The SET commands are valid for all three RT–11 monitors unless otherwise specified. They permanently modify the device handlers (except where noted); this means that the conditions remain set even across a reboot. For those SET commands that do not permanently modify the device handlers, the conditions return to the default setting after a reboot. To make these settings appear permanent, include the appropriate SET commands in your system's startup indirect command file (see Section 4.3.3). The command you enter must be completely valid for the modification to take place. The SET command will modify only the device handler that corresponds to the currently booted monitor. For example, if you issue the SET command while running under the XM monitor, any device handlers modified will be of the form %%X.SYS.

NOTE

If a handler (except for TT:) is already loaded when you issue a SET command for it, you must unload the handler and load a fresh copy from the system device for the modification to have an effect on execution.

The colon (:) after each device name is optional.

Figure 4–2: Format of a 12–bit Binary Number

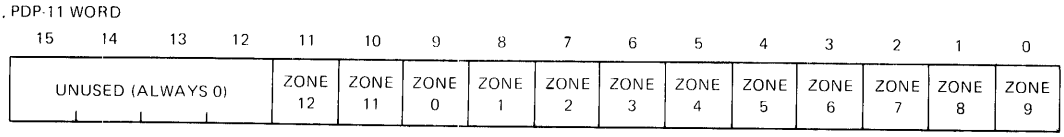


Table 4–13: SET Device Conditions and Modification

Device or Item	Condition	Modification
CR:	CODE = n	Modifies the card reader handler to use either the DEC 026 or DEC 029 card codes. The argument *n* must be either 26 or 29. The default value is 29.
CR:	CRLF	Appends a carriage return/line feed combination to each card image. This is the normal mode.
CR:	NOCRLF	Transfers each card image without appending a carriage return/line feed combination. The default is CRLF.
CR:	HANG	Waits for you to make a correction if the reader is not ready at the start of a transfer. This is the normal mode.
CR:	NOHANG	Generates an immediate error if the device is not ready at the start of a transfer. The handler waits (regardless of how the condition is set) if the reader is not ready at some point during a transfer (that is, the input hopper is empty, but an end-of-file card has not been read). The default is HANG.
CR:	IMAGE	Causes each card column to be stored as a 12-bit binary number, one column per word. The CODE option has no effect in IMAGE mode. Figure 4–2 illustrates the format of the 12-bit binary number. This format allows the system to read binary card images. It is especially useful if you use a special encoding of punch combinations. Mark-sense cards can be read in this mode. The default is NOIMAGE.
CR:	NOIMAGE	Allows the normal translation (as specified by the CODE option) to take place. The system packs data one column per byte. It translates invalid punch combinations into the error character, ASCII backslash (\), which is octal code 134. This is the normal mode.
CR:	TRIM	Removes trailing blanks from each card that the system reads. You should not use TRIM and NOCRLF together because card boundaries become difficult to read. TRIM is the normal mode.
CR:	NOTRIM	Transfers a full 80 characters per card. The default is TRIM.
CT:	RAW	Performs a read-after-write check for every record written. The system retries if an output error occurs. If three retries fail, the system indicates an output error. The default is NORAW.
CT:	NORAW	Writes every record directly without reading it back for verification. This setting significantly increases transfer rates at the risk of increased error rates. This is the normal mode.
DD:	VECTOR = n	Modifies the DECtape II handler to use *n* as the vector address for the first DECtape II controller (*n* is an octal number). This option, and the next three, enable you to set vector and Control and Status Register (CSR) values in the handler itself, without having to modify the handler source code and reassemble. Use these options if you have installed the DECtape II controller(s) at nonstandard addresses.

(continued on next page)

Table 4–13: SET Device Conditions and Modification (Cont.)

Device or Item	Condition	Modification
DD:	CSR = n	Modifies the DECtape II handler to use *n* as the CSR address for the first DECtape II controller. When you use this option, the system prints a message that indicates where to patch the DECtape II handler bootstrap, if you want to use a DECtape II as your system volume.
DD:	VEC2 = n	Modifies the DECtape II handler to use *n* as the vector for the second DECtape II controller. This option is valid only if you create the DECtape II dual controller handler (through system generation).
DD:	CSR2 = n	Modifies the DECtape II handler to use *n* as the CSR address for the second DECtape II controller. This option is valid only if you create the DECtape II dual controller handler (through system generation).
EDIT	EDIT	Invokes the text editor EDIT with the keyboard monitor EDIT command. This is the normal mode. The system returns to this condition after a reboot.
EDIT	KED	Invokes the Keypad Editor (KED). For more information on the Keypad Editor, see the *PDP–11 Keypad Editor User's Guide*. This condition is valid only for VT100 terminals. The system returns to EDIT EDIT after a reboot.
EDIT	K52	Invokes the Keypad Editor (K52); valid if your terminal is a VT52. For more information on the Keypad Editor, see the *PDP–11 Keypad Editor User's Guide*. The system returns to EDIT EDIT after a reboot.
EDIT	TECO	Invokes the text editor TECO with the keyboard monitor EDIT command. The default is EDIT. The system returns to that condition after a reboot.
ERROR	WARNING	Causes indirect command files and keyboard monitor commands to abort if warnings, errors, or severe or fatal errors occur. See SET ERROR ERROR, which is the default setting. Warning error messages contain the -W- characters. The system returns to that condition after a reboot.
ERROR	ERROR	Causes indirect command files and keyboard monitor commands that perform multiple operations (such as EXECUTE, which combines assembling, linking, and running) to abort if errors or severe or fatal errors occur. This setting causes indirect files and keyboard monitor commands to abort on MACRO assembly errors. An example of an error is an undefined symbol in an assembly. An example of a severe error is a device that is write-locked when the system attempts to write to it. If either condition occurs, the indirect command file or keyboard monitor command aborts the next time the monitor gets control of the system. Error error messages contain the -E- characters. This is the normal setting. The system returns to this condition after a reboot.

(continued on next page)

Table 4–13: SET Device Conditions and Modification (Cont.)

Device or Item	Condition	Modification
ERROR	SEVERE	Causes indirect command files and keyboard monitor commands to abort only if severe or fatal errors occur. Severe error messages contain the -F- characters. See SET ERROR ERROR, which is the default setting. The system returns to that condition after a reboot.
ERROR	NONE	Allows indirect command files and keyboard monitor commands to continue to execute even though they contain significant errors. Most monitor fatal errors still cause the indirect command file or keyboard monitor command to abort. Fatal errors that always abort indirect command files contain the -U- characters in the error messages. See SET ERROR ERROR, which is the default setting. The system returns to that condition after a reboot.
LP:	CR	Sends carriage returns to the printer. To allow overstriking on the printer, use this condition for any FORTRAN program that uses formatted input and output. Use CR also for any LS11 or LP05 line printer to prevent loss of the last line in the buffer. LP NOCR is the normal mode.
LP:	NOCR	Prevents the system from sending carriage returns to the printer. This setting produces a significant increase in printing speed on LP11 printers, where the line printer controller causes a line feed to perform the functions of a carriage return. This is the default setting.
LP:	CSR=n	Modifies the line printer handler to use *n* as the Control and Status Register (CSR) address for the line printer controller. The value you supply must be an octal word address not less than 160000. This option enables you to set a special CSR value in the line printer handler itself, without having to modify and reassemble the handler source code. Use this option if you have installed the line printer controller at a nonstandard address.
LP:	CTRL	Passes all characters, including nonprinting control characters, to the printer. Use this condition to pass the bell character to the LA180 printing terminal. You can use this mode for LS11 line printers. (Other line printers print a space for a control character.) The default is NOCTRL.
LP:	NOCTRL	Ignores nonprinting control characters. This is the normal mode.
LP:	FORM	Declares that the line printer has hardware form feeds, causing the line printer handler to send form feeds to the controller. When you use this option, the line printer handler sends the form feed character to the printer each time the handler encounters a form feed. This is the default setting.

(continued on next page)

Table 4–13: SET Device Conditions and Modification (Cont.)

Device or Item	Condition	Modification
LP:	NOFORM	Causes the line printer handler to simulate hardware form feeds by sending one or more line feeds to the printer. When you use this setting, you must also use the LENGTH=n setting and position the paper at the top of a form (that is, at the page perforation) before you start to use the printer. Using the NOFORM condition is useful if you are using a preprinted form that has a nonstandard length. You must use this setting if your printer does not accommodate form feeds. FORM is the default setting.
LP:	FORM0	Issues a form feed before a request to print block 0. This is the normal mode.
LP:	NOFORM0	Turns off FORM0 mode, which is the default.
LP:	HANG	Waits for you to make a correction if the line printer is not ready or is not ready at some point during printing. If you expect output from the line printer and the system does not respond or appears to be idle, check to see if the line printer is powered on and ready to print. This is the normal mode.
LP:	NOHANG	Generates an immediate error if the line printer is not ready. The default is HANG.
LP:	LC	Allows the system to send lower-case characters to the printer. Use this condition if your printer has a lower-case character set. The default is NOLC.
LP:	NOLC	Translates characters in lower case to upper case before printing. This is the normal mode.
LP:	LENGTH=n	Causes the line printer to use *n* as the number of lines per page. The default number of lines per page is 66. Use this option with the NOFORM and SKIP=n settings.
LP:	SKIP=n	Causes the line printer handler to send a form feed to the printer when it comes within n lines of the bottom of a page. Use this setting to prevent the printer from printing over page perforations. The value you supply for *n* should be an integer from 0 to the maximum number of lines on the paper. If you set SKIP=0, the handler sends lines to the printer regardless of the position of the paper. If you have set SKIP to a value other than 0, set SKIP=0 to disable this condition. When you use this setting, you must also use the LENGTH=n setting. The default is SKIP=0.
LP:	TAB	Sends TAB characters to the line printer. The default is NOTAB.
LP:	NOTAB	Expands TAB characters by sending multiple spaces to the line printer. This is the normal mode.
LP:	VECTOR=n	Modifies the line printer handler to use *n* as the vector of the line printer controller. The value you supply for *n* must be an even octal address below 500. This option enables you to set a special vector value in the line printer handler itself, without having to modify the handler source code and reassemble. Use this option if you have installed the line printer controller at a nonstandard address.

(continued on next page)

Table 4–13: SET Device Conditions and Modification (Cont.)

Device or Item	Condition	Modification
LP:	WIDTH=n	Sets the line printer width to *n*, where *n* is a decimal integer between 30 and 255, inclusive. The system ignores any characters that print past column *n*. The default is 132.
LS:	CR	Sends carriage returns to the printer. To allow overstriking on the printer, use this condition for any FORTRAN program that uses formatted input and output. (Use CR also for any LS11 or LP05 line printer to prevent loss of the last line in the buffer.) This is the normal mode.
LS:	NOCR	Prevents the system from sending carriage returns to the printer. This setting may produce a significant increase in printing speed on some line printers. Where the printer controller causes a line feed to perform the functions of a carriage return. The default is CR.
LS:	CSR=n	Modifies the line printer handler to use *n* as the Control and Status Register (CSR) address for the printer controller. The value you supply for *n* must be an octal word address not less than 160000. This option enables you to set a special CSR value in the printer handler itself, without having to modify the handler source code and reassemble. Use this option if you have installed the printer controller at a nonstandard address.
LS:	CTRL	Passes all characters, including nonprinting control characters, to the printer. Use this condition to pass the bell character to the LA180 printing terminal. The default is NOCTRL.
LS:	NOCTRL	Ignores nonprinting control characters. This is the normal mode.
LS:	FORM	Declares that the line printer has hardware form feeds, causing the line printer handler to send form feeds to the controller. When you use this option, the line printer handler sends the form feed character to the printer each time the handler encounters a form feed. This is the default setting.
LS:	NOFORM	Causes the line printer handler to simulate hardware form feeds by sending one or more line feeds to the printer. When you use this setting, you must also use the LENGTH=n setting and position the paper at the top of a form (that is, at the page perforation) before you start to use the printer. Using the NOFORM condition is useful if you are using a preprinted form that has a nonstandard length. You must use this setting if your printer does not accommodate form feeds. FORM is the default setting.
LS:	FORM0	Issues a form feed before a request to print block 0. This is the normal mode.
LS:	NOFORM0	Turns off FORM0 mode. The default is FORM0.
LS:	HANG	Waits for you to make a correction if the line printer is not ready or becomes not ready during printing. If you expect output from the printer and the system does not respond or appears to be idle, check to see if the printer is powered on and ready to print. This is the normal mode.

(continued on next page)

Table 4-13: SET Device Conditions and Modification (Cont.)

Device or Item	Condition	Modification
LS:	NOHANG	Generates an immediate error if the printer is not ready. The default setting is HANG.
LS:	LC	Allows the system to send lower-case characters to the printer. Use this condition if your printer has a lower-case character set. This is the normal mode.
LS:	NOLC	Translates lower-case characters to upper-case before printing. The default is LC.
LS:	LENGTH=n	Causes the printer to use *n* as the number of lines per page. The default number of lines per page is 66. Use this option with the NOFORM and SKIP=n settings.
LS:	SKIP=n	Causes the line printer handler to send a form feed to the printer when it comes within n lines of the bottom of a page. Use this setting to prevent the printer from printing over page perforations. The value you supply for *n* should be an integer from 0 to the maximum number of lines on the paper. If you set SKIP=0, the handler sends lines to the printer regardless of the position of the paper. If you have set SKIP to a value other than 0, set SKIP=0 to disable this condition. When you use this setting, you must also use the LENGTH=n setting. The default is SKIP=0.
LS:	TAB	Sends TAB characters to the printer. The default is NOTAB.
LS:	NOTAB	Expands TABS by sending multiple spaces to the printer. This is the normal mode.
LS:	VECTOR=n	Modifies the printer handler to use *n* as the vector of the line printer controller. The value you supply for *n* must be an even octal address below 500. This option enables you to set a special vector value in the line printer handler itself, without having to modify the handler source code and reassemble. Use this option if you have installed the printer controller at a nonstandard address.
LS:	WIDTH=n	Sets the printer to width *n*, where *n* is a decimal integer between 30 and 255, inclusive. The system ignores any characters that print past column *n*. The default is 132.
MM:	DEFALT=9	Returns to default settings for 9-track tape. The 9-track defaults are: DENSE=809 ODDPAR NODUMP
MM:	DENSE=[800 or 809 or 1600]	Sets density for the 9-track tape handler. Do not alter the density setting within a volume. A density setting of 1600 bits per inch (BPI) automatically sets parity to odd. The valid density settings for 9-track tape are: 800 BPI 1600 BPI
MM:	ODDPAR	Sets parity to odd for 9-track tape. DIGITAL recommends this setting.

(continued on next page)

Table 4-13: SET Device Conditions and Modification (Cont.)

Device or Item	Condition	Modification
MM:	NOODDPAR	Sets parity to even for 9-track tape. DIGITAL does not recommend this setting for normal operation, and provides it only for compatibility with other systems.
MT:	DEFALT=[7 or 9]	Returns to default settings for 7- or 9-track tape. The 7-track defaults are equivalent: DENSE=800 ODDPAR
MT:	DENSE=[200 or 556 or 800 or 807 or 809]	Sets density for 7- or 9-track tape. Specifying 200, 556, or 807 for 7-track tape sets 6-bit mode. Density settings 800 and 809 are equivalent. Specifying either 800 or 809 for 7-track tape sets core dump mode. Settings 800 and 809 are only valid settings for 9-track tape. Thus, the valid density settings are as follows: 7-track: 200 = 200 bpi six-bit mode 556 = 556 bpi six-bit mode 807 = 800 bpi six-bit mode 800 or 809 = 800 bpi core dump mode 9-track: 800 or 809 = 800 bpi
MT:	DUMP	Sets core dump mode for 7-track tape. This is equivalent to setting DENSE = 800 or 809. NOTE These SET command options apply to all units of the magtape controller. Six-bit mode and core dump mode are described in the *RT-11 Software Support Manual.*
MT:	ODDPAR	Sets parity to odd for 7- or 9-track tape. DIGITAL recommends this setting.
MT:	NOODDPAR	Sets parity to even for 7- or 9-track tape. DIGITAL does not recommend this setting for normal operation, and provides it only for compatibility with other systems.
TT:	CONSOL=n	Directs the system to use the terminal whose logical unit number you specify as the console terminal. The terminal whose logical unit number you specify must not be currently attached by the foreground or any system job. To use this setting, you must have a multi-terminal configuration. The system returns to this default after a reboot. You cannot use this setting for a remote line.
TT:	CRLF	Issues a carriage return/line feed combination on the console terminal whenever you attempt to print past the right margin. You can change the margin with the WIDTH command. This is the normal mode. This setting is invalid with a non-multi-terminal SJ monitor. The system returns to this condition after a reboot.
TT:	NOCRLF	Takes no special action at the right margin. This setting is invalid with a non-multi-terminal SJ monitor. The default is CRLF. The system returns to that condition after a reboot.

(continued on next page)

Table 4–13: SET Device Conditions and Modification (Cont.)

Device or Item	Condition	Modification
TT:	FB	Treats CTRL/B and CTRL/F (and CTRL/X in system job monitors) as background and foreground program control characters and does not transmit them to your program. This is the normal mode. This setting is not valid for the SJ monitor. The system returns to this condition after a reboot.
TT:	NOFB	Causes CTRL/B and CTRL/F (and CTRL/X in system job monitors) to have no special meaning. Issue SET TT: NOFB to KMON, which runs as a background job, to disable all communication with the foreground or system job. To enable communication with the foreground job, issue the command SET TT FB. This setting is not valid for the SJ monitor. The default is FB. The system returns to that condition after a reboot.
TT:	FORM	Indicates that the console terminal is capable of executing hardware form feeds. This setting is invalid with a non-multi-terminal SJ monitor.
TT:	NOFORM	Simulates form feeds by generating eight line feeds. This setting is not valid for the non-multi-terminal SJ monitor. This is the normal mode. The system returns to this condition after a reboot.
TT:	HOLD	Enables the Hold Screen mode of operation for the VT50, VT52, and VT61 terminals. The command has no effect on any other terminals, but it can cause a left square bracket ([) to print. This setting is valid for all monitors. NOHOLD is the default setting. The system returns to that condition after a reboot.
TT:	NOHOLD	Disables the Hold Screen mode of operation for the VT50 terminal. The command has no effect on any other terminal, but it can cause a backslash (\) to print. This setting is valid for all monitors. The default is NOHOLD. The system returns to that condition after a reboot.
TT:	PAGE	Treats CTRL/S and CTRL/Q characters as terminal output hold and unhold flags and does not transmit them to your program. You must use this setting if you are using a VT100 terminal. This setting is not valid for the non-multi-terminal SJ monitor. This is the normal mode. The system returns to this condition after a reboot.
TT:	NOPAGE	Causes CTRL/S and CTRL/Q to have no special meaning. This setting is not valid for the non-multi-terminal SJ monitor. The default is PAGE. The system returns to that condition after a reboot.
TT:	QUIET	Prevents the system from echoing lines from indirect files. The default is NOQUIET. The system returns to that condition after a reboot.
TT:	NOQUIET	Echoes lines from indirect files. This is the default mode. The system returns to this condition after a reboot.

(continued on next page)

Table 4–13: SET Device Conditions and Modification (Cont.)

Device or Item	Condition	Modification
TT:	SCOPE	Echoes RUBOUT characters as backspace-space-backspace. Use this mode if your console terminal is a VT50, VT05, VT52, VT55, VT61, VT100, or if GT ON is in effect. The default is NOSCOPE. The system returns to that condition after a reboot. Note that you delete TAB characters by typing a single RUBOUT or DELETE, even though the cursor does not move back the correct number of spaces. This is a restriction in SCOPE modes.
TT:	NOSCOPE	Echoes RUBOUT characters by enclosing the deleted characters in backslashes. This is the normal mode. The system returns to this condition after a reboot.
TT:	TAB	Indicates that the console terminal is capable of executing hardware tabs. This setting is not valid for the non-multi-terminal SJ monitor. The default is NOTAB. The system returns to that condition after a reboot.
TT:	NOTAB	Simulates tab stops every eight positions. Many terminals supplied by DIGITAL have hardware tabs. This setting is not valid for the non-multi-terminal SJ monitor. This is the normal mode. The system returns to this condition after a reboot.
TT:	WIDTH = n	Sets the terminal width to *n*, where *n* is an integer between 30 and 255. The system initially sets the width to 80. This setting is not valid for the non-multi-terminal SJ monitor. (See SET TT CRLF.) The system returns to 80 after a reboot.
USR	SWAP	Allows the background job to place the user service routine (USR) in a swapping state. This setting is not valid for the XM monitor. This is the normal mode for FB and SJ monitors. The system returns to this condition after a reboot.
USR	NOSWAP	Prevents the background job from placing the USR in a swapping state. This setting is not valid for the XM monitor. The default is SWAP for FB and SJ monitors. The system returns to that condition after a reboot.
WILD	EXPLICIT	Causes the system to recognize file specifications exactly as you type them. If you omit a file name or a file type in a file specification the system does not automatically replace the missing item with an asterisk (*). Wildcards are described in Section 4.2 of this manual. The default is IMPLICIT. The system returns to that condition after a reboot.
WILD	IMPLICIT	Causes the system to interpret missing fields in file specifications as asterisks (*). Wildcards are described in Section 4.2 of this manual. Table 4–2 shows how the system interprets commands that have missing fields. This is the normal mode. The system returns to this condition after a reboot.

The following examples illustrate the SET command. This command allows the system to send lower-case characters to the printer:

```
.SET LP LC
```

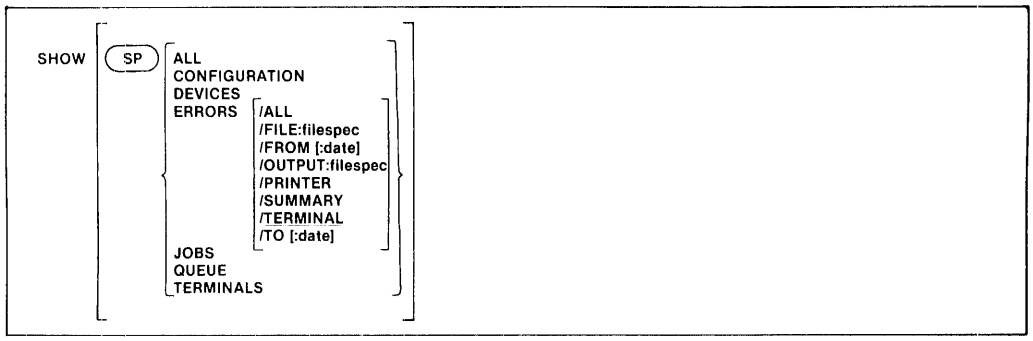
The next command sets the system wildcard default to implicit.

```
.SET WILD IMPLICIT
```

As a result of this command the system inserts an asterisk in place of a missing file name or file type in a file specification. See Table 4–2 for a list of these commands.

SHOW

The SHOW command prints information about your RT–11 system on the console terminal.



The information includes hardware configuration, monitor version, special features in effect, device names and logical device name assignments, terminal characteristics for terminals currently active on a multi-terminal system, and device handler status. If you are running the Error Logger or QUEUE, the SHOW command can provide information on errors and the update status of files waiting to be sent to an output device.

If you specify SHOW without an option, SHOW displays your system's device assignments. The devices the system lists are those known by the RT–11 monitor currently running in memory. This list reflects any additions or deletions you have made with the INSTALL and REMOVE commands. The listing also includes additional information about particular devices. The informational messages and their meanings are:

Message	Indicated Condition
(RESORC) or =RESORC	The device or unit is assigned to the background job RESORC (for FB and XM monitors only).
(FORE) or =FORE	The device or unit is assigned to the foreground job (for FB and XM monitors only and monitors without system job support).
(jobname) or =jobname	The device or unit is assigned to the system or foreground job (for FB and XM monitors that have system job support), where *jobname* represents the name of the system or foreground job.
(Loaded)	The handler for the device has been loaded into memory with the LOAD command.
(Resident)	The handler for the device is included in the resident monitor.

=logical-device-name(1), logical-device-name(2)... ,logical-device-name(n)	The device or unit has been assigned the indicated logical device names with the ASSIGN command.
xx free slots	The last line tells the number of unassigned, or free, device slots.

The following example was created under an FB monitor that has system job support. It shows the status of all devices known to the system.

```
.SHOW
TT   (Resident)
RK   (Resident)
  RK1    = SY, DK, OBJ, SRC, BIN
  RK2    = LST, MAP
MQ   (Resident)
DL   (Loaded)
DM
DX   (Loaded)
  DX0:   (MYPROG)
  DX1:   (RESORC)
LP:  (Loaded=QUEUE)
MT
CT
5 free slots
```

The listing shows first that TT and RK are resident in memory. The other device handlers known to the system are MQ, DL, DM, DX, LP, MT, and CT. There are five free slots in the table. RK0: has the logical names SY, DK, OBJ, SRC, and BIN. RK1: has the logical names LST and MAP. The DX handler is loaded and device DX0: belongs to the foreground job, MYPROG. The LP: handler is loaded and belongs to the system job, QUEUE.

The options for the SHOW command follow.

ALL This option is a combination of CONFIGURATION, DEVICES, JOBS, and TERMINALS, in that order. The ALL option also tests the device assignments.

CONFIGURATION This option displays the monitor version number and patch level, the monitor SET options in effect, the hardware configuration, and the special features in effect (if any). The listing varies, of course, depending on which monitor and which hardware system you are using.

First, the listing always shows the version number and patch level of the currently running monitor.

Next, information about the monitor is displayed. The first line indicates the device from which the system was bootstrapped. The next line prints the resident monitor's base address, in octal. Then the listing shows whether the user service routine (USR) is set to SWAP or NOSWAP. Another line prints out if a foreground job is loaded. The listing shows whether TT is set QUIET or NOQUIET, and whether the indirect file abort level is set to NONE, WARNING, ERROR, or SEVERE. The indirect file nesting depth prints out as a decimal number.

Next, the listing displays the system hardware configuration. It lists the processor type, which can be one of the following:

LSI 11
PDT 130/150
PDP 11/04
PDP 11/05,10
PDP 11/15,20
PDP 11/23
PDP 11/34
PDP 11/35,40
PDP 11/45,50,55
PDP 11/60
PDP 11/70

A separate line prints out for each of the following items that is present on your system:

FP11 Hardware Floating Point Unit
Commercial Instruction Set (CIS)
Extended Instruction Set (EIS)
Floating Instruction Set (FIS)
KT11 Memory Management Unit
Parity Memory
Cache Memory

If you have graphics hardware (VT11 or VS60), another line is printed out to indicate it. The clock frequency (50 or 60 cycles) prints next, followed by a line for the KW11–P programmable clock, if there is one on your system.

Finally, the listing either shows that there are no special features in effect, or it lists the appropriate features from the following list:

Device I/O time-out support
Error logging support
Multi-terminal support
Memory parity support
SJ timer support
System job support

The following example was created on a PDP 11/23 processor:

```
.SHOW CONFIGURATION

RT-11FB(S)  V04.00

Booted from RK0:
Resident Monitor base is 137500 (48960.)
USR is set SWAP
TT is set NOQUIET
Indirect file abort level is ERROR
Indirect file nesting depth is 3

PDP 11/23
60 Cycle System Clock
Error logging support
Device I/O time-out support
Memory parity support
```

DEVICES This option displays the RT–11 device handlers, their status, and their vectors. The messages for handler status are as follows:

Installed
Not installed
-Not installed (the handler special features do not match those of the monitor)
nnnnnn (load address of handler)
Resident

The following example uses SHOW DEVICES.

```
.SHOW DEVICES
Device          Status          Vector

  DX        Installed       264
  RK        Resident        220
  RF        Not installed   204
  DT        Installed       214
  LP        Installed       200
  CR        Not installed   230
  NL        Installed       000
  PC        Installed       070 074
  CT        Installed       260
  DS        Installed       204
  DM        Installed       210
  DL        Installed       330
  DP        Installed       254
  DY        Installed       270
  MT        Installed       224
  MM        Not installed   224
```

In the preceding example, note that the PC handler has two vectors. One is for the paper tape reader and the other is for the paper tape punch. Because of its special format, the TT handler is never listed.

JOBS This option displays data about the jobs that are currently loaded. This option also tells the following:

- the job name and number (if you have not enabled system job support on your monitor, the foreground job name appears as FORE, and its priority is 1)
- the console the job owns (if a non-multi-terminal monitor, this space is blank)
- the priority level of the job
- the job's running state (running, suspended, or done (but not unloaded))
- the low and high memory limits of the job
- the start address of the job's impure area

The example that follows displays data about currently running jobs:

```
•SHOW JOBS

Job    Name   Console Level State     Low     High   Impure
14     QUEUE      0     6   Suspend 116224  130306  115254
 0     RESORC     0     0   Run     000000  115210  134126
```

ERRORS The SHOW ERRORS command is valid only if you have error logging enabled on your monitor. For a complete description of the error logger and directions on how to start it, see Chapter 19, Error Logging. Note that the error logger is a special feature, available only through the system generation process. Because the error logger compiles statistics on each I/O transfer that occurs, in addition to hardware errors that occur, it is a good idea to enable error logging on a spare system volume that you use only when you want to compile error statistics.

The SHOW ERROR command invokes ERROUT, one of the three programs in the error logging package that runs as a background job. ERROUT creates reports on the I/O and error statistics the error logger compiles, and can print the reports at the terminal, line printer, or store the reports in a file you specify. For complete descriptions of the reports ERROUT creates, see Chapter 19, Error Logging.

ERRORS <RET>	prints a full report on each I/O transfer that has occurred in addition to each I/O, memory parity, and cache memory error that has occurred.
ERRORS/ALL	same as SHOW ERRORS <RET>
ERRORS/FILE:filespec	prints a full I/O transfer and error report from the file you specify. The file you specify must be of the same format that the error logger uses for its statistics compilations.
ERRORS/FROM[:date]	prints a full I/O transfer and error report for errors that occurred starting from the date you specify. Enter the date as *dd:mmm:yy*, where dd is a two-digit date (decimal) mmm is the first three characters of a month yy is a two-digit year
ERRORS/TO[:date]	prints a full I/O transfer and error report for errors that occurred up to the date you specify.
ERRORS/OUTPUT:filespec	enters the I/O transfer and error report in the output file you specify. This is useful if you want to save the error logging reports.

ERRORS/PRINTER	prints the I/O transfer and error report at the line printer.
ERRORS/SUMMARY	prints a summary error report at the terminal. The summary error report lists only the errors that occurred, not the successful I/O transfers.
ERRORS/TERMINAL	prints the I/O transfer and error report at the terminal. /TERMINAL is the default setting.

QUEUE Use the SHOW QUEUE command to get a listing of the contents of the queue. This option is invalid if you are not running QUEUE (see Chapter 20, Queue Package). The listing shows the output device, job name, input files, job status, and number of copies for each job that is queued. The sample command line that follows lists the current contents of the queue.

```
.SHOW QUEUE
DEVICE     JOB      STATUS     COPIES    FILES

 LP0:      LAB2       P          1       PASS3  .LST
                                 2       PASS4  .LST
                                 2       PASS5  .LST
 LP0:      HODG       Q          3       MESMAN.DOC
 DT1:      JUDITH     Q          2       PART1  .DOC
                                 2       PART2  .DOC
 MT1:      SZYM       Q          1       REFMAN.TXT
 LP0:      JOYCE      Q          1       SSM    .DOC
                                 1       DOCPLN.DOC
```

The job STATUS column prints a *P* if the job is currently being output, an *S* if the job being output is suspended, or a *Q* if the job is waiting to be output. If you have a large lineup of files, and your console is a video terminal, you can use the CTRL/S and CTRL/Q commands to control the scrolling of the listing.

TERMINALS This option indicates the status of and special features in effect for currently active terminals on multi-terminal systems. If your system has only the console terminal, the following message prints:

```
No multi-terminal support
```

Multi-terminal support is a special feature; it is not part of the distributed RT–11 monitors.

If your system does have multi-terminal support, SHOW TERMINALS prints a table of the existing terminals and lists the following information:

Unit number	**(0-15)**
Owner:	Background, foreground, system job owner, or none
Type:	Local Remote (dial-up) Console

(continued on next page)

Unit number **(0-15)**

S-console (shared by background and foreground or system job)
Is attached to another job (the foreground)

Interface type: DL DZ

Width: (width in characters, up to 255)

SET options in effect:
TAB
CRLF
FORM
SCOPE

Line speed: (baud rate if DZ; not applicable if DL)

The following example shows the terminal status of an RT–11 system.

```
.SHOW TERMINALS
Unit  Owner      Type        WIDTH TAB CRLF FORM SCOPE SPEED

  0   RESORC  S-Console DL    132  No  Yes   No   No    N/A
  1   FORE    Local     DL     80  Yes No    No   Yes   N/A
```

SQUEEZE

The SQUEEZE command consolidates all unused blocks into a single area on the device you specify and consolidates the directory entries on the device.

In the command syntax illustrated above, *device* represents the block replaceable volume to be compressed. To perform a squeeze operation, the system moves all the files to the beginning of the device you specify, producing a single unused area after the group of files. The squeeze operation does not change the bootstrap blocks of a device. The system prints a confirmation message before it performs the squeeze operation. You must type Y followed by a carriage return to execute the command.

The squeeze operation does not move files with .BAD file types. This feature prevents you from reusing bad blocks that occur on a disk. During a squeeze operation, files with a .BAD file type will be renamed FILE.BAD. The system inserts files before and after .BAD files until the space between the last file it moved and the .BAD file is smaller than the next file to be moved. Note that you should use the SQUEEZE command when you get a *directory full* error, even if there is still space remaining on the volume.

If you perform a squeeze operation on the system device, the system automatically reboots the running monitor when the compress operation completes. This reboot takes place in order to prevent system crashes that might occur when the monitor file or handler files are moved. The system volume cannot be squeezed if a foreground or system job is loaded.

The options for the SQUEEZE command follow.

/OUTPUT:filespec Use this option to transfer all the files from the input device to the output device in compressed format, an operation that leaves the input device unchanged. The output device must be an initialized block replaceable volume. (Use the INITIALIZE command to do this.) Note that the system does not query you for confirmation before this operation proceeds. If the output device is not initialized, the system prints an error message and does not execute the command. Note that /OUTPUT does not copy boot blocks; you must use the COPY/BOOT command to make the output volume bootable. The following example transfers all the files from RK0: to RK1: in compressed format, leaving RK0: unchanged.

```
.SQUEEZE/OUTPUT:RK1: RK0:
```

/QUERY This option causes the system to print a confirmation message before it executes a squeeze operation. You must respond by typing a Y followed by a carriage return for execution to proceed. This is the default operation. /QUERY is invalid with the /OUTPUT option.

/NOQUERY Use this option to suppress the confirmation message that prints before a squeeze operation executes. The following command compresses all the files on device DT1: and does not query.

```
.SQUEEZE/NOQUERY DT1:
```

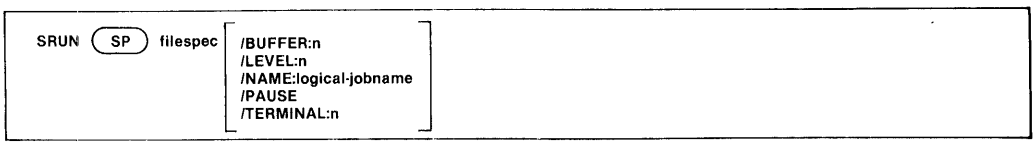
/WAIT This option is useful if you have a single-disk system. When you use /WAIT, the system initiates the SQUEEZE operation, but then pauses and waits for you to mount the volume you want to squeeze. When the system pauses, it prints *Mount input volume in <device>; Continue?*, where *<device>* represents the device into which you mount the volume. When the volume is mounted, type Y followed by a carriage return.

The following sample command line squeezes an RK05 disk:

```
.SQUEEZE/WAIT RK0:
RK0:/Squeeze? Are you sure? Y
Mount input volume in RK0:? Continue? Y
Mount system volume in RK0:? Continue? Y
```

Note that the system may repeat the *Mount input volume—Mount output volume* cycle several times to complete the SQUEEZE operation.

The SRUN command initiates system jobs.



In the command syntax illustrated above, *filespec* represents the program to be executed. Because this command runs a system job, it is valid only for FB and XM monitors that have system job support — a special feature enabled through the system generation process.

You can run up to six system jobs simultaneously, in addition to the foreground and background jobs. If you attempt to run a system job that is already active, an error message prints on the terminal.

Note that when you issue the SRUN command, the monitor assumes a .SYS file type. If you want to issue the SRUN command for a program that has a .REL file type, either enter the file type with the file name (for example, SRUN QUEUE.REL), or rename the file so it has a .SYS file type.

In an XM monitor, you can use the SRUN command to run a virtual .SAV image program. You must type the file type explicitly.

The options that you can use with SRUN follow.

/BUFFER:n Use this option to reserve space in memory over the actual program size. The argument *n* represents the number of octal words of memory to allocate. You must use this option to run any FORTRAN program as a system job. If you use this option for a virtual job linked with the /V option (or /XM), the system ignores /BUFFER because the system provides a buffer in extended memory.

/LEVEL:n Use this option to assign an execution priority level to the job, where *n* can be 1, 2, 3, 4, 5, or 6. If you attempt to assign the same priority level to two system jobs, an error message prints at the terminal. If omitted, the priority level defaults to the highest level that is thus far unassigned.

/NAME:logical-jobname Use this option to assign a logical job name to a program. This is the name that programmed requests and SYSLIB calls use to reference a system job. If you attempt to assign the same logical job name to two system jobs, an error message prints at the terminal. If you do not specify a logical job name, the system assumes the file name of the program.

/PAUSE Use this option to help you debug a program. When you type the carriage return as the end of the command string, the system prints the load address of your program and waits. By means of ODT, you can examine or modify the program before starting execution (see Chapter 21). You must use the RESUME command to restart the system job. The following com-

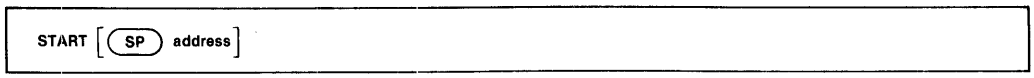
mand loads the program MFUNCT.SYS, prints the load address, and waits for a RESUME command to begin execution.

```
.SRUN MFUNCT/PAUSE
Loaded at 126556
.RESUME MFUNCT
```

/TERMINAL:n Use this option to change the console of the system job. Your system must have multi-terminal support — a special feature available only through system generation — before you can use it. The argument *n* represents a terminal logical unit number. By assigning a different terminal to interact with the system job, you eliminate the need for system, foreground, and background jobs to share the console terminal. Note that the original console terminal still interacts with the background job and with the keyboard monitor, unless you use the SET TT: CONSOL command to change this.

START

The START command initiates execution of the program currently in memory (loaded with the GET command) at the address you specify.



In the command syntax shown above, *address* is an even octal number representing any 16-bit address. If you omit the address or if you specify 0, the system uses the starting address that is in location 40. If the address you specify does not exist or is invalid for any reason, a trap to location 4 occurs and the monitor prints an error message. Note that this command is valid for background jobs only, and not for extended memory virtual jobs. The following command loads MYPROG.SAV into memory and begins execution.

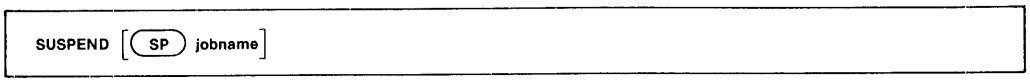
```
.GET MYPROG
.START
```

The next example loads MYPROG.SAV and ODT.SAV into memory, and begins execution at ODT's starting address.

```
.GET MYPROG
.GET ODT
.START 1222
 ODT V01.04
*
```

SUSPEND

The SUSPEND command temporarily stops execution of the foreground or system job.



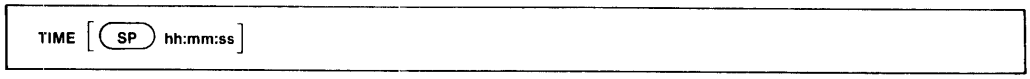
If you have system job support enabled on your monitor, specify the name of the system or foreground job you wish to suspend. If you do not have system job support, then do not include an argument with the SUSPEND command. The SUSPEND command is not valid for the SJ monitor. The system permits foreground input and output that are already in progress to finish; however, it issues no new input or output requests and enters no completion routines (see the *RT–11 Programmer's Reference Manual* for a detailed explanation of completion routines). You can continue execution of the job by typing the RESUME command. The following command suspends execution of the foreground job that is currently running on a system that does not have system job support.

```
.SUSPEND
```

The next command suspends execution of the system job, QUEUE, that is currently running on a system that does have system job support.

```
.SUSPEND QUEUE
```

Use the TIME command to set the time of day or to display the current time of day.



In the command syntax shown above, *hh* represents hours (from 0 to 23); *mm* represents minutes (from 0 to 59) and *ss* represents seconds (from 0 to 59). The system keeps time on a 24-hour clock.

To enter the time of day, specify the time in the format described above. You should do this as soon as you bootstrap the system. The following example enters the time, 11:15:00 A.M.

```
.TIME 11:15
```

As this example shows, if you omit one of the arguments the system assumes 0.

To display the current time of day, type the TIME command without an argument, as this example shows.

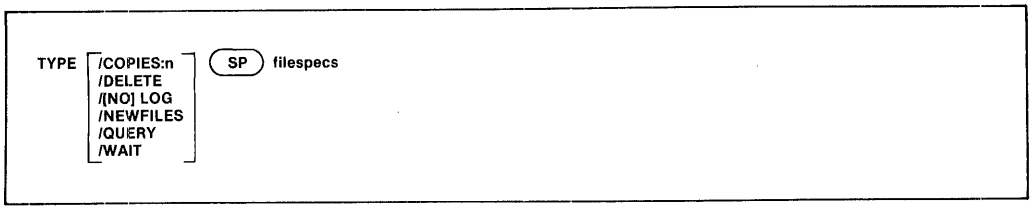
```
.TIME
11:15:01
```

When you install the standard RT-11 monitors, the clock rate is preset to 60 cycles. Consult the *RT-11 Installation and System Generation Guide* for information on setting the clock to a 50-cycle rate.

The FB and XM monitors automatically reset the time each day at midnight when a TIME command is used, or if a .GTIM programmed request is issued. (The TIME command issues a .GTIM programmed request.) The SJ monitor resets the time under these conditions only if you select timer support during the system generation process.

TYPE

The TYPE command prints the contents of one or more files on the terminal.



In the command syntax illustrated above, *filespecs* represents the file or files to be typed. You can explicitly specify up to six files as input to the TYPE command. The system types the files in the order in which you specify them in the command line. You can also use wildcards in the file specification. In this case, the system types the files in the order in which they occur in the directory of the device you specify. If you specify more than one file, separate the files by commas. If you omit the file type for a file specification, the system assumes .LST. You can specify the entire command on one line, or you can rely on the system to prompt you for information. The TYPE command prompt is *Files?*.

The TYPE command options and examples follow.

/COPIES:n Use this option to list more than one copy of the file. The meaningful range of values for the decimal argument *n* is from 2 to 32 (1 is the default). The following command, for example, types three copies of the file REPORT.LST on the terminal.

```
.TYPE/COPIES:3 REPORT
```

/DELETE Use this option to delete a file after it is typed on the terminal. This option must appear following the command in the command line. The TYPE/DELETE operation does not ask you for confirmation before it executes. You must use /QUERY for this function. The following example types a BASIC program on the terminal, then deletes it from DX1:.

```
.TYPE/DELETE DX1:PROG1.BAS
```

/LOG This option prints on the terminal the names of the files that were typed by the current command. Normally, the system prints a log only if there is a wildcard in the file specification. If you specify /QUERY, the query messages replace the log, unless you specifically type /LOG/QUERY in the command line. The following example shows a TYPE command and the resulting log.

```
.TYPE/LOG OUTFIL.LST
 Files copied:
DK:OUTFIL.LST    to TT:
```

/NOLOG This option prevents a list of the copied files from printing on the terminal. You can use this option to suppress the log if you use a wildcard in the file specification.

/NEWFILES Use this option in the command line if you need to type only those files that have the current date. The following example shows a convenient way to type all new files after a session at the computer.

```
.TYPE/NEWFILES *.LST
 Files copied:
DK:REPORT.LST    to TT:
```

/QUERY If you use this option, the system requests confirmation before it performs the operation. /QUERY is particularly useful on operations that involve wildcards, when you may not be sure which files the system selected for an operation. Note that if you specify /QUERY in a TYPE command line that also contains a wildcard in the file specification, the confirmation messages printed on the terminal replace the log messages that would normally appea You must respond to a query message by typing Y (or anything that begins with y) and a carriage retun to initiate execution of a articular operation. The system interprets ay other response to mean NO and does not perform the specific operation.

```
.TYPE/QUERY/DELETE *.LST
 Files copied/deleted:
DK:OUTFIL.LST    to TT:? NO
DK:REPORT.LST    to TT:? Y
```

/WAIT This option is useful if you have a single-disk system. When you use this option, the system initiates the TYPE operation, but then pauses and waits for you to mount the volume on which you want the operation to take place. When the system pauses, it prints *Mount input volume in <device>; Continue?,* where *<device>* represents the device into which you mount the volume. When you have mounted the volume, type Y followed by a carriage return.

The following sample command types AJAX.DOC from an RK05 disk:

```
.TYPE/WAIT RK0:AJAX.DOC
Mount input volume in RK0:; Continue?Y
```

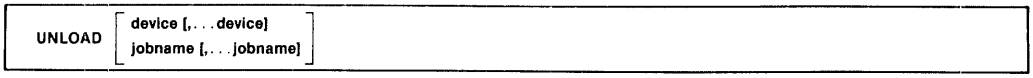
After the system has typed AJAX.DOC at the terminal, it issues the following prompt:

```
Mount system volume in RK0:;  Continue?
```

When you mount the system volume, and type a Y followed by a carriage return, you terminate the TYPE operation.

UNLOAD

The UNLOAD command removes previously loaded handlers from memory, thus freeing the memory space they occupied. It also removes terminated foreground or system jobs.



In the command syntax shown above, *device* represents the device handler to be unloaded. The colon that follows the device handler is optional with SJ, FB, and XM monitors and monitors that do not have system job support. You must include the colon if your system has system job support.

UNLOAD clears ownership for all units of the device type you specify. A request to unload the system device handler clears ownership for any assigned units for that device, but the handler itself remains resident. After you issue the UNLOAD command, the system returns any memory it frees to a free memory list. The background job eventually reclaims free memory. Note that if you interrupt an operation that involves magtapes or cassette, you must unload and then load (with the LOAD command) the appropriate device handler (MM, MT, MS, or CT).

The system does not accept an UNLOAD command while a foreground job is running if the foreground job owns any units of that device. This is because a handler that the foreground job needs might become nonresident. You can unload a device while a foreground job is running if none of its units belong to the foreground job.

A special function of this command is to remove a terminated foreground or system job and reclaim memory, since the system does not automatically return the space occupied by the foreground or system job to the free memory list. The following command unloads the foreground job and frees the memory it occupied. This command is valid only if the foreground job is not running.

```
.UNLOAD F
```

The following command unloads the system job QUEUE.

```
.UNLOAD QUEUE
```

The following command clears ownership of all units of RK:. If RK: is the system device, the RK handler itself remains resident.

```
.UNLOAD RK:
```

The next command releases the line printer and DECtape handlers and frees the area they previously held.

```
.UNLOAD LP:,DT:
```

Table 5-1: EDIT Key Commands (Cont.)

Key	Explanation
CTRL/C	In the first command, the three CTRL/C characters are part of the text object of the Insert command. EDIT treats them like any other character. In the second command string, the CTRL/C occurs at command level, which causes the editor to terminate. If no commands (other than CLOSE) are executed between the time you terminate the editor and the time you issue a REENTER command, the text buffer is preserved as it was at program termination. However, only the text buffer is preserved. The input and output files are closed, and the save and macro buffers are reinitialized. If you inadvertently terminate an editing session before the output file can be closed, you can often use the monitor CLOSE command to make permanent the portion of the output file that has already been written (see Section 4.4). You can then reenter the editor, open a new output file, and continue the editing session.
CTRL/O	Echoes as ^O and a carriage return. Inhibits printing on the terminal until completion of the current command string. Typing a second CTRL/O resumes output.
CTRL/U	Echoes as ^U and a carriage return. Deletes all characters on the current terminal input line. (Typing CTRL/U has the same effect as pressing the RUBOUT key until all the characters back to the beginning of the line are deleted.)
RUBOUT or DELETE	Deletes a character from the current command line; echoes as a backslash followed by the character deleted. Each succeeding RUBOUT you type deletes and echoes another character. An enclosing backslash prints when you type a key other than RUBOUT. This erasure is done from right to left. Since EDIT accepts multiple-line commands, RUBOUT can delete past the carriage return/line feed combination and delete characters on the previous line. You can use RUBOUT in both command and text modes.
TAB	Spaces to the next tap stop. Tab stops are positioned every eight spaces on the terminal; pressing the TAB key causes the carriage to advance to the next tab position.

CTRL/X	Echoes as ^X and a carriage return. CTRL/X causes the editor to ignore the entire command string you are currently entering. The editor prints a carriage return/line feed combination and an asterisk to indicate that you can enter another command. For example:
	
	`*IABCD`
	`EFGH^X`
	`*`
	
	A CTRL/U would cause only deletion of EFGH; CTRL/X erases the entire command.
	
	If you are running a system job, you must SET TT:NOFB to enable this function of CTRL/X. If you do not and you type CTRL/X, the system intercepts the CTRL/X and prompts you for a system job name.

5.4 Command Structure

EDIT commands fall into eight general categories. Table 5–2 lists these categories, the commands they include, and the sections of this manual where you will find information on the particular command.

Table 5–2: EDIT Command Categories

Category	Commands	Section
File open and close		
	Edit Backup	5.6.1.3
	Edit Read	5.6.1.1
	Edit Write	5.6.1.2
	End File	5.6.1.4
File input/output		
	EXit	5.6.2.4
	Next	5.6.2.3
	Read	5.6.2.1
	Write	5.6.2.2
Immediate mode		
	ESCAPE	5.7.2
	CTRL D	5.7.2
	CTRL G	5.7.2
	CTRL N	5.7.2
	CTRL V	5.7.2
	RUBOUT	5.7.2
Pointer location		
	Advance	5.6.3.3
	Beginning	5.6.3.1
	Jump	5.6.3.2
Search		
	Find	5.6.4.2
	Get	5.6.4.1
	Position	5.6.4.3
Text listing		
	List	5.6.5.1
	Verify	5.6.5.2
Text modification		
	Change	5.6.6.4
	Delete	5.6.6.2
	eXchange	5.6.6.5
	Insert	5.6.6.1
	Kill	5.6.6.3
Utility		
	Edit Console	5.7.1
	Edit Display	5.7.1
	Edit Lower	5.6.7.6
	Edit Upper	5.6.7.6
	Edit Version	5.6.7.5
	Execute Macro	5.6.7.4
	Macro	5.6.7.3
	Save	5.6.7.1
	Unsave	5.6.7.2

to specify a special procedure for tape handling during cassette operations with PIP. The following operations are valid for use with cassettes: /A, /B, /C, /D, /G, /M, /P, /Q, /R, /S, /U, /W, and /Y. The following options are invalid with cassettes: /K, /N, /O, /R, /F, /Z, and /T. If you omit the /M:n option in a cassette operation, the cassette rewinds before each operation (using /M:0 has the same effect). The character, *n* in /M:n, represents a count of the number of files from the present position on the cassette. Note that the /M:n option has a different meaning for magtape (Section 7.2.1.2 describes how to use /M:n with magtape).

In copying from cassettes, /M:n functions as follows:

1. If *n* is 0:

   The cassette rewinds and PIP searches for the file you specify. If you specify more than one file, or if you use a wildcard in the file specification, the cassette rewinds before PIP searches for each file.

2. If *n* is a positive integer:

   PIP starts from the cassette's present position and searches for the file you specify. If PIP does not find the file by the time it reaches the *n*th file from its starting position, it uses the *n*th file for the read operation. Note that if PIP's starting position is not the beginning of the cassette, it is possible that PIP will not find the file you specify, even though it does exist on the tape.

3. If *n* is a negative integer:

   The cassette rewinds, then PIP follows the procedure outlined in step 2 above.

In writing to cassettes, /M:n functions as follows:

1. If *n* is 0:

   The cassette rewinds and PIP writes the file you specify starting at the logical end-of-tape (LEOT) position. PIP deletes any file it finds along the way that has the same name and file type as the file you specify.

2. If *n* is a positive integer:

   PIP starts from the cassette's present position and searches *n* files ahead, deleting along the way any file it finds that has the same name and file type as the file you specify. If it does not reach LEOT before it reaches the *n*th file from its starting position, it enters the file you specify over the *n*th file and deletes any files beyond it on the tape. If PIP reaches LEOT before it reaches the *n*th file, it writes the file you specify at the end-of-tape.

3. If *n* is a negative integer:

   The cassette rewinds, then PIP follows the procedure outlined in step 2 above.

If you are copying a file to cassette and reach the physical end-of-tape before the copy completes, PIP automatically continues the file on another cassette. The cassette device handler prints the *CTn: PUSH REWIND OR MOUNT NEW VOLUME* message. If you want to halt the copy operation at this point, push the cassette rewind button. The tape rewinds, PIP prints an error message, and then prompts you for a new command. However, if you want to continue the file on another cassette, remove the first cassette and put another initialized cassette in its place. The new cassette rewinds immediately. PIP then continues copying the file. The continued part of the file has the same file name and file type as the first part of the file, but PIP adds one to its sequence number to show that it is a continued file. Make sure you have a supply of initialized cassettes handy for cassette copy operations; you cannot interrupt the copy operation to initialize a cassette when PIP is waiting for a new volume. The following example shows a copy operation that fills one cassette and continues to another.

```
*CT1:*.*=RK:RT*.SYS,Z%.SYS/Y/W/M:1
 Files copied:
RK:RT11SJ.SYS  to CT1:RT11SJ.SYS
CT1: PUSH REWIND OR MOUNT NEW VOLUME
RK:RT11FB.SYS  to CT1:RT11FB.SYS
RK:DT.SYS      to CT1:DT.SYS
RK:DP.SYS      to CT1:DP.SYS
RK:DX.SYS      to CT1:DX.SYS
RK:RF.SYS      to CT1:RF.SYS
RK:RK.SYS      to CT1:RK.SYS
RK:DM.SYS      to CT1:DM.SYS
RK:DS.SYS      to CT1:DS.SYS
RK:TT.SYS      to CT1:TT.SYS
RK:LP.SYS      to CT1:LP.SYS
RK:CR.SYS      to CT1:CR.SYS
RK:MT.SYS      to CT1:MT.SYS
RK:MM.SYS      to CT1:MM.SYS
RK:NL.SYS      to CT1:NL.SYS
RK:PC.SYS      to CT1:PC.SYS
RK:CT.SYS      to CT1:CT.SYS
RK:BA.SYS      to CT1:BA.SYS
*
```

A directory listing of the second cassette shows that the first file, RKMNFB.SYS, is continued from a previous tape. (The number of blocks in a cassette directory listing is not meaningful; it really represents the total of sequence numbers in the directory.)

```
.DIRECTORY CT1:
 15-Apr-79
RT11FB.SYS     1  15-Apr-79      DT .SYS    0  15-Apr-79
DP     .SYS    0  15-Apr-79      DX .SYS    0  15-Apr-79
RF     .SYS    0  15-Apr-79      RK .SYS    0  15-Apr-79
DM     .SYS    0  15-Apr-79      DS .SYS    0  15-Apr-79
TT     .SYS    0  15-Apr-79      LP .SYS    0  15-Apr-79
CR     .SYS    0  15-Apr-79      MT .SYS    0  15-Apr-79
MM     .SYS    0  15-Apr-79      NL .SYS    0  15-Apr-79
PC     .SYS    0  15-Apr-79      EL .SYS    0  15-Apr-79
CT     .SYS    0  15-Apr-79      BA .SYS    0  15-Apr-79
 18 Files, 1 Blocks
```

1. If the file sequence number is greater than the current position, PIP searches the tape in the forward direction.
2. If the file sequence number is more than one file before the current position, or if the file sequence number is less than five files from the beginning-of-tape (BOT), the tape rewinds before PIP begins its search.
3. If the file sequence number is at the current position, or if it is one file past the current position, PIP searches the tape in the reverse direction.

Whenever you fetch or load a new copy of the magtape handler, the tape position information is lost. The "new" handler searches backward until it locates either BOT or a label from which it can learn the position of the tape. It then operates normally, according to steps 1, 2, and 3 described above.

If you omit the /M:n option, the tape rewinds between each operation. Using /M:0 has the same effect as omitting /M:n. When *n* is positive, it represents the file sequence number. When *n* is negative, it represents an instruction to the magtape handler.

In copying from magtapes, /M:n functions as follows:

1. If *n* is 0:

   The tape rewinds and PIP searches for the file you specify. If you specify more than one file, the tape rewinds before each search. If the file specification contains a wildcard, the tape rewinds only once and then PIP copies all the appropriate files.

2. If *n* is a positive integer:

   PIP goes to file sequence number *n*. If the file it finds there is the one you specify, PIP copies it. Otherwise, PIP prints the *?PIP-F-File not found* message. If you use a wildcard in the file specification, PIP goes to file sequence number *n* and then begins to search for matching files.

3. If *n* is –1:

   PIP starts the search at the current position. If the current position is not the beginning of the tape, PIP may not find the file you specify, even though it does exist on the tape.

In writing to magtapes, /M:n functions as follows:

1. If *n* is 0:

   The tape rewinds before PIP copies each file. PIP prints a warning message if it finds a file with the same name and file type as the input file and does not perform the copy operation.

2. If *n* is a positive integer:

   PIP goes to the file sequence number *n* and enters the file you specify. If PIP reaches logical end-of-tape (LEOT) before it finds file sequence number *n*, it prints the *?PIP-F-File sequence number not found* message. If you specify more than one file or if you use a wildcard in the file speci-

fication, the tape does not rewind before PIP writes each file, and PIP does not check for duplicate file names.

3. If *n* is –1:

   PIP goes to the LEOT and enters the file you specify. It does not rewind, and it does not check for duplicate file names.

4. If *n* is –2:

   The tape rewinds between each copy operation. PIP enters the file at LEOT or at the first occurrence of a duplicate file name.

If PIP reaches the physical end-of-tape before it completes a copy operation, it cannot continue the file on another tape volume. Instead, it deletes the partial file by backspacing and writing a logical end-of-tape over the file's header label. You must restart the operation and use another magtape.

If you type consecutive CTRL/Cs during any output operation to magtape, PIP does not write a logical end-of-tape at the end of the data. Consequently, you cannot transfer any more data to the tape unless you follow one of the following recovery procedures.

1. Transfer all good files from the interrupted tape to another tape and initialize the interrupted tape in the following manner:

```
*dev1:*.*=dev0:*.*
^C
.R DUP
*dev0:/Z/Y
```

2. Determine the sequential number of the file that was interrupted and use the /M:n construction to enter a replacement file (either a new file or a dummy) over the interrupted file. PIP writes the replacement file and a good LEOT after it. The following example assumes the bad file is the fourth file on the tape:

```
*dev0:file.new/M:4=file.dum
```

7.2.2 Copy Operations

PIP copies files in image, ASCII, and binary format. Other options let you change the date on the files, access .SYS files, combine files, and perform other similar operations. PIP automatically allocates the correct amount of space for new files in copy operations (except for concatenation). For block-replaceable devices, PIP stores the new file in the first empty space large enough to accommodate it. If an error occurs during a copy operation, PIP prints a warning message, stops the copy operation, and prompts you for another command. You cannot copy .BAD files unless you specifically type each file name and file type.

7.2.2.1 Image Mode – If you enter a command line without an option, PIP copies files onto the destination device in image mode. Note that you cannot

block on either the input or output volume, it prints an error message. However, it retries the operation and performs the copy one block at a time. If only one error message prints, you can assume that the transfer completed correctly.

Qualifiers to the /I option let you:

1. Specify the blocks to be read from the input device and a starting block number for the write operation on the output device.
2. Copy a file to a device, or a device to a file, by specifying a file name with either the input or output device. For example, you can copy a diskette to an RL01 as a file, or a file on an RK05 to a diskette.

NOTE

When you use /F and you have specified a magtape or cassette as the input device, you must specify an input file name.

The syntax of the command is:

output-device: $\left\{\begin{matrix}\text{filename} \\ \text{A}\end{matrix}\right\}$ [/G:rn]=input-device[filename]/I[/G:rn/E:rn][/F][/H]

where:

filename	represents the output file name of the input device, or (when specified with the input device) represents the input file name you are copying to the output device. If you specify an input file, use the dummy file name *A* with the output specification. Note that you can use a file name with either the input or output, but never with both
A	represents a dummy file name (required if the output device is not a magtape or cassette). Note that either *filename* or *A*, but not both, can be specified with the output device
/G:rn	when specified with the output device, represents the starting block number for the write operation. When specified with the input device, it represents the starting block number of the read operation
/E:rn	represents the ending block number on the input device for the read operation
/F	indicates that a file is involved in the transfer
/H	verifies that the input matches the output

The command string must include an input and an output specification; there is no default device. The /I operation does not copy to or from a device that has logical bad blocks. (Physical bad blocks can be logically replaced or

covered, as Sections 8.2.12.3 and 8.2.12.4 describe.) If one device is smaller than the other, DUP copies only the number of blocks of the smaller device.

You can copy blocks between disk and magtape. DUP stores the data on the tape, formatting it in 1K-word blocks. It is possible to store only one disk image on a magtape, regardless of the size of the tape.

You can use the /H option with /I to verify that the input matches the output after an image mode copy operation.

NOTE

The /I option does not copy track 0 of diskettes. However, this restriction has no impact on any copy operations if your diskette was supplied by DIGITAL.

The following examples use the /I option. The file name A is not significant; it is a dummy file name required by the Command String Interpreter.

```
*RK1:A=RK0:/I
RK1:/Copy:  Are you sure?
```

The command shown above copies all blocks from RK0: to RK1:.

```
*RK1:A/G:501=RK0:/I/G:0/E:500
RK1:/Copy:  Are you sure?Y
```

The command shown above copies all blocks 0–500 from RK0: to blocks 501–1000 on RK1:.

```
RK1:FLOPPY.BAK/F=DX0:/I
RK1:/Copy:  Are you sure?Y
```

The last command copies device DX0: to RK1: in a file named FLOPPY.BAK.

8.2.3 Bad Block Scan Option (/K)

Some mass storage volumes (disks, diskettes, DECtape, and DECtape II) have bad blocks, or they develop bad blocks as a result of age and use. You can use the /K option to scan a device and locate bad blocks on it. DUP prints the absolute block number of those blocks on the device that return hardware errors when DUP tries to read them. If you specify an output file, DUP prints the bad block report in that file. Remember that block numbers are octal and the first block on a device is block 0. If DUP finds no bad blocks, it prints an informational message. A complete scan of a volume takes from one to several minutes depending on the size of the volume. It does not destroy data that is stored on the device.

```
*RK0:RT11SJ.SYS/O
RT-11SJ     V04.00
```

To boot a different monitor, for example the foreground/background monitor (for DX0:), type:

```
*DX0:RT11FB.SYS/O
```

8.2.5 Boot Foreign Volume Option (/Q)

Use the /Q option with /O to boot a volume that has a monitor other than the RT-11 version 4 monitor. Note that you must use /Q to boot any version 3B or earlier volume of RT-11. The following example boots an RT-11 version 3B volume.

```
*RK0:/O/Q
RT-11SJ V03B-00B
```

DUP does not retain the date and time when you use the /Q option.

8.2.6 Squeeze Option(/S)

Use the /S option to compress a volume (disk, diskette, DECtape) onto itself or onto another disk or DECtape. To do this, DUP moves all the files to the beginning of the device, producing a single, unused area after the group of files. The squeeze operation does not change the bootstrap blocks of a device. The output device you specify, if any, must be an initialized device. If you specify an output device, DUP does not query you for confirmation before it performs the operation. If you do not specify an output device, DUP prints the *Are you sure?* message and waits for your response before proceeding. You must type Y followed by a carriage return for the command to be executed. Since it is critical to perform an error-free squeeze operation, be sure to scan a device (with /K) before you use /S.

The /S option does not operate on files with .BAD file types, preventing you from reusing bad blocks that occur on a disk. You can rename files containing bad blocks, giving them a .BAD file type, and therefore cause DUP to leave them in place when you execute a /S. During a squeeze operation, files with a .BAD file type are renamed FILE.BAD. DUP inserts files before and after .BAD files until the space between the last file it moved and the .BAD file is smaller than the next file to be moved. If an error occurs during a squeeze operation, DUP continues the operation, performing it one block at a time. If only one error message prints, you can assume that the operation completed correctly.

The syntax of the command is:

[output-device =]input-device/S

Do not use /S on the system device (SY:) when a foreground or system job is loaded. A *?DUP-F-Can't squeeze SY: while foreground loaded* error message results if you attempt this, and DUP ignores the /S operation. You must unload the foreground job before using the /S option.

NOTE

If you perform a compress operation on the system device, the system automatically reboots when the compress operation is completed. This operation takes place in order to prevent system crashes that can occur when a system file is moved.

You can use /X with /S to suppress the automatic reboot and leave DUP running. However, you should use /X only if you are certain that the monitor file will not move. Even then, you should reboot the system when the squeeze operation completes if the device handlers have moved.

The following examples use the /S option:

```
*SY:/S
SY:/Squeeze?  Are you sure?Y

RT-11SJ     V04.00
```

The command shown above compresses the files on the system device and reboots the system when the compress operation completes.

NOTE

If you compress your system volume, make sure DUP program within has the name DUP.SAV. If not, a system failure may occur.

```
*DT1:A=DT2:/S
```

This command transfers all the files from device DT2: to device DT1:, leaving DT2: unchanged. The file name A is not significant; it is a dummy file name required by the Command String Interpreter.

8.2.7 Extend Option (/T:n)

Use the /T option to extend the size of a file. The syntax of the command is:

filespec = /T:n

where:

filespec	represents the device, file name, and file type of the file to be extended
n	represents the number of blocks to add to the file

You can extend a file in this manner only if it is followed by an unused area at least *n* blocks long. Any blocks not required by the extend operation remain in the unused area.

The following example uses the /T option:

```
*DT1:ZYZ.TST=/T:100
```

This command assigns 100 more blocks to the file named ZYZ.TST on device DT1:.

8.2.8 Bootstrap Copy Option (/U[:xx])

In order to use a volume as a system volume, you must copy a bootstrap onto it. To do this, first make sure that the appropriate monitor file and handler are stored on the volume. For a diskette system, for example, check to see that the file DX.SYS is in the diskette directory. If it is, then you can copy the desired monitor onto the diskette, using the /U option. The option argument, *xx*, represents a target system device name. Use this argument when you are creating a bootable PDT volume if the current system is a PDP-11, and vice versa. Also you can use this argument when the current system is on an RX02 diskette and you wish to create a bootable RX01 diskette.

NOTE

When you use the /U option, make sure that the input volume is also the output volume.

To copy a bootstrap for the single-job monitor on RK1:, for example, use the following procedure:

1. Obtain a formatted disk. (Most disks, diskettes, DECtape, and DECtape II volumes are formatted by the manufacturer. However, Chapter 18, FORMAT, does outline the procedure for reformatting RK05, RK06, RK07, RP02, RP03 disks, and RX01 and RX02 diskettes.)
2. Initialize the disk with /Z (see Section 8.2.12).
3. Copy files onto the disk.
4. Copy the monitor and RK05 handler, RK.SYS, onto the disk.
5. Copy the monitor bootstrap onto the disk with /U.

The following example shows how to initialize a diskette, copy files to it, and write a bootstrap onto the diskette:

```
*DX1:/Z/Y
```

The command shown above (step 2 of the procedure described above) initializes the diskette.

```
*DX1:A=DX0:/S
```

This command, which combines steps 3 and 4, squeezes all the files from DX0: onto DX1:.

```
*DX1:A=DX0:RT11FB.SYS/U
```

The last command (step 5) writes the bootstrap for the foreground/background monitor onto the bootstrap blocks (blocks 0 and 2–5) of DX1:. The file name A is not significant; it is a dummy file name required by the Command String Interpreter.

To create a bootable PDT-11/150 system diskette while running on a PDP-11, specify PD with the /U option. Likewise, if you are running on a PDT-11/150, and you wish to create a bootable PDP-11 system diskette,

specify DX (for single-density diskette) or DD (for DECtape II) with the /U option. The following command creates a bootable PDT–11/150 diskette while the current system is on a PDP–11:

```
*DX0:A=DX0:RT11SJ.SYS/U:PD
```

The next command creates a bootable PDP–11 diskette while the current system is on a PDT–11/150:

```
*PD0:A=PD0:RT11SJ.SYS/U:DX
```

8.2.9 Volume ID Option (/V[:ONL])

You can use the /V option as an action option to print the volume ID of a device or to change the volume ID. The syntax of the command is:

device:[/Z]/V[:ONL]

where:

device: is the device whose volume ID you want to display or change

If you specify only /V, DUP prints out on the console terminal the volume ID and owner name of the device you specify. If you specify /Z with /V, DUP initializes the device and prompts you for a new volume ID and owner name. If you specify /Z/V:ONL, DUP assumes you want only to change the volume ID and owner name and not initialize the device.

When you specify either /Z/V or /Z/V:ONL, DUP prompts you for a volume ID:

```
Volume ID?
```

Respond with a volume ID that is up to 12 characters long for a block-replaceable device. Terminate your response with a carriage return. DUP then prompts for an owner name:

```
Owner?
```

Respond with an owner name that is up to 12 characters long for a block-replaceable device. Terminate your response with a carriage return. DUP ignores characters you type beyond the legal length. You cannot change the volume ID of a magtape or cassette without initializing the entire tape. The /V:ONL command changes only the volume ID and owner name; it does not initialize the device. Section 8.2.12.2 describes how to use /V with the /Z option to initialize a device and write new volume identification on it.

The following example uses the /V:ONL option:

```
*RK1:/Z/V:ONL
RK0:/Volume ID change; Are you sure? Y
Volume ID? FORTRAN VOL
Owner?     Marcy
```

This command writes a new volume ID and owner name on device RK1:.

If you specify :RET with /B, DUP will retain through initialization all FILE.BAD files created by a previous /B.

If you use the /B option to initialize a volume that has been previously initialized using the /R option, DUP creates FILE.BAD files to cover the bad blocks on the volume, and the system then ignores the bad block replacement table.

If the volume being initialized contains bad blocks, the system prints the locations of the bad blocks in octal and in decimal, as in the following example:

```
*RK0:/Z/B
RK0:/Initialize;  Are you sure ? Y
     Block           Type
000120      80.    Hard
000471     313.    Hard
000521     337.    Hard
?DUP-I-Bad blocks detected 3.
```

The left column lists the locations in octal, and the middle column lists the locations in decimal. The right column indicates the type of bad block found: hard or soft.

8.2.12.5 Restoring a Disk (/D) -- Use /D to "uninitialize" a volume if you have not transferred any files to it since initialization. DUP will restore all files and directory entries that were present before the volume was initialized. This option is useful if you initialize a volume by mistake.

Note that /D does not restore boot blocks. Thus, if you use /D to restore a previously bootable volume, use the bootstrap copy option, /U[:xx], to make the volume bootable again.

11.3 Input and Output

Linker input and output is in the form of modules; the linker uses one or more input modules to produce a single output (load) module. The linker also accepts library modules and symbol table definition files as input, and can produce a load map and/or symbol table definition file. The sections that follow describe all valid forms of input to and output from the linker.

11.3.1 Input Object Modules

Object files, consisting of one or more object modules, are the input to the linker. (Entering files that are not object modules may result in a fatal error.) Object modules are created by language translators such as the FORTRAN compiler and the MACRO–11 assembler. The module name item declares the name of the object module (see Section 11.3.4).

The first six Radix–50 characters of the .TITLE assembler directive are used as the name of the object module. These six characters must be Radix–50 characters (the linker ignores any characters beyond the sixth character). The linker prints the first module name it encounters in the input file stream (normally the main routine of the program) on the second line of the map following TITLE:. The linker also uses the first identity label (issued by the .IDENT directive) for the load map. It ignores additional module names.

The linker reads each object module twice. During the first pass it reads each object module to construct a global symbol table and to assign absolute values to the program section names and global symbols. The linker uses the library files to resolve undefined globals. It places their associated object modules in the root. On the second of its two passes, the linker reads the object modules, links and relocates the modules, and outputs the load module.

Symbol table definition files are special object files that can serve as input to LINK anywhere other object files are allowed.

11.3.2 Input Library Modules

The RT–11 linker can automatically search libraries. Libraries consist of library files, which are specially formatted files produced by the librarian program (described in Chapter 12) that contain one or more object modules. The object modules provide routines and functions to aid you in meeting specific programming needs. (For example, FORTRAN has a set of modules containing all necessary computational functions — SQRT, SIN, COS, and so on.) You can use the librarian to create and update libraries. Then you can easily access routines that you use repeatedly or routines that different programs use. Selected modules from the appropriate library file are linked as needed with your program to produce one load module. Libraries are further described in Chapter 12.

NOTE

Library files that you combine with the monitor COPY command or with the PIP /U or /B option are invalid as input to both the linker and the librarian.

You specify libraries in a command string in the same way as you specify normal modules; you can include them anywhere in the command string. If you are creating an overlay structure, specify libraries before you specify the overlay structure. Do not specify libraries on the same line as overlay segments. If a global symbol is undefined at the time the linker encounters the library in the input stream, and if a module is included in the library that contains that global definition, then the linker pulls that module from the library and links it into the load image. Only the modules needed to resolve references are pulled from the library; unreferenced modules are not linked.

Modules in one library can call modules from another library; however, the libraries must appear in the command string in the order in which they are called. For example, assume module X in library ALIB calls Y from the BLIB library. To correctly resolve all globals, the order of ALIB and BLIB should appear in the command line as:

```
*Z=B,ALIB,BLIB
```

Module B is the root. It calls X from ALIB and brings X into the root. X in turn calls Y, which is brought from BLIB into the root.

NOTE

LINK may place a library module in an overlay segment when the library module should be in the root. If this occurs you must relink using the /I option to avoid invalid results. Examine the load map for any library modules located in an overlay segment. The global symbols defined in these library modules should never be referenced outside of the overlay in which they were defined. (Global symbols in an overlay that are externally referenced are identified by an "@" sign in the load map.) When you relink using /I, respond to the *Library search?* prompt with the names of the symbols designated in the load map as externally referenced symbols. LINK then places the library module in the root.

Library Module Processing

The linker selectively relocates and links object modules from specific user libraries that were built by the librarian. Figure 11-1 diagrams this general process. During pass 1 the linker processes the input files in the order in which they appear in the input command line. If the linker encounters a library file during pass 1, it takes note of the library in an internal save status block, and then proceeds to the next file. The linker processes only non-library files during the initial phase of pass 1. In the final phase of pass 1 the linker processes only library files. This is when it resolves the undefined globals that were referenced by the non-library files.

The linker processes library files in the order in which they appear in the input command line. The default system library (SY:SYSLIB.OBJ) is always processed last.

The search method the linker uses allows modules to appear in any order in the library. You can specify any number of libraries in a link and they can be positioned anywhere, with the exception of forward references between libraries, and they must come before the overlay structure. The default system library, SY:SYSLIB.OBJ, is the last library file the linker searches to resolve any remaining undefined globals.

Some languages, such as FORTRAN, have an Object Time System (OTS) that the linker takes from a library and includes in the final module. The most efficient way to accomplish this is to include these OTS routines (such

(up to five digits in length) specifying the overlay region to which the module is assigned. The /O option must follow (on the same line) the specification of the object modules to which it applies, and only one overlay region can be specified on a command line. Overlay regions cannot be specified on the first command line; that is reserved for the root segment. You must use /C or // for continuation.

You specify co-resident overlay routines (a group of subroutines that occupy the overlay region and segment at the same time) as follows:

```
*OBJA,OBJB,OBJC/O:1/C
*OBJD,OBJE/O:2/C
  .
  .
  .
```

All modules that the linker encounters until the next /O option will be co-resident overlay routines. If you specify, at a later time, the /O option with the same value you used previously (same overlay region), then the linker opens up the corresponding overlay area for a new group of subroutines. This group occupies the same locations in memory as the first group, but it is never needed at the same time as the previous group. The following commands to the linker make R and S occupy the same memory as T (but at different times):

```
*MAIN,LP:=ROOT/C
*R,S/O:1/C
*T/O:1
```

The following example establishes two overlay regions.

```
*OUTPUT,LP:=INPUT//
*OBJA/O:1
*OBJB/O:1
*OBJC/O:2
*OBJD/O:2
*//
```

You must specify overlays in ascending order by region number. For example:

```
*A=A/C
*B/O:1/C
*C/O:1/C
*D/O:1/C
*G/O:2
```

The following overlay specification is invalid since the overlay regions are not given in ascending numerical order. An error message prints in each case, and the overlay option immediately preceding the message is ignored.

```
*X=LIBR0//
*LIBR1/O:1
*LIBR2/O:0
?LINK-W-/O or /V option error, re-enter line
*
```

In the above example, the overlay line immediately preceding the error message is ignored, and should be re-entered with an overlay region number greater than or equal to one.

11.5.13 Library List Size Option (/P:n)

The /P:n option lets you change the amount of space allocated for the library routine list. Normally, the default value allows enough space for your needs. It reserves space for approximately 170 unique library routines, which is the equivalent of specifying /P:170. (decimal) or /P:252 (octal). See the *RT–11 Installation and System Generation Guide* for details on customizing this default number for the library routine list.

The error message *?LINK-F-Library list overflow, increase size with /P* indicates that you need to allocate more space for the library routine list. You must relink the program that makes use of the library routines. Use the /P:n option and supply a value for *n* that is greater than 170.

You can use the /P:n option to correct for symbol table overflow. Specify a value for *n* that is less than 170. This reduces the space used by the library routine list and increases the space allocated for the symbol table. If the value you choose is too small, the *?LINK-F-Library list overflow, increase size with /P* message prints. In the following command, the amount of space for the library routine list is increased to 300 (decimal).

```
*SCCA=RK1:SCCA/P:300.
```

11.5.14 Absolute Base Address Option (/Q)

The /Q option lets you specify the absolute base addresses of up to eight p-sects in your program. This option is particularly handy if you are preparing your program sections in Absolute Loading format for placement in ROM storage. When you use this option in the first command line, the linker prompts you for the p-sect names and load addresses. The p-sect name must be six characters or less, and the load address must be an even octal number. Terminate each line with a carriage return. If you enter only a carriage return in response to any of the prompts, LINK ceases prompting.

If you use /E, /Y, or /U with /Q, LINK processes those options before it processes /Q.

When you use the /Q option, observe the following restrictions:

- Enter only even addresses. If you enter an odd address, no address, or invalid characters, LINK prints an error message and then prompts you again for the p-sect and load address.
- /Q is invalid with /H or /R. These options are mutually exclusive.
- LINK moves your p-sects up to the specified address; moving down might destroy code. If your address requires code to be moved down, LINK

The following example creates a new library called NEWLIB.OBJ on the default device (DK:). The modules that make up this library file are in the files FIRST.OBJ and SECOND.OBJ, both on the default device.

```
*NEWLIB=FIRST,SECOND
```

12.2.4 Inserting Modules into a Library

Whenever you specify an input file without specifying an associated option, the librarian inserts the input file's modules into the library file you name on the output side of the command string. You can specify any number of input files. If you include section names (by using /P) in the global symbol table and if you attempt to insert a file that contains a global symbol or PSECT (or CSECT) having the same name as a global symbol or PSECT already existing in the library file, the librarian prints a warning message (see Section 12.2.13 for multiple definition library creation). The librarian does, however, update the library file, ignore the global symbol or section name in error, and return control to the CSI. You can then enter another command string.

Although you can insert object modules that exist under the same name (as assigned by the .TITLE statement), this practice is not recommended because of possible confusion when you need to update these modules (Sections 12.2.10 and 12.2.11 describe replacing and updating).

NOTE

> The librarian performs module insertion, replacement, deletion, merge, and update when creating the library file. Therefore, you must indicate the library file to which the operation is directed on both the input and output sides of the command line, since effectively the librarian creates a "new" output library file each time it performs one of these operations. You must specify the library file first in the input field.

The following command line inserts the modules included in the files FA.OBJ, FB.OBJ, and FC.OBJ on DT1: into a library file named DXYNEW.OBJ on the default device. The resulting library also includes the contents of library DXY.OBJ.

```
*DXYNEW=DXY,DT1:FA,FB,FC
```

The next command line inserts the modules contained in files THIRD.OBJ and FOURTH.OBJ into the library NEWLIB.OBJ.

```
*NEWLIB,LIST=NEWLIB,THIRD,FOURTH
```

Note that the resulting library (1) contains the original library plus some new modules, and (2) replaces the original library because the same name was used in this example for the input and output library.

12.2.5 Delete Option (/D)

The /D option deletes modules and all their associated global symbols from a library file's directory. Since modules are deleted only from the directory (and not from the object module itself), all modules that were previously deleted are restored whenever you update that library, unless you use /D again to delete them.

When you use the /D option, the librarian prompts:

```
Module name?
```

Respond with the name of the module to be deleted, followed by a carriage return. Continue until you have entered all modules to be deleted. Type a carriage return immediately after the *Module name?* message to terminate input and initiate execution of the command line.

The following example deletes the modules SGN and TAN from the library file TRAP.OBJ on DT3:.

```
*DT3:TRAP=DT3:TRAP/D
Module name? SGN
Module name? TAN
Module name?
```

The next example deletes the module FIRST from the library LIBFIL.OBJ; all modules in the file ABC.OBJ replace old modules of the same name in the library; it also inserts the modules in the file DEF.OBJ into the library.

```
*LIBFIL=LIBFIL/D,ABC/R,DEF
Module name? FIRST
Module name?
```

In the following example, the librarian deletes two modules of the same name from the library file LIBFIL.OBJ.

```
*LIBFIL=LIBFIL/D
Module name? X
Module name? X
Module name?
```

12.2.6 Extract Option (/E)

The /E option allows you to extract an object module from a library file and place it in an .OBJ file.

When you specify the /E option, the librarian prints:

```
Global?
```

Respond with the name of the object module you want to extract. If you specify a global name, the librarian extracts the entire module of which that global is a part. Type a carriage return to terminate the prompting for a global.

You cannot use the /E option on the same command line as another option.

Chapter 19
Error Logging

The Error Logger utility monitors the hardware reliability of the system. The Error Logger keeps a statistical record of all I/O operations that occur on any of the following devices:

DD	DX
DL	DY
DM	PD
DP	RF
DS	RK
DT	

In addition to keeping these statistics, the Error Logger detects and records memory parity or cache errors and any errors that occur during I/O operations. At intervals you determine, the Error Logger produces individual and/or summary reports on some or all of these errors. The Error Logger is available only as a special feature; that is, you must perform the system generation process to enable Error Logging. It runs only with the foreground/background or extended memory monitors, as either a foreground or system job.

19.1 Uses

Error logging reports are useful for maintaining the hardware on which RT-11 runs. Problems such as line noise, static discharges, or inherently error-prone media can cause recoverable errors on systems that are otherwise functioning normally. By studying error logging reports, you can learn to distinquish these errors from those that might be symptoms of an impending device failure. Also, some recoverable errors that are insignificant in themselves might be related to other more serious errors; their effects might not be immediately apparent to you. Information contained in the reports about each error and about the status of the system when the error occurred may alert you to a previously unforeseen hardware problem.

Sometimes a device fails so quickly that the Error Logger is unable to predict failure in time for you to prevent it. In this case, you can determine the cause more quickly if a report is available that describes the errors that occurred immediately prior to the failure.

In general, the Error Logger:

- Detects each I/O transfer and I/O error as it occurs
- Gathers information about each I/O transfer or error
- Stores the information in a file
- Formats the information to produce a report

NOTE

Because the Error Logger records data on each I/O transfer, thereby using additional computer time and memory, you may wish to use the Error Logger only when you experience difficulty with a device. Keeping a backup system volume on which the Error Logger is enabled makes this easy.

19.2 Error Logging Subsystem

The Error Logger consists of three programs and a statistics file. When you run the Error Logger, you coordinate these programs to gather I/O and error-related information into its statistics file and create the error report you want. The Error Logger names the statistics file it creates ERRLOG.DAT. At any time you specify, you can direct the Error Logger to create error reports from the information it has gathered in ERRLOG.DAT. The names and functions of the three Error Logger programs follow.

EL — A foreground or system job that gathers information about every I/O transfer and system error and stores it in ERRLOG.DAT. The EL job communicates with a device handler as each I/O transfer occurs, and it gathers from the handlers all the necessary statistics for a complete error report. When you initiate error logging, EL instructs you to start up the second Error Logging program, ELINIT.

ELINIT — A background job that creates and maintains the statistics file, ERRLOG.DAT. You can direct ELINIT to initialize ERRLOG.DAT every time you have a session at the terminal, or you can direct ELINIT to continue compiling statistics in ERRLOG.DAT on a daily basis.

When you run ELINIT, it prompts you for the information it needs to maintain ERRLOG.DAT's size. By default, ELINIT allocates 100 decimal blocks for ERRLOG.DAT. Each time you run ELINIT, it prints a message that tells how many of those 100 blocks are filled. If ERRLOG.DAT fills to its limit, the EL job is unable to store more information in it. So that you can increase ERRLOG.DAT's size, ELINIT prompts you for a file size change each time you run the program.

If you rearrange your RT-11 configuration, or if you do something to alter the device handlers, ELINIT may print a message indicating that it must initialize ERRLOG.DAT to make the statistics it has been maintaining compatible with the new configuration. When this happens, ELINIT renames the ERRLOG.DAT it formerly maintained to ERRLOG.TMP and creates a new ERRLOG.DAT. The Error Logger can still create a report from ERRLOG.TMP.

ERROUT A background job that creates a report from the statistics in ERRLOG.DAT or any file of that format. When you run ERROUT, you can direct the program to list the error report at the terminal or to create a file for the error report. You can also indicate whether you want a detailed report on each error that occurred or simply a summary report.

Figure 19–1 provides a diagram of the Error Logging subsystem.

Chapter 24
Source Language Patch Program (SLP)

The Source Language Patch Program (SLP), is a patching tool you can use for maintaining source files that exist on any RT–11 device.

SLP accepts as input a source file you wish to patch and a command file that you create when you compare two source programs using the source compare program, SRCCOM, described in Chapter 15. When you use SLP along with the SRCCOM command file, you can quickly and easily patch one version of a source program to match another version.

Chapter 15, Source Compare Program (SRCCOM), describes the procedure you can use to create a patch command file that is suitable for input to SLP.

24.1 Calling and Using SLP

To call SLP from the system device, type the following in response to the keyboard monitor dot (.):

```
R SLP<RET>
```

The Command String Interpreter prints an asterisk (*) at the left margin of the terminal and waits for a command string. If you enter only a carriage return in response to the asterisk, SLP prints its current version number. You can type CTRL/C to halt SLP and return control to the monitor when SLP is waiting for input from the console terminal. To restart SLP, type R SLP or REENTER in response to the monitor's dot. Chapter 6, Command String Interpreter, describes the general syntax of the command line that SLP accepts.

Enter a command line according to this general syntax:

```
[outfil][,listfil]=infil,comfil/[option...]
```

where:

outfil	represents the updated source file. The default file type is .MAC
listfil	represents the listing file. When you specify this file, SLP creates a numbered listing of the updates SLP made to the source file. The default file type is .LST
infil	represents the source file you want SLP to update. The default file type is .MAC

comfil represents the command file that contains the commands for updating the source file. The default file type is .MAC. You can create this file by using SRCCOM with the /P option

/option represents one of the options listed in Table 24-1

Although either output files can be omitted, you must use one or both.

24.2 Options

Table 24-1 lists the options you can use in the command line.

Table 24-1: SLP Options

Option	Function
/A	Disables audit trail generation. The audit trail is a string of characters that SLP appends to the end of each updated line in the output files. The audit trail keeps track of the update status of each line in the output file. You can use the /A option if you do not want SLP to use the audit trail in both the updated source file and the listing file.
/B	Inserts spaces instead of tabs between the source line and the audit trail.
/D	Creates a double-spaced listing. When you use this option, SLP double-spaces between the lines in a listing file.
/L:n	Specifies the size of the source line, where *n* represents the maximum number of characters you want in the source line. The default buffer size for formatting lines is 200 (decimal) bytes. If you expect the size of the command lines or source lines to be greater than what can fit in the line buffer, you can use this option to change the buffer size. SLP interprets the number you specify for *n* as an octal number; if you enter a decimal number, use a decimal point. The line buffer must be as least as long as the sum of the column number where the audit trail begins and the number of characters in the audit trail.
/P:n	Specifies the start column of the audit trail, where *n* represents the column number in which you want the audit trail to start. If the number you specify for *n* is decimal, be sure to use a decimal point after the number. By default, SLP starts the audit trail in column 73 (decimal). If a source line extends beyond the column where the audit trail begins, the audit trail can overstrike the source line. If you use the /P:n option, you start the audit trail in any tab stop column. SLP rounds up the number you specify to the nearest tab stop column. If, for example, you specify 46 for n, SLP rounds this number to 49.
/S:n	Specifies size of the audit trail, where *n* represents the number of characters you want in the audit trail. If the number you specify is decimal, be sure to use a decimal point after the number. The default number of characters in the audit trail is 12 (decimal). The maximum number of characters you can specify for the audit trail is 16 (decimal).
/T	Retains trailing blanks and tabs in the input source file. By default, SLP removes spaces and tabs that appear at the end of lines in the input source file.

24.3 Example

This section uses SLP to patch the source file, ANTONY.MAC, so that it matches the source file CAESAR.MAC. CAESAR.MAC consists of the following lines.

```
FRIENDS, ROMANS, COUNTRYMEN!
LEND ME YOUR EARS!
I COME TO BURY CAESAR,
NOT TO PRAISE HIM.
THE EVIL THAT MEN DO
LIVES AFTER THEM.
THE GOOD IS OFT INTERRED
WITH THEIR BONES;
SO LET IT BE WITH CAESAR!
```

The file this example will patch, ANTONY.MAC, follows.

```
FRIENDS, ROMANS, COUNTRYMEN!
LEN ME YOUR EARS!
I COME TO BURY CAESAR,
NOT TO PRAISE HIM.
THE EVIL THAT MAN DO
LIVES AFTER THEM.
THE GOOD IS OFT ENTERED
WIT THEIR HOMES;
SO LET IT BE WITH CAESAR!
```

Using the /P option in SRCCOM, this example obtains a command file, CAESAR.DIF. CAESAR.DIF contains the necessary commands to make ANTONY.MAC match CAESAR.MAC. The following command line directs SLP to patch ANTONY.MAC so that it matches CAESAR.MAC.

```
.R SLP
*ANTONY,ANTONY=ANTONY,CAESAR
```

After executing the command above, SLP assigns a .BAK file type to the input file ANTONY.MAC. It assigns .MAC file type to the updated source file. SLP has also created a listing of ANTONY.MAC that lists each line by number and appends an audit trail to each new line. The updated file, ANTONY.MAC, is now identical to CAESAR.MAC. ANTONY.LST appears below.

```
SLP V04.00                          ANTONY,ANTONY=ANTONY.MAC,CAESAR,DIF

     1.  FRIENDS, ROMANS, COUNTRYMEN!          ;**NEW**
     2.  LEND ME YOUR EARS!                    ;**-1
     3.  I COME TO BURY CAESAR,
     4.  NOT TO PRAISE HIM.                    ;**NEW**
     5.  THE EVIL THAT MEN DO                  ;**-1
     6.  LIVES AFTER THEM.                     ;**NEW**
     7.  THE GOOD IS OFT INTERRED              ;**NEW**
     8.  WITH THEIR BONES;                     ;**-2
     9.  SO LET IT BE WITH CAESAR!
```

Note that when SLP updates a line, it appends an additional audit trail below the audit trail of the updated line. The additional audit trail keeps

track of the number of consecutive lines that have been updated. In ANTONY.LST, above, note the audit trails ;**-1 and ;**-2.

24.4 Creating and Maintaining a Command File

SLP is a line-oriented patching tool. That is, you make changes to entire lines, and not to individual or strings of characters within a line. If you want to change only a few characters within a line, it will be necessary for you to enter a new line.

Although DIGITAL recommends you use the SRCCOM /P option to create the SLP input command file, you can use any RT-11 editor to create it yourself. The section that follows describes the commands, or operators, you use to create the command file. This procedure is tedious, however, and in most cases unnecessary. But for completeness, this procedure is included with this chapter. Table 24-2 lists the commands, or operators, you enter into the command file.

The section ends with a description of various line manipulations that SLP can effect.

Table 24-2: SLP Command File Operators

Operator	Explanation
-	Indicates the start of an update.
\	Disables the audit trail. Note that this operator must appear on a line by itself. If it appears in the first column of a line with additional information following it, the audit trail will be disabled, but the rest of the command line will be ignored. If used in any column other than column one, a syntax error occurs.
%	Enables the audit trail.
/	Indicates the end of an update or series of updates; it appears as the last character in the command file.
<	Serves as an escape character for characters SLP would otherwise interpret as operators. For example, if you want to include a slash (/) in a source file, type the less-than character (<) before the slash. Then, SLP will not interpret the slash as an operator. You can use the less-than character as an escape character for all SLP command file operators.

24.4.1 Update Line Format

The general format of the SLP command file update line follows.

```
-locator1,[locator2],[/audit trail/][;]
input line
   .
   .
   .
```

where:

- indicates that this is an update line

locator1 represents a character string that serves as a line locator. SLP moves the line pointer to the line specified by the line locator. You can specify this line locator with any of the locator forms described below

locator2 represents a character string that, when used with *locator1*, defines a range of lines you want to delete or replace. You can specify this line locator with any of the locator forms described below. You cannot define a range of lines in a backwards direction; the line referenced by locator2 must occur in the source file after the line referenced by locator1

/audit trail/ represents a character string you use as an audit trail. SLP appends the audit trail to the right of each updated line. You must delimit the audit trail with slashes (/)

inputline represents a line of new text that SLP inserts into the file immediately following the current line. You can enter as many input lines as you want

; is an optional command line terminator

All fields in the update line are positional. That is, if you specify only locator1 and an audit trail, you must use two commas between those two fields. If you want to specify only the audit trail, you must precede the audit trail with two commas.

The update lines in a command file must edit the source file in a forwards direction, from beginning to end. Each locator1 must point to a line that appears in the source file before the lines pointed to by any succeeding locator1.

The line locators can take one of the following forms:

```
/string/[+n]
/string...string/[+n]
number[+n]
.+n
```

where:

/string/[+n] represents an ASCII character string. You must delimit any string you enter with slashes. SLP locates the first occurrence of this string, and moves the line pointer to the line that contains that string. *+n* represents the offset from the line that contains the string. You must use the plus character (+) with the n notation

/string...string/[+n] represents an ASCII character string. SLP locates the line in which the two strings delimit a larger string. Use the ellipsis (...) in this locator form to separate the two strings. *+n* represents the offset from the line specified by the string...string locator

number[+n]	represents the line number to which SLP is to move the line pointer. *+n* represents the offset from the line specified by number
.+n	represents the offset from the current line pointer. SLP interprets the period (.) as the current line pointer location, and the *+n* as the offset from it. You must use the plus character (+)

24.4.2 Creating a Numbered Listing

You can use SLP to create a numbered listing of the input source file. In creating a command file, you should use a numbered listing when you prepare command input. To generate a numbered listing, enter the following lines:

```
R SLP
*,listfile=infile
```

Listfile represents the listing file SLP produces, and *infile* represents the input source file. Here is a file, PROG.MAC, from which SLP is to create a numbered listing:

```
        .TITLE  PROG.MAC   VERSION 1
        .MCALL  .TTYOUT, .EXIT, .PRINT

EXP:    .PRINT  #MESSAG
        MOV     #N,R5
FIRST:  MOV     #N+1,R0
        MOV     #A,R1
```

The following command line creates a numbered listing, PROG.LST of the file above, PROG.MAC:

```
*,PROG=PROG
```

After SLP processes the command above, it produces the following listing of PROG.MAC:

```
SLP  V04.00                          ,PROG=PROG.MAC

     1.             .TITLE  PROG.MAC   VERSION 1
     2.
     3.             .MCALL  .TTYOUT, .EXIT, .PRINT
     4.
     5.   EXP:      .PRINT  #MESSAG
     6.             MOV     #N,R5
     7.   FIRST:    MOV     #N+1,R0
     8.             MOV     #A,R1
```

24.4.3 Adding Lines to a File

To add lines to a file, enter in the command file one of the three locator forms below:

```
-number
-.[+n],,[/audittrail/]
-/string/
```

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

CSD SOFTWARE PUBLICATIONS ML 5-5/E45
DIGITAL EQUIPMENT CORPORATION
146 MAIN STREET
MAYNARD, MASSACHUSETTS 01754

Do Not Tear - Fold Here

Cut Along Dotted Line