RT–11 Commands Manual

Order Number AA–PDU0A–TC

August 1991

This manual tells you how to use RT–11 DCL commands. If you are unfamiliar with RT–11, you should read the *Introduction to RT–11* before using this manual.

Revison/Update Information: This manual supersedes the *RT–11 System User's Guide*, AA–5279E–TC.

Operating System: RT–11 Version 5.6

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, August 1991

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.

Any software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013.

© Digital Equipment Corporation 1991
All rights reserved. Printed in U.S.A.

The Reader's Comments form at the end of this document requests your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation: CTS–300, DDCMP, DECNA, DECnet, DECUS, DECwriter, DEQNA, DEUNA, DIBOL, Ethernet, MASSBUS, MicroPDP–11, Micro/RSX, PDP, Professional, Q-bus, RSTS, RSX, RT–11, RTEM–11, UNIBUS, VMS, VT, and the DIGITAL logo.

S1479

This document was prepared using VAX DOCUMENT, Version 1.2.

Contents

Part I Using RT–11 Commands

Chapter 1 How to Use DCL Commands

Chapter 2 Using Special-Function Keys

Chapter 3 Using Files, Volumes, and Devices

Part II RT–11 Command Descriptions

Appendix A SL Command-Line Editing Functions

Appendix B Printer Support for the SETUP Command

Appendix C National Replacement Character Sets

Index

Figures

Tables

Preface

Document Structure

This manual is divided into the following two parts:

- Part I, Using RT–11 Commands, describes how to use the RT–11 command language.
- Part II, Command Descriptions, alphabetically describes individual RT–11 commands.

Audience

This manual, written for users of the RT–11 operating system, describes the DCL commands used by the operating system.

Conventions

The following conventions are used in this manual.

Convention	Meaning

Black print	In examples, black print indicates output lines or prompting characters that the system displays. For example:
	`.BACKUP/INITIALIZE DL0:F*.FOR DU1:WRK`
	`Mount output volume in DU1:; continue? Y`

Red print	In examples, red print indicates user input.
Braces ({ })	In command syntax examples, braces enclose options that are mutually exclusive. You can choose only one option from the group of options that appears in braces.
Brackets ([ ])	Square brackets in a format line represent optional parameters, qualifiers, or values.

Convention	**Meaning**
[/option1 /option2]	Square brackets in an option summary surrounding two or more options mean that you can specify any one of the enclosed options or any combination of them. The following example means you can specify either the /BEFORE or the /SINCE option or a combination of both to express one or two ranges of dates: [/BEFORE[:date] /SINCE[:date]]
Bold options	**Bold** options in an option summary indicate default options; that is, the option that RT–11 uses if you do not specify any choice of action.
lowercase characters	In command syntax examples, lowercase characters represent elements of a command for which you supply a value. For example: `DELETE filespec`
UPPERCASE characters	In command syntax examples, uppercase characters represent elements of a command that should be entered exactly as given.
RET	RET in examples represents the RETURN key. Unless the manual indicates otherwise, terminate all commands or command strings by pressing RET.
RETURN	RETURN in the text represents the RETURN key.
CTRL/*x*	CTRL/*x* indicates a control-key sequence. While pressing CTRL key, press another key. For example: CTRL/C

Associated Documents

Basic Books

- *Introduction to RT–11*
- *Guide to RT–11 Documentation*
- *PDP–11 Keypad Editor User's Guide*
- *PDP–11 Keypad Editor Reference Card*
- *RT–11 Quick Reference Manual*
- *RT–11 Master Index*
- *RT–11 System Message Manual*
- *RT–11 System Release Notes*

Installation Specific Books

- *RT–11 Automatic Installation Guide*
- *RT–11 Installation Guide*
- *RT–11 System Generation Guide*

Programmer Oriented Books

- *RT–11 IND Control Files Manual*
- *RT–11 System Utilities Manual*
- *RT–11 System Macro Library Manual*
- *RT–11 System Subroutine Library Manual*
- *RT–11 System Internals Manual*
- *RT–11 Device Handlers Manual*
- *RT–11 Volume and File Formats Manual*
- *DBG–11 Symbolic Debugger User's Guide*

New and Changed Features

This manual is a revised edition of the *RT–11 System User's Guide*. The name has been changed, since some introductory and utility information has been removed and since the manual's contents are primarily RT–11 commands.

New Features

- The manual has been updated, rewritten, and reorganized in content, format, and style.
- SET and SETUP command descriptions have been alphabetically incorporated in the command descriptions.
- The latest changes in the operating system and the previous release notes have been incorporated in the manual.
- New RT–11 Version 5.6 commands:

  LINK/IDSPACE

  SET CLI   [NO]DCL, [NO]CCL, [NO]UCL, [NO]UCF

  SET LS   LINE=n

  SET LS   [NO]MTTY

  SET MODE   [NO]SJ

  SET NL   [NO]TOY

  SET RUN   [NO]VBGEXE

  SET XL   LINE=n

  SET XL   [NO]MTTY

  V/VRUN

  See the individual command descriptions for the command meanings.

Changed Features

The GT command is no longer supported by RT–11 and no longer documented in this manual.

This manual has been changed to contain primarily DCL commands and material relating to those commands.

- For general operating-system introductory information that used to be in the *RT–11 System User's Guide*, see the *Introduction to RT–11*.

- For RT–11 procedural information; that is, for how to do general tasks with RT–11, see the *Introduction to RT–11*.
- For how to create IND control files, command files written in the IND command language and including DCL commands as part of them, see the *RT–11 IND Control Files Manual*.
- For CSI commands, for DCL and CSI command equivalents, and for utility program descriptions, see the *RT–11 System Utilities Manual*.

Part I
Using RT–11 Commands

Part I describes how to use DCL commands. RT–11 DCL (Digital Command Language) commands are also called KMON (Keyboard Monitor) commands, since KMON is the program that interprets them.

For further information on how to use RT–11, see the following manuals:

- *Introduction to RT–11*

  Using the command-line editor
  Defining your own commands
  Using command files
  Setting up a system start-up command file (STRTxx.COM)
  Doing common system tasks

- *RT–11 System Utilities Manual*

  Using CSI (Command String Interpreter) commands
  and CCL (Concise Command Language) commands

- *RT–11 IND Control Files Manual*

  Creating and executing control files

Chapter 1

How to Use DCL Commands

This chapter explains how to use DCL commands. For more information, use the on-line INDEX or the *RT–11 Master Index*, or see the *Introduction to RT–11*. That manual covers in more detail many topics that are only briefly described here.

DCL Commands Defined

The Digital Command Language (DCL) is a standard interface between a user and a Digital operating system. DCL as implemented on the RT–11 operating system is also called the KMON (keyboard monitor) command language, because you issue DCL commands at the keyboard and because the KMON monitor interprets the commands.

A DCL command is an instruction to the computer enabling you to control the computer from the keyboard. Most DCL commands are English-like words that symbolize the action you want done. For example, DIRECTORY, PRINT, SHOW, and TIME are DCL commands.

DCL Command Parameters Defined

A command's parameters are the command's input; that is, the information, file, or files on which the command is to act. Most DCL commands act on files, but some act on storage locations in your processor's memory. For example, the PRINT and COPY commands print and copy files, but the DATE and TIME commands modify memory locations holding the current time and date.

How to Type a Command with Its Parameter(s)

If you do not correctly type a command with its parameter(s), you can get partial execution of the command (depending on the command issued) or an error message with no command execution.

Correct DCL Command Format

The general DCL command format or syntax is as follows. The specific format varies slightly depending on the command.

command[/option...] parameter

where:

command is the name of the command you want to execute.

/option... is one or more command qualifiers that modify the command.

parameter(s) is a file or files on which to act or data to be used by the command.

or

command[/option...] in-filespec[/option...] out-filespec[/option...]

where:

command is the name of the command you want to execute.

/option is a command qualifier that modifies the command.

in-filespec is the file you want to access.

/option is a file qualifier adjusting the command's action on the input file. A file option applies only to the file it qualifies, whereas a command option applies to all files the command executes.

out-filespec is the file that is to receive the results of the operation.

/option is a file qualifier adjusting the command's action on the output file.

Square brackets specify optional elements in a command line. A line such as [NO]QUERY specifies that there can be two mutually exclusive options: QUERY and NOQUERY.

Conventions Used to Describe Command Format

The preface to this manual contains a detailed list of the conventions used throughout the manual when describing command formats and commands. You should familiarize yourself with these conventions and their meanings before reading further in the manual.

Conventions Used to Describe File Specifications

You name or specify a file with a file specification, called a *filespec*. This consists of a device name, the file name, and the file type in the following format. A colon (:) separates the device name from the file name, and a period (.) separates the file name from the file type.

dev:filnam.typ

where:

dev: is either a logical-device name or a physical-device name, which is a one- to three-character name. You can omit this name if the file is on your default storage device.

filnam is a one- to six-character alphanumeric name of the file.

.typ is a zero- to three-character alphanumeric file type.

You can omit the file type in some file specifications, if your file has the default file type RT–11 assumes for the operation. For example, the PRINT command assumes a LST file type and the MACRO command assumes a MAC file type for input files. Many commands, however, do not assume a file type.

Issuing Commands

A prompt character, a period at the left margin of the terminal screen or printer, appears whenever KMON is ready for you to type a command. Usually command examples begin with that prompt character.

After typing a command, you *issue* or execute it by pressing RETURN. This is sometimes indicated in a command example by the abbreviated RET symbol.

Most of the command descriptions in this manual presume that you press RETURN to issue a command. However, the *Introduction to RT–11*, a tutorial on how to use the operating system, includes the abbreviated RET symbol in all its command examples.

Getting Help with Issuing Commands

For on-line command descriptions, issue the HELP command. The command HELP * displays a summary listing of all DCL commands with a description of each. See the HELP description in this manual for more information.

If you get an error message that you do not understand, see the *RT–11 System Message Manual*. That manual lists each error message in alphabetical order with an explanation of each and how to deal with the situation.

Issuing Incomplete Commands

RT–11 accepts commands as either a complete string containing all information necessary to execute a command or a partial string. In the latter case, RT–11 prompts you to supply the rest of the information. For example, if you issued the command PRINT without any file parameters, RT–11 would prompt *Files?* and wait for you to type in what you wanted printed:

```
.PRINT  RET
Files?
```

Continuing Commands on More Than One Line

You can use a hyphen at the end of a command line to continue the command to the next line. However, the entire command line, including wildcard file names and types and default devices, must include no more than 80 characters. For example, the following DCL command copies three input files to the output file DK:OUTFIL.TXT:

```
.COPY/CONCATENATE MYFIL1.TXT,RK0:MYFIL2.TXT,-
MYFIL3.TXT DK:OUTFIL.TXT
```

Using the Type-Ahead Feature When Issuing a Command

The monitor has a type-ahead feature that lets you enter terminal input while a program is executing. For example, you can issue the DIRECTORY command and while the files are being displayed on your terminal, you can issue another command; for example, DATE. RT–11 stores your second command in a buffer and executes it when RT–11 completes the first operation. As soon as the directory listing is completed, RT–11 will execute the DATE command by displaying the date.

If type-ahead input exceeds the input buffer capacity (usually 134 characters), the terminal bell rings and RT–11 accepts no characters until a program uses part of the type-ahead buffer, or until you delete characters. Any input typed after the terminal bell rings is lost. Type-ahead is particularly useful when you issue multiple command lines to system programs.

Note that after you bootstrap any RT–11 monitor, RT–11 does not recognize the type-ahead feature until either the keyboard prompting character (.) displays or the start-up command file begins executing. If you type ahead before this, RT–11 either ignores or truncates your input.

If you press a single CTRL/C while typing ahead, RT–11 puts CTRL/C into the buffer. The program currently executing exits when it makes a terminal input request. Pressing a double CTRL/C returns control to the monitor immediately. If you terminate a job by pressing CTRL/C twice, RT–11 discards any unprocessed type-ahead input.

Aborting or Canceling a Command

If you change your mind about issuing a command, you can either abort or cancel the command.

- You abort the current command that is executing by pressing CTRL/C twice. This also dumps any commands in the type-ahead buffer.
- You cancel a command you are currently typing in by pressing CTRL/C or CTRL/U once.
- You exit a program that is waiting for input by pressing CTRL/C once.

Running Jobs in a Completely Virtual Environment

Many RT–11 commands can automatically run jobs in a completely virtual environment. For example, the LINK, MACRO, R, and RUN commands can do so. This often enables programs to run faster and use less low-memory space than is otherwise required. For further information on this topic, see the SET RUN and V/VRUN command descriptions in this manual and the Completely Virtual Mapping Environment section in Chapter 3 of the *RT–11 System Internals Manual*.

Abbreviating Command Lines

You can abbreviate DCL commands, command options, and command parameters.

Abbreviating Commands and Options

You can abbreviate DCL commands to four or fewer characters, if there is no ambiguity. For example, the command DIRECTORY and DIR both display a directory listing of all the files on your default storage device. The command PR, however, is an ambiguous command since it can refer to either PRINT or to PROTECT. So, to abbreviate either of these two commands, you must type either PRI or PRO.

You can also abbreviate DCL (command or file) options to four or fewer characters, if there is no ambiguity. For example, the BACKUP /DEVICE option can be abbreviated to /DE to distinguish it from the BACKUP /DIRECTORY option.

Abbreviating File-Specification Parameters

You can abbreviate file specifications in three ways:

- By using system defaults
- By *factoring* multiple file specifications
- By substituting *wildcards* for characters

Using System Defaults

You do not have to specify a device in your file specification, if you want RT–11 to look for your file on your default storage device. So, by assigning (see the ASSIGN command) the device you want to be your default storage device, you can save yourself some typing. Whenever a device is not specified in the examples in this manual, the default storage device is presumed.

You also do not have to specify a file type, if the file you want to specify has a file type the command you want to use recognizes. These are called *default file types* for a command. See the individual command descriptions to know if a command has default input and/or output file types for its parameter(s) and what those file types may be.

The following MACRO command, for example, assembles the file PROG1.MAC on the default storage device:

```
.MACRO PROG1
```

Factoring Multiple File Specifications

If you want to specify several files on the same device, you can use factoring. That is, you enclose in parentheses the part of a multiple file specification that differs. This part can be multiple file names, multiple sections of a file name, multiple file types, or multiple sections of a file type. For example, the following two command lines are equivalent:

```
.PRINT DU1:MEMO(1,2,30).TXT
```

```
.PRINT DU1:MEMO1.TXT,DU1:MEMO2.TXT,DU1:MEMO30.TXT
```

In this factoring example, the device name outside the parentheses applies to each file specification inside the parentheses.

Factoring is useful for complicated command lines. However, RT–11 internally expands the command and interprets the command as if you had typed in all the characters. The command string that results from the command expansion must not exceed 80 characters. So, if you use six-character file names and you also use factoring, it is recommended that you specify only five files in a command line.

Substituting Wildcards for Characters

When you want a command to act on a number of files with similar file specifications, you can usually abbreviate the specifications by substituting wildcards; that is, you substitute the wildcard for any part of a file specification that is the same for all the specified files.

Most DCL commands that can act on more than one file at a time accept wildcards, though not all do. See the command descriptions for the exceptions.

RT–11 has two wildcard symbols: the asterisk (*) and the percent sign (%).

- **Abbreviating File Specifications with Asterisk Wildcards**

  You can use an asterisk (*) in a file specification to replace the following:

  — A file name

  ```
  The command DIR *.MAC lists all files having a MAC file type.
  ```

  — A file type

  ```
  The command DIR TEST.* lists all files having a TEST name.
  ```

  — Both a file name and a file type (this means *all files*)

  ```
  The command DIR *.* lists all files.
  ```

  — Any number of characters, including zero characters, in the input file name or file type

  ```
  The command DIR A*B.MAC lists all files whose name begins with A,
  ends with B, and has a file type of MAC. Even a file with the name
  AB.MAC (a file with zero characters between the A and B), if it
  exists, is listed.
  ```

  RT–11 ignores the contents of the wildcard field and selects all the files that match the remaining field or fields in the specification.

- **Abbreviating File Specifications with Percent Wildcards**

  You can use the percent (%) wildcard to substitute for a single character in the input file name or file type. For example, the following command displays all the files on the system device that have a two-letter file name that begins with a D and have a MAC file type:

  ```
  .DIR SY:D%.MAC
  ```

  Note that the % wildcard is not valid in output file specifications, though the * wildcard by itself (with no file specification) is valid as an output specification. For further information on using wildcards, see the description of the command you want to use. See also the SET WILD command.

Substituting Symbols for Command Strings

The LET utility enables you to substitute single characters for a command string. KMON then translates that character as the command string. However, every time you type that substitution character, you no longer get that character; rather, you get the command for which it stands.

To substitute symbols for commands, see the LET utility description in the *RT–11 System Utilities Manual* and the SET SL command description in this manual.

Editing Command Lines

If you make a typing mistake, you can correct it:

- By pressing the DELETE key ⌫ once to delete the single character to the left of the cursor.

  or

- By pressing CTRL/U; that is, while holding down the CTRL key, press the U key to delete all the characters to the left of the cursor.

If you want easier command-line editing, you should install the SL command-line editor.

Simplifying Your Editing with the SL Command-Line Editor

The SL (Single-Line) editor is the RT–11 command-line editor. This editor enables you to store, recall, manipulate, and edit command lines. Using SL saves you time, because SL streamlines the command-issuing process. See the chapter on using the SL editor in the *Introduction to RT–11*.

SL stores previous commands in a 512-byte buffer. Under any RT–11 monitor, you can continuously press ↑ to scroll through those previous commands. You then press RETURN to issue any of those previous commands.

Under a mapped monitor, you can use the SL command RECALL/ALL to display the stored commands. You can then recall a specific command by using either the number displayed next to each command or the shortest unambiguous abbreviation for that command. You then press RETURN to issue that recalled command.

You can repeatedly edit, compile, link, and run a program (issue a set of the same commands). Using SL, you can sequentially display and repeat each command in the set without having to retype them.

Enabling the Command-Line Editor

SL (the command-line editor) is normally installed automatically when you boot your operating system. If, for some reason, it is not installed on your system, you can install it with the following command. See the *Introduction to RT–11* if you have any questions.

```
.INSTALL SL
```

You must enable SL after it is installed. You do that with the command:

```
.SET SL KMON,ON
```

or

```
.SET SL KMON,KED,RECALL,ON
```

The last command gives you full SL functionality, while the shorter SET command gives you only basic SL functionality. See the SET SL command for more information.

Basic SL Editing of the Command Line (SET SL KMON,ON)

Basic SL command-line editing, enabled with the SET SL KMON,ON command, generally involves two types of operations:

- You use the arrow keys to move your cursor through the command line and into and out of the stack of previous commands.
- You can press CTRL/A to switch between INSERT and REPLACE modes.

  As SL is distributed, any character you type within a command line is added; that functionality is called INSERT mode. You can change that functionality while editing a command line so that any character you type replaces the character located at the cursor position; that is called REPLACE mode. You alternate between INSERT and REPLACE modes by pressing CTRL/A. SL returns to INSERT mode each time you press RETURN to issue a command.

Press PF2 to display a screen of helpful SL information on the top half of your screen (above the command line). If you first issue the command SET SL LEARN, subsequently pressing PF2 causes the SL help screen to be locked onto the display. Issue the command SET SL NOLEARN to remove the help screen.

See Table A–1 for the basic SL command-line editing functions.

Keypad SL Editing of the Command Line (SET SL KED,ON)

SL keypad editing lets you use the keypad to edit the command line. This simplifies the editing. See Table A–2 for the keypad SL command-line editing functions. See also Section A.3 for SL's full editing capabilities with a mapped monitor.

Creating Command, Control, and BATCH Files

RT–11 has three types of files that issue commands to the operating system:

- Command files—with a default COM file type
- Control files—with a default COM file type (also called IND files)
- BATCH files—with a default BAT file type

These files are called *indirect* files since, once you execute them, they issue the commands contained in them, rather than you issuing the commands. In this way, you indirectly execute the commands contained in the files.

While control and BATCH files provide the most functionality, they are more complex to use than command files.

Command Files

You can group a collection of DCL commands that you want to execute sequentially into a command file.

Command files are best suited for:

- Tasks that require a significant amount of computer time and do not require your supervision or intervention.
- Any series of commands that you are likely to type often.
- Setting up your operating system when you boot it.

  Each monitor invokes its own startup command file when you bootstrap the operating system, and you can modify these files to perform standard system configurations.

Command files can contain DCL commands, commands to programs (CCL (Concise Command Language) or CSI (Command-String Interpreter language) commands), and/or valid commands you create using the UCL (User Command Linkage) or the UCF (User Command First) command processor. See the *Introduction to RT–11* to create and use command files, startup command files, and files with your own commands. To use utility program commands, see the *RT–11 System Utilities Manual*.

An example of a command file is:

```
SET SL SYSGEN
INSTALL SL
SET SL RECALL, KED, KMON
```

The monitor startup command files that come with your operating system are other examples of command files. These files are called STRTxx.COM with the *xx* being the abbreviation of the monitor controlling the operating system; for example, STRTXM.COM.

Using a Command File As a Command Parameter

You can abbreviate a command by supplying the at command (@) with a command file as the command's parameter; for example:

```
.LINK @MYFILS.COM
```

When you embed the at command (@) within a DCL command, KMON does an append substitution. That is, in the preceding example, the first line and then any subsequent lines in the file MYFILS.COM are substituted as the object of the LINK command.

Control Files

Control files are files containing commands to IND (the INDirect control file processor). These files can also contain DCL commands.

To create and execute control files, see the *RT–11 IND Control Files Manual*.

BATCH Files

BATCH files contain BATCH commands processed by the BATCH compiler so they can be run by the BATCH run-time handler. See the BATCH appendix in the *RT–11 System Utilities Manual* to create and execute BATCH files.

A Sample Command File

The following example command file is taken from Using the Virtual Memory (VM) Device As the System Device in the *Introduction to RT–11*. See that chapter to use the VM device and to use this command file.

```
! Edit this file to suit your situation.
! VM.COM, created dd-mmm-yy, edited dd-mmm-yy.
! Configures the VM (virtual memory) working system disk.
! Substitute the source device for ddn.
! Substitute the source device handler for ddx.
!
UNLOAD VM
REMOVE VM
SET VM SIZE=0
SET VM BASE=5000
INSTALL VM
LOAD VM
INIT/NOQ VM:
COPY ddn:SWAP.SYS VM:
COPY ddn:RT11XM.SYS VM:
COPY ddn:(VMX,ddx,XLX,LSX).SYS VM:
COPY ddn:(LDX,SLX,SPX).SYS VM:
COPY ddn:(DUP,DIR,PIP,KEX,RESORC,SETUP).SAV VM:
COPY ddn:(VBGEXE,VTCOM,SPOOL).SAV VM:
COPY ddn:STARTV.COM VM:STARTX.COM
COPY/BOOT VM:RT11XM.SYS VM:
```

How to Make a Bootable Volume

The preceding command file shows you one way of making a bootable VM device. You can modify that command file to make any volume bootable.

To make a bootable volume, copy the files listed in Table 1–1 to that volume and then copy the boot code to the boot blocks on the volume.

Table 1–1: Files Needed to Make a Bootable Volume

File	Description
RT11xx.SYS	One monitor—Can be any RT–11 monitor except the RT11AI (Automatic Installation) or RT11MT (Magnetic Tape for use with MDUP) monitors
SWAP.SYS	External monitor swap blocks
ddx.SYS	Device handler for your working system volume
ddx.SYS	Device handler for your auxiliary volume (the volume you want to boot), if your auxiliary volume is not of the same type as your working system volume
SLx.SYS	Command line editor (optional)
LDx.SYS	Logical disk handler, if you intend to use logical disks (optional)

Table 1–1 (Cont.): Files Needed to Make a Bootable Volume

File	Description
PIP.SAV	File maintenance utility
DUP.SAV	Disk maintenance utility
DIR.SAV	Directory utility
RESORC.SAV	System resource display utility, handles mostly SHOW commands (optional)

The following example procedure creates a bootable RX50 diskette:

1. Initialize a diskette that has no file you want to save.

   ```
   INIT DU1:
   ```

2. Copy needed SYS files onto the diskette.

   ```
   COPY SY:(RT11FB,SWAP,DU,SL,LD).SYS DU1:
   ```

3. Copy needed SAV files onto the diskette.

   ```
   COPY SY:(PIP,DUP,DIR,RESORC).SAV DU1:
   ```

4. Copy the boot procedure to the boot blocks of the diskette.

   ```
   COPY/BOOT DU1:RT11FB.SYS DU1:
   ```

A Sample Control File

The following control file restores files backed up into savesets on tape. This file illustrates the complexity of control files. The list following the example explains some of this file’s contents. To create, use, and execute control files, see the *RT–11 IND Control Files Manual*.

```
.; ❶ RESALL.COM
.; Restores all files from all savesets on a tape.
.;
   ❷ .ASKS string Input device?
$ASSIGN ❸ 'string' ❹ BUD ❺
$LOAD 'string'
        .ASKS string Output device?
$ASSIGN 'string' OUT
        .SETS REWSWT "REWIND"          .; Start with a rewind
.100: ❻
$BACKUP/RESTORE/LOG/NOQUERY/'REWSWT' BUD:/SAVESET OUT:
        .SETS REWSWT "NOREWIND"
        .IF <EXSTAT> ❼ LT <ERROR>  .GOTO 100
;?RESALL-I-BACKUP exit status warrants termination of procedure. ❽
        .EXIT
```

❶ The period-semicolon combination (.;) specifies an internal comment. Anything on the line after them is comments for the programmer only and is not displayed when the COM file is run.

❷ The period-command combination (for example, `.ASKS`, `.SETS`, `.IF`, and `.EXIT`) specifies an IND directive or command.

❸ The dollar sign ($) followed by a DCL command specifies that command line is DCL. The dollar sign is optional, allowing an easy way for you to instantly identify a DCL command in a control file.

The IND processor forwards any line that does not begin with a period or a semicolon to KMON to interpret; and KMON disregards an initial dollar sign in a command line.

❹ The `'string'` specifies user input that the string will represent when the COM file is run.

❺ `BUD` is a logical device name meaning Back-Up Device.

❻ A label is an alphanumeric string preceded by a period and followed by a colon (.100:). Labels are local symbols specifying places in the control file so that the indicated lines can be referenced by an IND directive.

❼ `<EXSTAT>` and `<ERROR>` are IND special numeric symbols. This section of the example code means that if a saveset is successfully backed up and the tape does not rewind, then the next saveset is backed up.

❽ A line beginning with a semicolon specifies an external comment, which is displayed when control file is run.

Executing Command and Control Files

- To execute a command file, type $@command-file and press RETURN.

  ```
  .$@MYCOM.COM  RET
  ```

- To execute a control file, type IND control-file and press RETURN.

  ```
  .IND MYCTRL.COM  RET
  ```

- Depending on the KMON setting (see the SET KMON command), you can also use the at command (@) to execute either command files or control files.

  ```
  .@EXAMP.COM  RET
  ```

How DCL Commands Interact with CSI Commands

In addition to the DCL command language, RT–11 also has the CSI (Command String Interpreter) command language and the CCL (Concise Command Language) command language. The CCL language is a form of the CSI language; both are described in the *RT–11 System Utilities Manual* and both are not as easy to use as DCL.

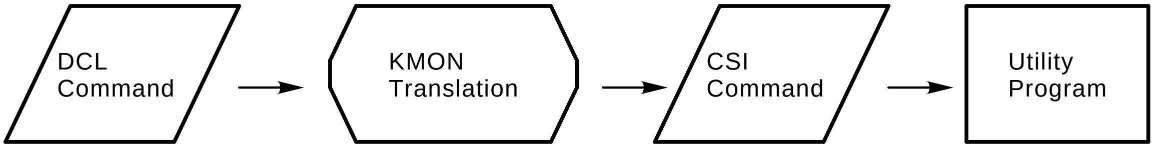
You can do many of the same things in all three languages, though if you are a system's programmer, you might want to use CSI or CCL for some additional functionality.

Whenever you issue a command, KMON inteprets that command, no matter what language it is in.

- If the command is a DCL one, KMON first checks to see if it is a command internal to KMON or external. If the command is an internal one, KMON itself executes the command; if the command is external (that is, requires action from a utility program or a device handler), KMON determines the correct utility program or device handler to execute that command, and then passes the command to that system component to execute.
- If the command is one you have defined, KMON translates it into standard DCL syntax and passes it to the UCL command processor for further interpretation.
- If the command is a CSI or CCL command, KMON passes that command to the utility you have named in the command; then the utility executes the operation.

Figure 1–1 illustrates the sequence of events when you issue a DCL complex command that involves a utility operation.

Figure 1–1: Sequence of Actions for DCL Commands Involving Utilities



See the *RT–11 System Utilities Manual* for a list of DCL and CSI equivalent commands and for any further information on how to use CSI commands.

Defining Your Own Commands

You can define your own commands or change the definition of existing commands to use in place of DCL commands.

The UCL (User Command Linkage) command-interpreter utility and the UCF (User Command First) command-interpreter utility both enable you to define your own commands. For more information, see Defining Your Own Commands in the *Introduction to RT–11*.

Chapter 2

Using Special-Function Keys

In addition to DCL commands, you can use special-function keys to control the operating system. These keys include:

- The top row of keys above the LK201 keyboard
- The middle set of keys on the LK201 keyboard
- The DELETE key ⌫ on the keyboard
- Control-key sequences on the keyboard

Useful Top-Row, Middle-Keypad, and DELETE Keys

See the manual that came with your terminal for instructions on how to use the top-row keys and the middle-keypad keys. The Hold Screen, the Print Screen, and the ESC keys on the top row are especially useful.

- To stop the display from scrolling, press Hold Screen once. To make the display continue scrolling, press Hold Screen a second time.
- To print information displayed on your terminal screen, press Print Screen. For the Professional series computer, Print Screen works only if the handler for your printer is loaded into the operating system and the transparent spooler is running. See the PRINT command description in the *Introduction to RT–11* for more information.
- If you are using the SL handler, you can move the cursor one character at a time left or right by respectively pressing ← or →. If you have SET SL KED,ON, you can also use the other arrow keys to recall commands previously executed.
- To delete the character to the left of the cursor, press the DELETE key ⌫.

Useful Control-Key Sequences

The following sections describe useful control-key sequences. To issue a control-key sequence, while pressing the CTRL key, press the other appropriate key in the sequence; for example, CTRL/C means while pressing CTRL, also press C.

Other than the CTRL/A key sequence listed in this section, the control keys you can use with the SL command-line editor are not listed here. See Table A–1 for the control-key sequences you can use with the command-line editor.

To Switch Between INSERT and REPLACE Modes (CTRL/A)

If you have SET the SL command-line editor ON, you can use CTRL/A to switch between INSERT and REPLACE modes.

As SL, the command-line editor, is distributed, any character you type within a command line is added; that functionality is called INSERT mode.

You can change that functionality while editing a command line so that any character you type replaces the character located at the cursor position; that is called REPLACE mode.

You alternate between INSERT and REPLACE modes by pressing CTRL/A. SL returns to INSERT mode each time you press RETURN to issue a command.

To Direct Keyboard Input to the Background Job (CTRL/B)

CTRL/B causes RT–11 to direct all keyboard input to the background job. The multi-job monitors echo B> on the terminal. The foreground or system job, however, has priority, so RT–11 returns control to the foreground or system job when it has output. In multiterminal systems, CTRL/B has no effect if the background console is not shared. CTRL/B directs all typed input to the background job until a CTRL/F redirects input to the foreground job or a CTRL/X directs input to a system job. CTRL/B has no effect when used under a single-job monitor or when a SET TT NOFB command is in effect, unless you are using SL.

To Terminate Program Execution (CTRL/C)

CTRL/C terminates program execution and returns control to the keyboard monitor. CTRL/C echoes ^C on the terminal.

You must type CTRL/C twice to terminate execution unless the program to be terminated is waiting for terminal input or is using the TT handler for input. In these cases, one CTRL/C terminates execution. Under the multi-job monitors, the job that is currently receiving input is the job that is stopped (determined by the most recently typed command, CTRL/F or CTRL/B). To make sure that the command is directed to the proper job, type CTRL/B, CTRL/F, or CTRL/X before typing CTRL/C.

To Direct Keyboard Input to the Foreground Job (CTRL/F)

CTRL/F causes RT–11 to direct all keyboard input to the foreground job and take all output from the foreground job. Multi-job monitors echo F> on the terminal unless output is already coming from the foreground job. If no foreground job exists, the monitor displays an error message (*F?*). Otherwise, control remains with the foreground job until redirected to the background job (with CTRL/B), or redirected to a system job (with CTRL/X), or until the foreground job terminates. In multiterminal systems, CTRL/F has no effect if the foreground console is not shared. CTRL/F has no effect when used under a single-job monitor, or when a SET TT NOFB command is in effect.

To Suppress Terminal Output (CTRL/O)

CTRL/O suppresses terminal output while continuing program execution. CTRL/O echoes as ^O on the terminal. RT–11 reenables terminal output when one of the following occurs:

- You type a second CTRL/O.
- You return control to the monitor by typing CTRL/C or by issuing the .EXIT request in your program.
- The running program issues a .RCTRLO or .MTRCTO programmed request (see the *RT–11 System Macro Library Manual*). RT–11 system programs reset CTRL/O to the echoing state each time you enter a new command string.

To Resume Terminal Display After Stopping It (CTRL/Q)

CTRL/Q resumes displaying characters on the terminal from the point the display previously stopped because of a CTRL/S. CTRL/Q echoes but has no effect under a multiterminal monitor if a SET TT NOPAGE command is in effect.

To Suspend Terminal Output (CTRL/S)

CTRL/S temporarily suspends output to the terminal until you type a CTRL/Q. CTRL/S does not echo. Under a multiterminal monitor, CTRL/S is not intercepted by the monitor if SET TT NOPAGE is in effect.

To Cancel/Delete the Current Command Line (CTRL/U)

CTRL/U cancels the current input line (all characters back to, but not including, the most recent line feed, CTRL/C, or CTRL/Z).

When SL is running, CTRL/U erases the current input line from the screen and from the command buffer. When SL is not running, CTRL/U echoes as ^U followed by a return and line feed at the terminal.

To Direct Keyboard Input to a System Job (CTRL/X)

CTRL/X causes RT–11 to prompt you for the name of a job to which it then directs all further keyboard input.

When you type CTRL/X, RT–11 displays the *Job?* prompt at the terminal. In response, specify the system-job name (or logical-job name) of the system job to which you want to direct input. Specify B or F to direct keyboard input to the background or foreground job, respectively.

If the specified job does not exist, RT–11 displays a question mark (*?*); otherwise, RT–11 displays the system-job name at the terminal. Control remains with the specified system job until the job terminates or control is redirected to the background job (with CTRL/B), the foreground job (with CTRL/F), or another system job (with CTRL/X), or a higher priority job gains control of the terminal. CTRL/X has no effect when used with a monitor that does not have system-job support or when a SET TT NOFB command is in effect.

To Terminate Terminal Input (CTRL/Z)

CTRL/Z terminates input when used with the terminal device handler (TT). It echoes as ^Z on the terminal. The CTRL/Z itself does not appear in the input buffer. Because CTRL/Z is a line terminator, you cannot delete it, once typed. If TT is not being used, CTRL/Z has no effect.

Chapter 3

Using Files, Volumes, and Devices

Most DCL commands work with files stored on volumes in devices. The following sections summarize information you should know when dealing with files, volumes, and devices. See the *Introduction to RT–11* for further information.

Using Storage Devices

Files are located on volumes in storage devices. When you issue a command affecting a previously created file, RT–11 has to first find that file on a volume. If you do not specify which volume, RT–11 presumes the specified file is on the default storage volume. Otherwise, you have to specify which volume your file is on.

You specify a volume by specifying the device the volume is in. You do that either by means of a standard one- or two-character physical-device name or by a logical-device name.

Specifying Physical-Device Names

If you do not specify a unit number (n) for devices with more than one unit, RT–11 assumes unit 0. If you have SYSGENed your monitor to have extended device-unit support, you can have more than eight device units.

RT–11 has *device* names not only for storage devices, printers, and terminals but also for *pseudodevices*. These are handler programs of different types that perform functions similar to real physical devices, such as enabling you to input and output data in various ways.

Table 3–1 lists the names by which RT–11 specifies real devices (storage devices, printers, terminals, and so on) and pseudodevices.

Table 3–1: Physical-Device Names

Name	Device Specified
BA:	BATCH device handler
DLn:	RL01, RL02 disk
DMn:	RK06, RK07 disk
DUn:	MSCP disk or diskette
DW:	RD type disk (Professional 300 series)
DXn:	RX01 diskette

Table 3–1 (Cont.): Physical-Device Names

Name	Device Specified
DYn:	RX02 diskette
DZn:	RX50 diskette (Professional 300 series)
EL:	Single-job monitor Error-Logger pseudodevice
LD:	Logical-disk subsetting handler pseudodevice
LP:	Line printer
LS:	Serial line printer
MMn:	TJU16/TJE16/TJU45/TJU77 magtape
MQ:	Message queue pseudodevice for interjob communication under multi-job monitors.
MSn:	TK25/TS11/TSV05/TSU05/TU80 magtape
MTn:	TM11/TMA11/TS03/TMB11/TU16/TE16 (TU1Q/TE1QW) magtape
MU:	TMSCP magtape handler
NC:	Ethernet handler (Professional 300 series)
NL:	Null pseudodevice
NQ:	Ethernet handler (Q-bus)
NU:	Ethernet handler (UNIBUS)
PI:	Professional 300 series interface handler
RKn:	RK05 disk cartridge drive
SPn:	Transparent spooler handler
TT:	Console terminal keyboard and display (hard-copy or video screen)
UB:	UNIBUS mapping handler
VM:	Virtual (Extended) memory handler
XC:	Communication port handler (Professional 300 series)
XL:	Communication port handler

Specifying Logical-Device Names

In addition to using the physical-device 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.

Specifying the RT–11 Default Logical-Device Names

RT–11 has two built-in logical-device names:

- **System (SY)**

  Specifies the system device and the volume containing the operating system files. RT–11 boots (starts up) from this volume when you turn on the computer. The system volume can be identified by its physical-device mnemonic and unit number or by the logical (symbolic) name SY. You cannot assign SY.

- **Default storage (DK)**

  Specifies the default storage volume, the volume that stores the files you create when you do not specify a device to store them. This volume can be identified by its physical-device mnemonic and unit number or by the logical (symbolic) name DK.

  When you first install the operating system, the system-device logical name (SY) and the default storage-device logical name (DK) specify the same device. However, at any time you can reassign the logical name DK to a different device.

Using Logical Disks

You can subset your physical disks into smaller logical disks to group together similar files in one place or if you have a large disk and cannot use all the space on that disk. The chapter Using the Logical Disk (LD) Utility in the *Introduction to RT–11* explains how to use logical disks.

Note: Do not confuse logical disks with logical-device names. They are similar in name, but are completely different. You can, however, assign logical-device names to specify logical disks.

Using a Virtual Disk

A virtual or VM (Virtual Memory) disk is an area in extended memory that you configure as a disk. The chapter on the virtual memory device in the *Introduction to RT–11* describes that device.

When you use a physical disk, your computer must periodically read and write code between the computer's memory and disk. But when you use a virtual disk, your computer reads and writes code between segments of memory.

Two advantages of using a virtual disk are:

- It lets you take advantage of memory space in your computer which you might not otherwise use.
- It can increase the speed of your computer, making your programs run more efficiently.

You should not use a virtual disk to store the only copy of files you need to keep, since the disk lasts only as long as the computer is turned on. However, you can copy files to a virtual disk and even boot your operating system from that disk, once you have first booted the operating system on a physical disk. See the *Introduction to RT–11* for that procedure. Then, you can use the virtual disk as a work-space disk.

Understanding Device Structures

RT–11 devices are categorized according to two characteristics:

- How devices process information
- How devices are physically structured

How Devices Process Information

All RT–11 storage devices are either randomly accessed or sequentially accessed.

- Random-access devices process blocks of data in random order; that is, independent of the data's physical location on the device or its location relative to any other information. All disks, diskettes, and VM devices fall into this category.

  Random-access devices are sometimes called block-replaceable devices, because you can manipulate (rewrite) individual data blocks without affecting other data blocks on the device.

- Sequential-access devices require sequential processing of data; the order in which the system processes the data must be the same as the physical order of the data. RT–11 sequential devices are magtape and printer.

How Devices Are Physically Structured

Devices are structurally categorized according to whether or not they can store files:

- **File-Structured Devices**

  File-structured devices store data under assigned file names. RT–11 devices that are file structured include all disk and diskette devices.

  File-structured devices that have a standard RT–11 directory at the beginning are called RT–11 directory-structured devices. A device directory consists of a series of directory segments that contain the names, lengths, and dates of the files on that device. The system updates the directory each time a program moves, changes, adds, or deletes a file on the device. (The *RT–11 Volume and File Formats Manual* contains a more detailed explanation of a device directory.) RT–11 directory-structured devices include all disks.

- **Nonfile-Structured Devices**

  Nonfile-structured devices, however, do not store files; they contain a single logical collection of data. These devices, which include the printer and terminal, are generally used for reading and listing information.

Special Directory Devices

Some devices that do not have the standard RT–11 directory structure, such as magtape, store directory information at the beginning of each file, but RT–11 must read the device sequentially to obtain all information about all files.

Table 3–2 shows the relationships among devices, access methods, and structures.

Table 3–2: Device Structures

	Structure			
Device	**File**	**Nonfile**	**RT–11 Directory**	**Non-RT–11 Directory**
Random Access				
Disk, diskette	x		x	
VM	x		x	
Sequential Access				
Magtape	x			x
Printer		x		
Terminal		x		

Specifying Files

You specify files by using 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; and 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. You can modify the default file types for some utilities, using the software customization described in the *RT–11 Installation Guide*.

Table 3–3 lists the standard file types used in RT–11.

Table 3–3: Standard File Types

Type	Meaning
ANS	SYSGEN answer file
BAC	Compiled BASIC–PLUS 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. A BAD file cannot be moved by SQUEEZE.
BAK	Editor backup file
BAS	BASIC–PLUS source file (BASIC–PLUS input)
BAT	BATCH command file
BLD	Command file to execute SYSGEN monitor (MON) and device handler (DEV) build files
BOT	A magtape boot file
BUP	Backup utility program output file
C	PDP–11 C source file (C input)
CMD	PDP–11 C command file
CND	SYSGEN conditional file
COM	DCL command file, IND control file, or SIPP command file
CRF	Cross-reference listing file
CTL	BATCH control file generated by BATCH compiler
CTT	BATCH internal temporary file
DAT	BASIC, FORTRAN, or IND data file
DBL	DIBOL source file

Table 3–3 (Cont.): Standard File Types

Type	Meaning
DDF	DIBOL data file
DEV	SYSGEN device handler build file
DIF	BINCOM or SRCCOM differences file
DIR	Directory listing file
DMP	DUMP output file
DSK	Logical disk file (for use with LD handler)
FNT	Font table file
FOR	FORTRAN IV or FORTRAN–77 source file (FORTRAN input)
H	PDP–11 C header file (C input)
IDX	Index file for the INDEX program
IMG	Data file for the INDEX program
JBK	Journal backup file
JOB	Queue file where a job is stored
JOU	Journal file
KED	KED macro file
LDA	Absolute binary (load image) file (optional linker output)
LOG	BATCH log file
LST	Listing file (MACRO, FORTRAN, PDP–11 C, LIBR, or DIBOL output)
MAC	MACRO source file (LIBR, MACRO, or SRCCOM input; PDP–11 C or LIBR output)
MAP	Map file (linker output)
MLB	MACRO library file
MON	SYSGEN monitor build file
MSG	PDP–11 C binary message file
OBJ	Relocatable object module (MACRO, FORTRAN, DIBOL, or PDP–11 C output; LIBR or LINK input)
OBJ	Relocatable object library (LIBR output; LIBR or LINK input)
REL	Foreground-job relocatable image (linker output, default for monitor FRUN and SRUN commands)
SAV	Executable program file
SLP	SLP command file
SML	System MACRO library

Table 3–3 (Cont.): Standard File Types

Type	Meaning
SOU	Temporary source file generated by BATCH
STB	Symbol table file in object format, containing the symbols produced during link
SYG	Monitor and handler files resulting from system generation
SYS	Monitor files and handlers
TBL	Monitor device table section created during SYSGEN
TMP	Temporary workfile (BASIC-PLUS, ERROUT, PDP–11 C)
TMx	Temporary SYSGEN files where x is a number from 1 through 5
TXT	Text file
WRK	Temporary work file

Allocating Space for Output Files

When you specify an RT–11 command to open an output file (for example, when you issue the COPY or the CREATE command), but do not specify the size of the output file, RT–11 tentatively allocates space for that file. RT–11 does this by allocating one-half of the largest space or the entire second largest space, whichever is bigger, on the specified storage device. If the output operation requires more space than that allocated by default, the operation fails. To allocate more space, you must specify with the /ALLOCATE option the amount of space you want the file to have, if there is sufficient space on the output device.

How Data Is Stored in Files

RT–11 stores data in several different binary codes. These codes are of two major types:

- Text-file code
- Nontext-file code

Text-File Code

A file that you can read is called a *text* file, and RT–11 stores the data in that file in ASCII (American Standard Code for Information Interchange) code. Each character in this code is represented by seven bits in a byte. The appendix of the *PDP–11 MACRO–11 Language Reference Manual* for the DEC multinational character set includes the ASCII character codes. Use the DUMP command to examine a file's binary content.

Nontext-File Code

Some files do not contain ASCII characters. These files include the output of compilers, assemblers, and linkers; and their binary code (that is, the format in which data is stored) varies.

RT–11 deals with the following nontext-file file types:

- **MACRO–11 Assembler Produced Files**
  - **OBJ Files**

    An object (OBJ) file consists of data and PDP–11 machine language code. It is used as input to the linker.
- **Linker Produced Files**
  - **SAV Files**

    A memory-image (SAV) file is an executable program file. This file does not require relocation and can run in a single-job environment, as a background program under the multi-job monitor including a mapped monitor, as a foreground virtual job under a mapped monitor, or as a completely virtual job under a mapped monitor.
  - **REL Files**

    A relocatable-image (REL) file is one linked as though its bottom address were 1000 and with relocation information included in its memory image. When you call such a program file with the FRUN or SRUN command, the file is relocated as it is loaded into memory. A relocatable-image file can run in a foreground environment.

- **LDA Files**

  Load-image (LDA) files are ones that you can produce, load, and execute in stand-alone environments without relocating them.

- **SYS Files**

  Operating system files and handler files (SYS) are those that run the operating system and let you process input and output on devices.

For more information about these file types, see the *RT–11 Volume and File Formats Manual*.

Part II
RT–11 Command Descriptions

Part II is an alphabetical listing of RT–11 keyboard monitor commands, most of which are DCL commands.

Each command description contains the following elements:

Type of Command

In parentheses underneath the command is the name of the RT–11 system component that executes that command. This is either KMON alone, a utility program(s), or a device handler.

Definition

A few sentences identifying the command.

Format

The syntax for using the command.

Option Summary

A listing of all the command’s options, if it takes options.

Parameter Summary

A brief description of the command’s parameters.

Description

An explanation of how to use the command.

Option Descriptions

An explanation of how to use each option.

Examples

Illustrations of how to use the command.

ABORT

(KMON)

Lets you abort from the shared console any foreground or system job not protected by the .SCCA programmed request.

Format

ABORT *jobname*

Parameter

jobname
The foreground or system job

Description

When you type the ABORT command at the system console, the foreground or system job is immediately aborted.

When running under a multi-job monitor that does not include system-job support, use F as the jobname to abort the foreground job. If your monitor includes system-job support, type the name of the job you want to abort.

The ABORT command is not supported by single-job monitors and cannot abort a job with SCCA in effect.

ASSIGN

(KMON)

Associates the logical name you specify with a physical device or a previously defined logical-device name.

Format

ASSIGN *device-name: logical-device-name:*

Parameters

device-name

Specifies either a physical-device or a logical-device name.

- **Physical-device name**

  The standard RT–11 permanent name referring to a particular device installed on your system. Table 3–1 lists these names. (The colon that follows the device name is optional.)

- **Logical-device name**

  A logical-device name previously defined (one that is currently assigned to a physical device).

logical-device-name

Specifies either a logical-device name or the catch-all device assignment.

- **Logical-device name**

  An alphanumeric name, from one to three characters long and followed by an optional colon that you assign to a particular device.

- **Catch-all device**

  If your monitor has extended device-unit support, in place of the second ASSIGN parameter, you can specify an asterisk (*) to assign *device-name* as the catch-all device. This device becomes the destination for all device references to undefined device names. See the Extended-Unit System-Generation Option section in the following command description for further information on the catch-all device.

Description

Assigning a logical name to a physical device simplifies programming by allowing you to write device-independent programs. When you write a program, for example, you can request input from a device called IN and direct output to a device called OUT. Each time you execute the program, you can assign those logical names to the physical devices you need to use for that job.

The ASSIGN command is helpful when you have a program that refers to an unavailable device. In this case, you can use the ASSIGN command to direct input and output to an available device.

If the logical name you supply is already associated with a physical device, RT–11 disassociates the logical name from that device and assigns it to the device you specify.

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.

Displaying Logical-Device Assignments

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

Two System Logical Names

RT–11 uses two special logical-device names: SY and DK. SY is the operating system device; that is, the device from which the monitor was bootstrapped. You cannot assign SY to represent any other device. DK is the default storage device. You can assign DK to any device.

When you boot the operating system, SY and DK are both assigned to the boot device. However, after you install the system files, you can reassign the logical name DK to a different device.

NOTE

> Logical-name assignments stay in effect until the system is rebooted. So, if you want your logical-name assignments to be used each time you start your operating system, you should include these assignments in your startup command file.
>
> You cannot assign BA and SY.

Extended-Unit System-Generation Option

If you do a system generation and select monitor support for extended-device units and if you boot the monitor you generated, you can use the ASSIGN command with the asterisk (*) as in the following format:

ASSIGN catch-all-device *

This command causes any operation that is directed to an undefined device to be directed to the *catch-all-device* without confirmation. That is, if a specified device does not exist as a logical or physical device, then operations directed to that undefined specified device are automatically redirected to the device you assigned as the catch-all device.

The catch-all device gets a device reference if the device is not found in the defined logical-device name table or is not a reference to a physical device in the system.

Physical device references are in the form of *ddn* or *dd* or, in the case of the extended-device units, *d* or *dnn*.

Use of the Catch-All Device

The device you assign as the catch-all device should be an initialized device that contains no data you want to preserve, such as a logical disk.

The catch-all device is useful in preventing command-file failure when:

- An unmounted output device has been associated with a logical name, and the logical name is required by the command file. For example, if you reference LD3 and LD3 is not mounted, it will not go to the catch-all device.
- A device has not been assigned the proper logical name.

Catch-All Restrictions

- You must have generated support for extended device-units, to be able to use the catch-all device.
- If you make a reference to a *d* or a *dd unit* that does not exist when *d* or *dd* is a valid physical device, you will get an error, even if you have assigned the catch-all device. A reference to a physical-device name with an invalid unit always gives you an error. For example, if you have a DU device and you reference DU3 and there is no DU3, you will get an error.

Warning

Digital recommends you use as a catch-all device, one whose only purpose is to be the catch-all device. You should not specify the DK or SY as the catch-all device. A mistyped command could unexpectedly alter or delete data residing on DK or SY.

Examples

1. RT–11 uses the LS handler for serial-interface printers and the LP handler for parallel-interface printers. However, by default RT–11 utilities send print jobs to a device named LP. So, if your printer is a serial-interface printer, you must logically associate the name LP with the LS handler by issuing the following ASSIGN command before you use the PRINT command for the first time. Otherwise, the output to LP will be sent through the parallel-interface handler:

   ```
   .ASSIGN LS: LP:
   ```

   The colons following the device names are optional.

2. The next command redefines the default storage device.

   ```
   .ASSIGN DU1: DK:
   ```

   If after executing this command you supply a file specification in a command and omit the device name, it now defaults to DU1. This does not affect the default system device, SY.

3. The following example is typical for a system that uses a dual-drive diskette device. Several users can share the same system software on DY0 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 DY1.

   ```
   .ASSIGN DY1: DK:
   ```

B (Base)

(KMON)

Sets a relocation base used to calculate the address of the location to be referenced in a subsequent Examine or Deposit command. RT–11 adds this relocation base to the address you specify.

Format

B *[address]*

Parameter

[address]

Specifies an octal address that RT–11 uses as a base address for subsequent Examine and Deposit commands. If the address you supply is an odd number, RT–11 decreases it by one to make the address even. If you do not specify an address, this command sets the base to zero.

Description

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 B command has no effect on program execution.) RT–11 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.

Examples

1. The following command sets the base to 0:

   ```
   .B
   ```

2. The next two commands both set the base to 1000:

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

BACKUP

(BUP)

- Backs up and restores RT–11 files or volumes in saveset or logical-disk files.
- Initializes backup volumes when needed.
- Verifies backed-up and restored data.
- Lists directories of backup volumes and logical disks.

See the *Introduction to RT–11* for a tutorial explanation of RT–11's backup facilities.

Format

BACKUP *in-spec out-spec*

Command Options	**In-Spec Options**
/DEVICE	{ [ssname]/SAVESET[,filename,...]
/DIRECTORY [{ /OUTPUT:filespec /PRINTER }]	[ldname]/SUBSET[,filename,...] }
/FILE	
/INITIALIZE	
/[NO]**LOG**	
/[NO]**QUERY**	
/RESTORE	
/[NO]**REWIND**	
/[NO]**SCAN**	
/SYSTEM	
/VERIFY	

Parameters

in-spec

You can type up to six input file specifications. If you specify a saveset or logical-disk subset, then it must be the first input specification and cannot contain wildcards. Otherwise, you can use wildcards.

You can specify only one input device; that is, all the input files must be on the same device. DK is the default input device.

The *ssname* file name means saveset name, and the *ldname* file name means logical-disk name. Both of these names can be from one to six alphanumeric characters long with a file type of up three alphanumeric characters.

Saveset

A saveset is a special file created by the BUP utility to hold one or more files or an RT–11 volume or device image you specified with the BACKUP command

and the /SAVESET option (the default). Unless you specify the /SUBSET option, BACKUP presumes it is working with a saveset.

Each saveset is the result of a single backup operation, is stored in the format of an RT–11 volume, and requires the BACKUP/DIRECTORY command to read its contents.

Subset

A subset is a logical disk created by the BUP utility to hold one or more files you specified with the BACKUP command and the /SUBSET option. Since more than one file can be in a logical disk, it can be considered to contain a subset of your files on a physical disk.

Subsets can be used like any other logical disks; however, unlike the logical disks you create with the CREATE command, subsets have no free blocks in them.

out-spec

Specifies the device in which you will mount the output volume(s) for the backup operation and the saveset or logical-disk file name for the backup. The output volume can be any type of media.

Out-Spec Defaults

- *BACKUP*—default output saveset or logical-disk name for a file backup.
- *ddn* (name of the device being backed up)—default output saveset name for a device-image backup.
- *BUP*—default output saveset type for all saveset backups (file and device-image)
- *DSK*—default output file type for all logical-disk subset backups.
- *BUPDIR.DIR*—default output file name and type for all BACKUP output files containing directories of savesets.
- *BACKUPDIR.DIR*—default output file name and type for all BACKUP output files containing directories of logical disks.

Input and Output Volumes

You can use random-access volumes and logical disks as either input or output volumes for both backup and restore operations.

Magtapes, however, can be used only as output volumes for a backup operation, and only as input volumes for a restore operation.

You can transport BUP-written magtapes to a VMS system and extract files from those magtapes. After copying the savesets to a VMS directory, use the VMS EXCHANGE utility to manipulate the files. The *Introduction to RT–11* describes this procedure.

Note:

- You must copy to a saveset even when you are backing up a volume.

  A saveset can be thought of as a container that holds one or more files or an RT–11 volume or device image from a single backup operation.
- You can use wildcards on input files but not on saveset specifications.
- You can type only one output specification for backing up your data.
- Output volumes for backing up data as savesets must be initialized by BUP. See the /INITIALIZE option for information on initializing backup volumes.

Bad Blocks on the Input Volume

By default, BUP successfully tolerates up to 25 bad blocks on the input device when backing up a disk to a backup volume or when restoring from a backup volume to a disk (copy back).

BUP issues a warning message each time it encounters a bad block on the input device, then continues to back up or restore. If BUP encounters more than 25 bad blocks on the input device, BUP issues a fatal error message and the operation is stopped.

You can change the number of bad blocks BUP accepts on the input volume by using the customization procedure described in the *RT–11 System Release Notes*.

Description

The BACKUP command has the following primary features:

- Backs up files or RT–11 volumes of any size to any media.

  This means you can back up the data from a large volume to as many smaller volumes as you need.
- Creates more than one saveset on a backup volume, if the volume is large enough. Since individual files are enclosed in a saveset, you do not have to worry about multiple copies of the save file (from different backups) overwriting each other.

  This means you can back up data from several small volumes to one large volume.
- Creates savesets on magtape that are easily transportable to VAX processors running the VMS operating system. Once the files on saveset are transported to a VAX, those files are easily read and manipulated.
- Backs up files to automatically created logical-disk files called subsets.
- Lists directories of savesets, of files in savesets, and of files in logical disks.
- Restores data from savesets and from logical-disk subset files.

Four Distinct BACKUP Command Operations

You can use the BACKUP command to do the following four operations:

- Back up data into a saveset or subset file.
- Verify the accuracy of your backed-up data.
- Get a directory of backed-up data.
- Restore backed-up data.

Each of these operations can be done separately, though the verification operation can also be done with a backup or restore operation. Table 1 lists all the valid BACKUP command option combinations for these backup operations.

Table 1: Valid BACKUP Option Combinations

Operation	Options
BACKUP to a saveset	/[NO]SCAN /DEVICE /[NO]REWIND /VERIFY /[NO]LOG /[NO]QUERY /INITIALIZE
BACKUP to a subset	/SUBSET /VERIFY
DIRECTORY	/DIRECTORY /DIRECTORY/OUTPUT[:filespec] /DIRECTORY/PRINTER /SUBSET /SAVESET
RESTORE from a saveset	/RESTORE /SYSTEM /FILE /DEVICE /[NO]REWIND /SAVESET /VERIFY /[NO]LOG
RESTORE from a subset	/SUBSET /RESTORE /SYSTEM /VERIFY /[NO]LOG
VERIFY (only)	/RESTORE/VERIFY:ONLY

Backing Up Files

If you supply no options, BUP assumes you want to back up a file(s) into a saveset. This procedure allows you to back up all files on a device without copying empty blocks.

You can use wildcards to back up all files of a particular name or type, or to back up all files (*.*), which is the default.

Examples

1. The following command illustrates the use of wildcards when backing up files:

```
.BACKUP DL0:F*.FOR,*.MAC,T*.SAV DU1:WRK
Mount output volume in DU1:; Continue?  Y
?BUP-W-Not a BACKUP volume DU1:
DU1:/BUP Initialize; Are you sure?  Y
?BUP-I-Bad block scan started
?BUP-I-No bad blocks detected
?BUP-I-Creating output volume 1
 Files backed up:
DL0:FUNC1.FOR
DL0:FUNC2.FOR
DL0:PROG1.MAC
DL0:PROG2.MAC
DL0:DEMO.MAC
DL0:KMON.MAC
DL0:TEST.SAV
?BUP-I-Backup operation is complete
.
```

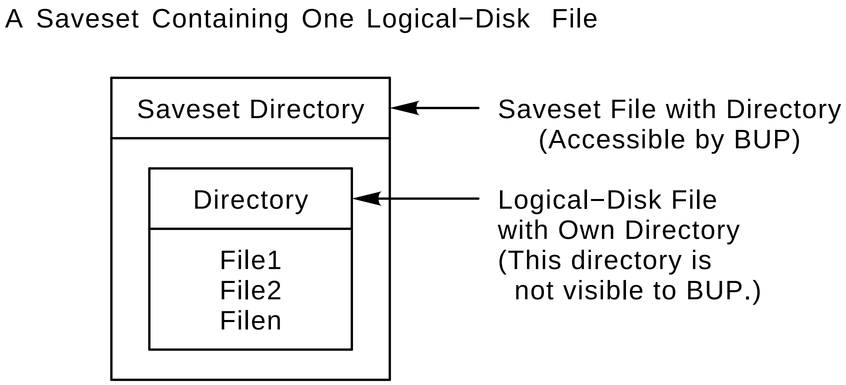
2. The next command illustrates backing up a logical-disk file into a saveset. This command stores the logical disk SAVE.DSK in the saveset OLDWRK.BUP on device DU1:

```
.BACKUP DL0:SAVE.DSK DU1:OLDWRK
```

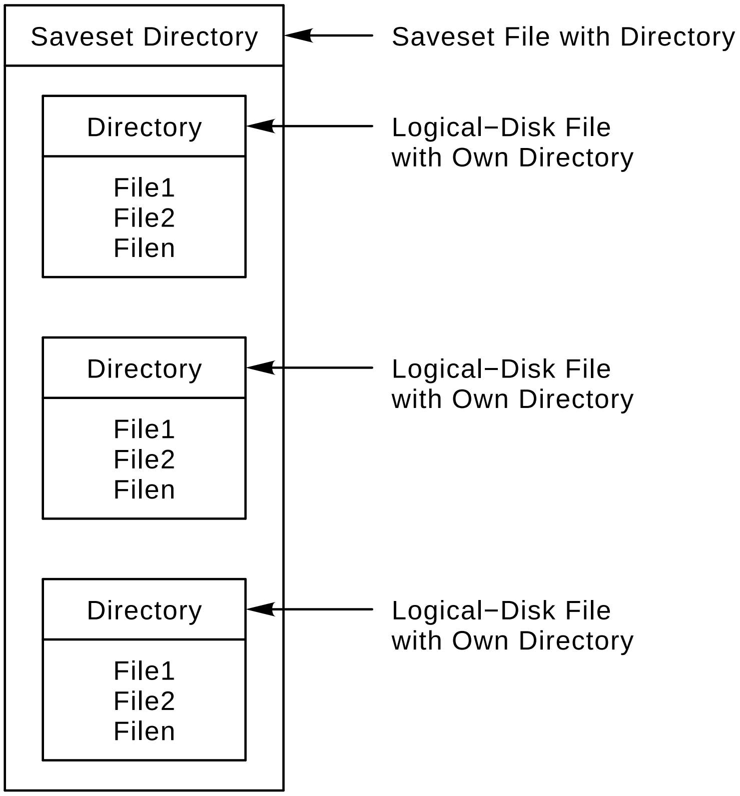
An advantage of backing up logical disks as individual files is that it can be convenient to save several logical disks in one saveset. A disadvantage is that you cannot access (get a directory listing or restore) individual files in each logical disk stored this way.

Figure 1 illustrates this way of backing up logical disks.

Figure 1: Savesets Containing Logical Disks Backed Up As Files


A Saveset Containing One Logical−Disk File
Saveset Directory
Saveset File with Directory (Accessible by BUP)
Directory
File1
File2
Filen
Logical−Disk File with Own Directory (This directory is not visible to BUP.)

A Saveset Containing Several Logical−Disk Files



Options

/DEVICE

Backs up or restores (with the /RESTORE option) an entire volume in image mode.

The backup operation backs up everything on a volume, including empty blocks and directory blocks. You can back up volumes to disks, diskettes, or magtapes.

BUP copies the input volume to one or more output volumes. If there is more data than will fit on the output volume, BUP verifies the output data it has already copied (if you have specified /VERIFY) and then prompts you to remove the backup volume and insert another. BUP repeats this process until the entire input volume has been copied.

Examples

1. The following command backs up a DU0 volume to several diskettes, using DU1. The backup volumes will contain the saveset file DU0.BUP when the backup operation is complete:

```
.BACKUP/DEVICE DU0: DU1:
```

If you specify no output file name, BUP uses the three-character mnemonic of the input device (for example, DU1). The default output file type is BUP.

2. The next command restores the device-image saveset DL1.BUP on device MS0 to device DL1:

   ```
   .BACKUP/RESTORE/DEVICE/VERIFY MS0:DL1 DL1:
   ```

3. Backing up logical disks as device images is the same as backing up any volume as a device image.

   To back up a logical-disk file as a device, you must first associate it with a logical-disk unit (see the MOUNT command). The following command backs up the logical disk mounted in logical-disk unit 0:

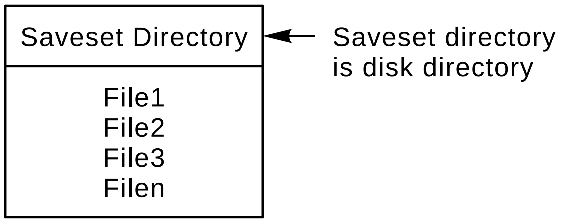
   ```
   .BACKUP/DEVICE  LD0: DU1:
   ```

   The default name of a logical-disk saveset is the name of the device from which you make the backup. In this case, the logical disk is backed up onto device DU1 with the default saveset name of LD0.BUP.

   The advantage of doing this is that you can access the files in the logical disk while it is in the saveset. That is, you can still get a directory listing of the files in the logical disk and restore them as individual files. The disadvantage is that you can store only one logical disk in a saveset to be able to access the files in the saveset.

   Figure 2 illustrates this way of backing up logical disks. Note that in this case, the logical-disk directory becomes the saveset directory; that is, they become one and the same directory.

Figure 2: Saveset Containing a Disk Backed Up As a Device Image



/DIRECTORY

Does the following three types of directory operations:

- Lists savesets on a backup volume or on a series of backup volumes.
- Lists files in a saveset.
- Lists files within a logical-disk file.

The default of the /DIRECTORY option is to display a directory listing on the terminal. See the /PRINTER and the /OUTPUT options for descriptions of how to print a directory listing or put it in a file.

1. **Listing a directory of savesets on a backup volume**

   You use the same command format to get backup volume directories on magtapes as on disks or diskettes. However, for certain magtape devices, this process can take some time. This is because BUP must read to the logical end of magtape volumes before completing a directory listing.

   The directory structure of savesets on backup disks, diskettes, and magtapes is different from the standard RT–11 directory structure. In addition, a saveset directory on a magtape is different from that of one on a disk or diskette.

   To display a backup volume directory, first mount the backup volume that contains either the entire saveset or the first section of a multivolume saveset. Then issue the BACKUP/DIRECTORY command.

   Examples

   a. In this example, the directory listing is of the backup volume in device DU1. The backup volume is a diskette whose backup directory shows that the volume is the first one in a series of five diskette backups that contains three savesets:

```
.BACKUP/DIRECTORY DU1:

 RT-11 BACKUP
 05-May-91 09:28
 Volume 1

 Saveset ❶    Section ❷  Blocks ❸      Date ❹

OBJ   .BUP    1/1         474/474      22-Apr-91
TEMP  .BUP    1/1         14/14        22-Apr-91
RUNOFF.BUP    1/5         304/3114     13-Mar-91

 3 Saveset sections, 792 Blocks
 0 Free blocks
```

❶ The *saveset* file name identifies a saveset section.

If more than one saveset is on a volume, each one on that volume is listed. In this case, there are three savesets.

❷ A *section* of a saveset is the amount of the saveset that fits on one backup volume. This means that the number of sections in a saveset is the same as the number of volumes used to back up a disk.

The *section number* in a saveset directory indicates what section of a saveset is on that volume. Disk and diskette directories have a second number following the section number and separated from it by a slash. This second number is not on magtape saveset directories.

The second number indicates how many sections a saveset file is divided into. For example, a 1/1 for saveset number information means the saveset is undivided and the entire saveset file is on that volume. However, a 1/2 means the saveset file is divided into two sections (since it did not fit on the volume), and the first section is contained on that backup volume.

In this case, the last saveset on the diskette, RUNOFF.BUP, is spread across five volumes.

❸ The first number under *blocks* lists the size in blocks of a saveset section on the backup volume. The second number lists the total size of the saveset. If the two numbers are the same, the entire saveset is on that volume.

❹ The *date* on which a saveset is backed up to a backup volume is listed next. If no system date is set, no creation date is entered in the directory.

b. The following example is of a backup magtape directory. The magtape backup volume contains the second section (2348 blocks) of a 5400-block saveset named BIGDSK.BUP, the complete savesets FIRST.TXT and SECOND.BUP, and the first (409-block) section of a 988-block saveset THIRD.BUP.

```
.BACKUP/DIRECTORY MU0:

 RT-11 BACKUP
 05-May-91 16:40

 Saveset      Section     Blocks      Date

BIGDSK.BUP    2           2348/5400   20-Mar-91
FIRST .TXT    1           800/800     20-Mar-91
SECOND.BUP    1           5400/5400   21-Mar-91
THIRD .BUP    1           409/988     26-Mar-91

 4 Saveset sections, 8957 Blocks
```

You would need to mount a previous magtape in this series to restore the saveset BIGDSK.BUP, because the first section of that saveset is not located on this volume. A printed directory of savesets for each magtape backup volume would direct you to the correct volume to mount. You can restore the savesets FIRST.TXT and SECOND.BUP from this volume. Proceed to the next magtape volume of this series to restore the second section of THIRD.BUP.

2. **Listing a directory of files in a saveset**

To get a directory of a saveset, you must specify the [ssname]/SAVESET option with the /DIRECTORY option. The saveset name specifies the saveset for which you want a file directory.

If you do not specify a saveset name, you have two possibilities:

- On a random-access device, BUP looks for the saveset BACKUP.BUP.
- On a tape, BUP displays a directory of the first saveset on the tape.

The following example command displays a directory of the files backed up in saveset TEMP.BUP, residing on DU1:

```
.BACKUP/DIRECTORY DU1:TEMP.BUP/SAVESET

 RT-11 BACKUP
 07-May-91 10:54
 Saveset: DU1:TEMP.BUP
 Created: Monday   06-May-91 09:23

 File         Blocks  Volume          Date

TEMP  .TMP    2       1        Monday   06-May-91
CACHE .TMP    3       1        Monday   06-May-91

 2 Files, 5 Blocks
```

3. **Listing the files in a logical disk**

Logical disks backed up with the /SUBSET option have the same format as regular logical disks. This means you can use the BACKUP/DIRECTORY or the DIRECTORY command to display logical-disk file directories, whether or not BUP created the disks.

The BACKUP/DIRECTORY command enables you to list logical-disk directories without having to mount them separately. To get a directory of a logical disk, you specify the [ldname]/SUBSET option with the /DIRECTORY option.

The following example command displays on your terminal a directory of the logical disk MYBACK.DSK on DU0:

```
.BACKUP/DIRECTORY DU0:MYBACK.DSK/SUBSET
```

/FILE

Used only with the /RESTORE option. The /RESTORE/FILE option combination restores an entire saveset to one file. Two reasons for using this option are:

- For restoring file-image backups not contained within a saveset

  Versions of RT–11 before Version 5.5 let you create a file-image backup that was not contained within a saveset. Such a file image has a format different from that of a saveset. You restore such a file image from a backup volume or series of backup volumes by including the /FILE option together with the /RESTORE option. Because you are performing a file restoration to a disk, BUP does not initialize that disk as part of the operation.

- For restoring a saveset as one logical-disk file

  This is helpful if your saveset file is a logical disk and you want to restore it as one file.

Using the /FILE option with both the /RESTORE and /DEVICE options (/RESTORE/DEVICE/FILE) is equivalent to using the /RESTORE option by itself. This operation restores one or more files from a backup volume saveset.

The following example command restores, with verification, the file image FIRST.TXT from magtape MS0 to device DL1:

```
.BACKUP/FILE/RESTORE/VERIFY MS0:FIRST.TXT DL1:
```

/INITIALIZE

Initializes a volume for use as an output volume in a backup operation.

The BACKUP command backs up data only to BUP-initialized volumes. So, you should use this option if your backup volume is an uninitialized backup volume or if you want to reinitialize a previously used backup volume.

Since initialization overwrites any entries in a volume's directory, BUP examines an uninitialized disk or diskette backup volume and prompts you for confirmation before initializing it:

- If the backup volume has a standard RT–11 format (not formatted for BUP backup savesets), the volume may contain files you wish to keep, and BUP prompts you with the message:

  ```
  ?BUP-W-Not a BACKUP volume <dev>
  <dev>:/BUP Initialize; Are you sure?
  ```

  If you type Y RETURN, BUP proceeds to initialize the volume indicated by the <dev>. If you type N RETURN, BUP prompts you:

  ```
  Mount output volume in device-name; Continue?
  ```

  If you mount a new output volume and type Y RETURN, BUP continues with the backup operation. If you type N RETURN, BUP returns you to the period prompt.

- If the output volume is not a valid RT–11 volume, BUP displays a message indicating that and allows you either to initialize it as a backup volume or to replace it; for example:

  ```
  ?BUP-W-Volume not RT-11 initialized
  DU1:/BUP Initialize; Are you sure?
  ```

You can use the /INITIALIZE option with /NOQUERY to suppress the confirmation messages.

When BUP creates a logical-disk backup file, BUP automatically initializes the logical disk before it backs up data into it.

The Initialization Procedure

- For random-access volumes, the initialization procedure:
  - — Clears the directory of the volume and writes information into the home block (block 1) so BUP can recognize the volume as a backup volume.

  - — Scans the output volume for bad blocks, since backup volumes must not contain bad blocks. If BUP finds no bad blocks, it displays the message:

    ```
    ?BUP-I-No bad blocks detected
    ```

    But if BUP finds a bad block on an output volume, BUP issues a fatal error message and stops the backup operation:

    ```
    ?BUP-F-Bad blocks detected; use another volume
    ```

    In this case, you must mount and initialize another volume.

    If you are sure all the backup media contain no bad blocks, you can include the /NOSCAN option in your BACKUP command to prevent a bad-block scan of each backup media.

- For magtapes, the initialization procedure rewinds the magtape and writes a volume and header label at the beginning of the magtape.
- For all backup volumes, BUP automatically prompts you for initializing each subsequent backup volume, when you need more than one.

Once you have an initialized backup volume, you should not reinitialize it to do further backup operations on that volume unless you want to delete backup information already on the volume.

To return a BACKUP-initialized volume to an RT–11 structured volume for use other than with the BACKUP command, initialize the volume, using the INITIALIZE command.

The following command example shows the initialization of the backup volume on DU1. The backup saveset WRK.BUP is created on that volume:

```
.BACKUP/INITIALIZE DL0:F*.FOR DU1:WRK
Mount output volume in DU1:; Continue?  Y
?BUP-W-Not a BACKUP volume DU1:
DU1:/BUP Initialize; Are you sure?  Y
?BUP-I-Bad block scan started
?BUP-I-No bad blocks detected
?BUP-I-Creating output volume 1
 Files backup up:
DL0:FUNC1.FOR
DL0:FUNC2.FOR
?BUP-I-Backup operation is complete
.
```

/[NO]LOG

Displays a list of all the files affected by a BACKUP command operation as the operation progresses, unless you specify /NOLOG. The /LOG option is the default.

/NOLOG suppresses the various informational messages BUP displays as backup operations are performed; for example, /NOLOG suppresses the *files processed* message (the listing of the files processed).

You might use this option with /NOQUERY, if you are using BUP from a KMON command and IND control file to write a single disk or magtape output volume.

/OUTPUT:filespec

Used only with the /DIRECTORY option. The /OUTPUT:filespec option with the /DIRECTORY option stores a backup or logical-disk volume directory in a file.

This command lists the directory of the saveset file MYBACK.BUP in the file MYBACK.DIR:

```
.BACKUP/DIRECTORY/OUTPUT:DU0:MYBACK.DIR  DU1:MYBACK.BUP/SAVESET
```

You must specify a device name with the /OUTPUT option if you want to place the directory listing in a file. If you do not also specify a file name for a saveset directory, the default directory-output file name is BUPDIR.DIR. If you do not also specify a file name for a logical-disk directory, the default directory-output file name is BACKUPDIR.DIR.

/PRINTER

Used only with the /DIRECTORY option. The /PRINTER option with the /DIRECTORY option prints directory listings of backup or logical-disk volumes.

The following command sends a backup volume directory to a printer:

```
.BACKUP/DIRECTORY/PRINTER MU0:
```

/[NO]QUERY

Prompts you for the various responses required from the terminal. This is the default. The /NOQUERY option inhibits prompting. You may want to do this in a KMON command or IND control file to write a single disk or magtape output volume. However, if the backup volume is not large enough to hold the data, the mount prompt(s) for the subsequent volumes requires terminal response, even if you use the /NOQUERY option.

/RESTORE

Restores backed-up data to a standard RT–11 formatted disk. You can restore:

- Complete savesets as a file or as a device
- Selected files from a saveset
- Selected files from a logical-disk file
- File-image backups not contained in a saveset or subset (data backed up as a file by RT–11 before Version 5.5)—see the /FILE option for a description of this operation

You can use wildcards to restore files of only a particular name or type.

Defaults

If you do not specify other options with /RESTORE, the BACKUP command uses the /SAVESET option and restores all the files in the saveset BACKUP.BUP. If that saveset does not exist on the input device, BUP returns an error message.

For example, the following command restores to DU0 all the files in the saveset BACKUP.BUP on DU1:

```
.BACKUP/RESTORE DU: DU1:
```

1. **Restoring complete savesets**

   You can restore complete savesets in the following three ways:

   - By restoring all the files from a saveset
   - By restoring a complete device image from a saveset
   - By restoring a saveset as a file
     (restoring a logical-disk file from a saveset)

   a. Restoring all the files from a saveset (/RESTORE [ssname]/SAVESET)

      You can restore all the files in a saveset in one step by combining the /SAVESET option with the /RESTORE option to specify the name of the saveset you want to restore.

      The following command restores the saveset 28MAY.BUP from device MU0 to DL1, and verifies the restored data. As the files are restored from the saveset, they are listed on the terminal:

      ```
      .BACKUP/RESTORE/VERIFY MU0:28MAY.BUP/SAVESET DL1:
      ```

      See the /SAVESET option for more information on using that option.

   b. Restoring a complete device image from a saveset (/RESTORE/DEVICE)

      Restoring a device image from a saveset means restoring the entire image of a volume with home block, boot blocks, directory, and any empty blocks to a standard RT–11 formatted disk. This means when you restore a device image to a disk, the disk is initialized as part of the operation.

      If you do not specify the saveset name in the command line, BUP does one of the following:

      - On a magtape backup volume, BUP restores the first saveset on the magtape.
      - On a random-access device, BUP looks for a saveset matching the output device name. If BUP does not find such a name, BUP returns an error message.

      In the following command, MYDISK is the named saveset contained on magtape MS0. The command restores MYDISK to disk device DL1 and verifies the restoration.

      ```
      .BACKUP/RESTORE/DEVICE/VERIFY MS0:MYDISK DL1:
      ```

   c. Restoring a saveset as a file (/RESTORE/FILE)

      Restoring a saveset to a file is useful if you want to restore a saveset of a logical disk back to a logical disk. See the /FILE option for a description of this procedure. See Figure 1 for a diagram of what happens when you back up a logical disk into a saveset.

2. **Restoring individual files from savesets**

   To restore individual files from savesets, use the /RESTORE and /SAVESET options in the following general command format:

   BACKUP/RESTORE in-dev:[ssname/SAVESET,]file1[,file2,...] out-dev:

   where the input saveset can contain an entire device image or only individual files.

   Depending on your input device, one of two things can happen if you do not specify the saveset name (with the /SAVESET option) in the command line:

   - If the input device is a magtape, BUP attempts to restore the specified file or files from the first saveset encountered on the magtape.
   - If the input device is a random-access one, BUP looks for a saveset named BACKUP.BUP. If BUP does not find BACKUP.BUP, it returns an error message. For example, the following command causes BUP to look for saveset BACKUP.BUP on device DL0:

     ```
     .BACKUP/RESTORE/VERIFY DL0: DL1:
     ```

   Use commas to separate files, when you specify more than one to restore. You can also use wildcards to restore files of a particular name or type or to restore all files (*.*).

   Assuming the saveset 28MAY.BUP contains the file FOO.OBJ, you can restore that file to device DL1 and verify the restoration by using the following command:

   ```
   .BACKUP/RESTORE/VERIFY MS0:28MAY.BUP/SAVESET,FOO.OBJ DL1:
   ```

   This example assumes that magtape MS0 contains the entire file; if not, you will be prompted to mount the next magtape input volume.

3. **Restoring complete logical disks**

   If you back up logical disks as savesets, you can store them as files (see Figure 1) or as device images. So, the phrase *restoring a logical disk* can apply to any one of the following three BUP operations:

   - Restoring a logical-disk saveset as a device image

     See the description of the /DEVICE option. Remember: Although this case restores a a logical disk, the /DEVICE in the command initializes the output device. That is, the logical-disk image is written to the output volume on a block-for-block basis, starting at block 0.

   - Restoring a logical-disk saveset as a file

     See the description of the /FILE option.

- Restoring a logical-disk file from a saveset containing several logical-disk files

  See the preceding description of restoring individual files from savesets, contained in the description of the /RESTORE option.

4. **Restoring/extracting individual files from logical disks**

   The phrase *restoring individual files from a logical disk* can apply to either of the following two operations:

   - Restoring files from a device-image saveset of a logical disk
   - Extracting one or more files from a logical disk

   The first operation is a saveset one, while the second is a subset one. For a description of the first operation, see the preceding description of restoring individual files from savesets, contained in the description of the /RESTORE option.

   For a description of the second operation, see the description of the /SUBSET option.

   The following command extracts the file MEMO1.TXT from the logical disk FRIDAY.DSK on DL1 and places that file on device DL0:

   ```
   .BACKUP/RESTORE/VERIFY DL1:FRIDAY.DSK/SUBSET,MEMO1.TXT DL0:
   ```

/[NO]REWIND

Rewinds a magtape before performing the next operation. This is the default.

The /NOREWIND option inhibits the rewinding of the magtape before appending the next saveset to that magtape. This increases the speed of BACKUP operations but also omits saveset name verification (the magtape must rewind to check for unique saveset names).

If you intend to create a number of savesets on a magtape, you may want to use the /NOREWIND option. However, since BUP cannot then check that the saveset name you use is unique, Digital recommends that you always explicitly assign unique saveset names.

/NOREWIND also inhibits initial tape rewind when performing /DIRECTORY and /RESTORE operations.

Note: If a fetch or load causes a magtape handler to be read into memory, then an automatic REWIND operation will be performed. So you must explicitly load a magtape handler before using the /NOREWIND option.

/SAVESET

Used in combination with either the /RESTORE or /DIRECTORY options. The /SAVESET option, specified with the first input specification, indicates the saveset containing a file(s) you want to restore or the saveset from which you want to obtain directories.

/SAVESET can only be used as a qualifier on the first input file.

If you do not specify a saveset name:

- On a random-access device, BUP looks for the saveset BACKUP.BUP. If BUP does not find such a name, BUP returns an error message.
- On a magtape, BUP restores the files from the first saveset encountered on the magtape.

Examples

1. The following command restores all the files in the saveset 28MAY.BUP on device MS0 to DL1 and verifies the restored data:

   ```
   .BACKUP/RESTORE/VERIFY MS0:28MAY.BUP/SAVESET DL1:
   ```

2. Assuming the saveset 28MAY.BUP contains the file FOO.OBJ, you can restore that file to device DL1 and verify the restoration by using the following command:

   ```
   .BACKUP/RESTORE/VERIFY MS0:28MAY.BUP/SAVESET,FOO.OBJ DL1:
   ```

   See the /RESTORE and /DIRECTORY options for more information on using the /SAVESET option.

/[NO]SCAN

Scans each output disk for bad blocks. This is the default during the backup initialization procedure. /NOSCAN inhibits the bad-block scan on disk output volumes. Digital recommends that you use /NOSCAN only on disk output volumes that you know contain no bad blocks, and that you use /VERIFY to assure a reliable backup.

/SUBSET

Specified with the input file specification, the /SUBSET option allows you to do the following:

- Create logical-disk images of the files you want to back up
- List directories of logical disks
- Restore/extract one or more files from a logical-disk file

1. Creating logical-disk images of the files you want to back up

   You can use the /SUBSET option only on standard RT–11 disks, and not on BUP-formatted disks or any magtape volume. The standard RT–11 disk format is not changed, and the disk can continue to be used normally.

   You can mount a logical disk created by the /SUBSET option and perform standard logical-disk operations, such as copying, printing, and deleting files.

   The default output device is DK, the default output file name is BACKUP, and the default output file type is DSK. You can use wildcards in the input file specification.

The following command backs up all files on DU0 of type OBJ to a logical disk, OBJ.DSK on device DU1. The success of the operation is verified by including the /VERIFY option in the command:

```
.BACKUP/VERIFY DU0:*.OBJ/SUBSET DU1:OBJ
```

The command displays all files backed up to DU1. If DU1 does not contain sufficient free blocks for all the OBJ files, BUP returns an error message indicating insufficient space, and no files are backed up.

Advantages of Using BACKUP to Back Up Logical Disks

The advantages of using the BACKUP command to back up information into logical disks are the following:

- You perform no CREATE, INITIALIZE, or MOUNT operations; the /SUBSET option performs the equivalent of those operations for you.
- The logical disk created by the /SUBSET option is identical to one you create manually, except no free blocks are allocated and the number of directory segments is only sufficient to contain the files being backed up. This makes for efficient storage of the information in logical disks and for ease in accessing it, since it remains a logical disk.

Disadvantages of Using BACKUP to Back Up Logical Disks

The disadvantages of using the BACKUP command to back up information into logical disks are the following:

- The /SUBSET option is an alternative to the /SAVESET option, but, unlike savesets, logical-disk images created by the /SUBSET option must reside on a single backup volume.
- The /SUBSET option is appropriate only for file operations. You cannot back up entire volumes (disk images) to logical disks with this option. You can use the COPY/DEVICE command for that type of operation. This means that, although BACKUP with the /SUBSET option will move all the files on a disk to a logical disk, that command will not copy the boot blocks.

2. Listing directories of logical disks

   To get a directory of a logical disk, specify the [ldname]/SUBSET option with the BACKUP/DIRECTORY option.

The following example command displays on your terminal a directory of the logical disk MYBACK.DSK on DU0:

```
.BACKUP/DIRECTORY DU0:MYBACK.DSK/SUBSET

 RT-11 BACKUP
 04-Jan-91 10:50
 Subset:  DU0:MYBACK.DSK

 File         Blocks            Date

PROG1 .OBJ    15         Friday     14-Dec-90
PROG2 .OBJ    234        Friday     14-Dec-90
PROG3 .OBJ    49         Monday     03-Dec-90
MEMO1 .TXT    10         Wednesday  02-Jan-91
MEMO2 .TXT    2          Thursday   03-Jan-91

 5 Files, 310 Blocks
```

3. Extracting one or more files from a logical disk

You can extract one or more selected files from mounted or unmounted logical disks by using the following general command syntax:

BACKUP/RESTORE in-dev:ldname.dsk/SUBSET,file1[,file2,...] out-dev:

where:

in-dev: specifies the volume containing the logical disk.

ldname.dsk specifies the name of the logical disk.

file(n) specifies the file or files you want to restore from the logical disk. You can use wildcards.

out-dev specifies the volume to which you want the file(s) restored.

The following command extracts the files REPORT.* in logical disk WRK.DSK on SY, to device DK:

```
.BACKUP/RESTORE SY:WRK.DSK/SUBSET,REPORT.* DK:
```

See the /RESTORE option for more information on restoring logical disks.

/SYSTEM

(Used only with /RESTORE) Allows you to restore SYS files when using wildcards. This is to prevent you from accidentally losing or corrupting SYS files.

/VERIFY[:ONLY]

Verifies that output data matches input data in a BACKUP/RESTORE operation. Although verification slows down the operation, it is usually worth the time spent in terms of higher data integrity and confidence in your BACKUP operations.

Three Verification Procedures

Depending on how you specify it, the /VERIFY option performs one of three different verification procedures:

- **/VERIFY** verifies a data transfer as you are backing up the data.

For each volume that is backed up, /VERIFY validates that volume in a separate pass immediately after the data is backed up to that volume. The /VERIFY option compares the original to the backed-up data.

- **/RESTORE/VERIFY** verifies a data transfer as you are restoring the data.

  For each volume that is restored, /RESTORE/VERIFY verifies that volume, record for record, as it is being restored. Because this verification procedure is different from /VERIFY and /RESTORE/VERIFY:ONLY, /RESTORE/VERIFY is less sensitive to position errors (for example, a slight slip in a magtape) than the other two verification procedures. It simply verifies that each restored data block can be correctly read.

- **/RESTORE/VERIFY:ONLY** verifies a data transfer after you back up the data but before you change or delete the original data.

  The /RESTORE/VERIFY:ONLY option compares the backed-up data to the original. That is, /RESTORE/VERIFY:ONLY first reads the data from the backed-up volume (as in a RESTORE operation), then reads the original data and compares the two. It simulates a RESTORE operation except for the actual restoring of the data.

  The /VERIFY:ONLY option is valid only when used with the /RESTORE option and when the original data is as it was before a backup.

Each verification procedure is alike in that they all:

- Check for device (read) errors.
- Check for data integrity of the blocks read.

The following command line includes the /VERIFY option to perform a post-backup validation of each output saveset section.

```
.BACKUP/VERIFY DU0:LGFIL.DAT DU1:
```

Verification Messages

Depending on the type of verification, when BUP starts the verification process, BUP displays one of the following messages on the terminal:

```
?BUP-I-Verify pass started
?BUP-I-Restore/Verify operation started
```

Again, depending on the type of verification, if the verification is successful, BUP displays one of the next messages:

```
?BUP-I-Backup/Verify operation is completed
?BUP-I-Restore/Verify operation is completed
```

If the output data and the input data differ, BUP displays the error message:

```
?BUP-F-Verification error <dev:file.type>
```

BOOT

(DUP)

Directs a new monitor to take control of the operating system, or reads into memory a new copy of the monitor that is currently controlling the system.

Format

BOOT *input-spec*

Command Options

[/FOREIGN]
[/WAIT]

Parameter

input-spec

Specifies the device or monitor file to be bootstrapped. If you omit the input-spec, RT–11 prompts you with *Device or file?*.

Description

The BOOT command can perform either of two operations: a boot-block bootstrap or a monitor-file bootstrap.

- **Boot-block bootstrap**

  The boot-block (hardware) bootstrap operation gives control of the operating system to the monitor whose bootstrap code is written *in the boot blocks* of a volume in a device. You can change the monitor booted by a hardware bootstrap by issuing the COPY/BOOT command.

  To perform a boot-block bootstrap:

  1. Make sure the appropriate device handler is present on the volume you want to bootstrap.
  2. Specify only a device in the command line. You can do this by entering the device's physical or logical name—if it has been assigned a logical name (see the ASSIGN command).

  The following example bootstraps the XM monitor, RT11XM, whose bootstrap information is written on the boot blocks of the system device:

  ```
  .BOOT SY:
  RT-11XM  V05.6
  ```

- **Monitor-file bootstrap**

  The monitor-file (software) bootstrap operation gives control of the operating system to the monitor whose bootstrap code is written *in a monitor file* on a volume in a device.

  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.

  The following example bootstraps the XM monitor on device SY:

  ```
  .BOOT RT11XM
  RT-11XM  V05.06
  ```

  You can use the BOOT command to alternate between monitors. When you do so, you do not have to reenter the date and time. The system clock, however, can lose a few seconds during a reboot.

NOTE

If you are running a foreground or system job that is sending input to the system volume, using a monitor-file BOOT command can cause your system to hang. You should terminate such a job in the foreground before using such a BOOT command.

Options

/FOREIGN

Boots a pre-RT–11 V4 volume or a non-RT–11 operating system. The /FOREIGN option does not preserve the date or time, and you cannot specify a file name with this option.

/WAIT

Initiates the BOOT procedure but then pauses and waits for you to mount the volume you want to bootstrap. This is useful if you have a single-disk system, or if you want to bootstrap a different volume in the drive unit currently occupied by your system volume.

When the system pauses, it displays the following message at the terminal:

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

<device> represents the device into which you mount the volume. You can respond in two ways:

- To continue, mount the volume you want to bootstrap, and type Y RETURN.
- To abort the operation and return control to the original monitor, type N RETURN, or press CTRL/C twice.

Any other response causes the message to repeat.

The following sample command boots an MSCP diskette:

```
.BOOT/WAIT DU1:
Mount input volume in DU1:; Continue? Y
```

CLOSE

(KMON)

Closes and makes permanent all output files that are currently open in the background job.

Format

CLOSE

Description

The CLOSE command accepts no options or arguments.

You can use the CLOSE command to make tentative open files permanent; otherwise, they are lost.

The CLOSE command is particularly useful in two cases:

- After you type a CTRL/C to abort a background job
- After an unexpected program termination

In these cases, CLOSE preserves any new files that were being used by the terminated program.

The CLOSE command has no effect on a foreground job and will not make permanent any files opened on magnetic tape.

The CLOSE command does not work if your program defines new 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.

Example

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

```
.R PROG RET
   .
   .
   .
CTRL/C  CTRL/C
.CLOSE RET
```

COMPILE

(DIBOL, FORTRA, F–77, F77XM, MACRO, and CREF)

Invokes the appropriate language processor to compile the files you specify.

Format

COMPILE *filespec[s]*

Command Options	**Filespec Options**
/LIST[:filespec] /ALLOCATE:size	/LIBRARY
/[NO]**OBJECT**[:filespec] /ALLOCATE:size	
/DIBOL	
/ALPHABETIZE /CROSSREFERENCE /[NO]**LINENUMBERS** /LOG /ONDEBUG /PAGE:length /TABLES /[NO]**WARNINGS**	
/FORTRAN	
/CHECK /CODE:type /CONTINUATIONS:value /DIAGNOSE /EXTEND **/F4** or /F77 /HEADER /I4 /[NO]**LINENUMBERS** /ONDEBUG /[NO]**OPTIMIZE** /RECORD:length /SHOW[:type] /STATISTICS /[NO]**SWAP** /TRACE:type /UNITS:value /[NO]**VECTORS** /[**NO**]**WARNINGS** /WIDE /WORKFILE:value	

Command Options	Filespec Options
/MACRO	
/CROSSREFERENCE[:type[:type...]] /DISABLE:type[:type...] /ENABLE:type[:type...] /[NO]SHOW:type[:type...]	

Parameter

filespec[s]

Specifies one or more files to be included in the assembly or compilation.

Compiling Multiple Source Files

You can compile up to six source files into a single object file or into multiple object files.

- To compile multiple source files *into a single object file*, separate the files with plus (+) signs in the command line. RT–11 creates an object file with the same name as the first input file and gives it an OBJ file type, unless you specify a different name and type.
- To compile multiple source files *in independent compilations*, separate the files with commas (,) in the command line. This generates a corresponding object file for each input file or for each set of input files.

Description

The following are the default input and output file types.

Default File Types

File Type	Description
DBL	DIBOL source file
FOR	FORTRAN source file
LST	Listing file
MAC	MACRO source file
MLB	MACRO macro-library file
OBJ	Object file

Choosing a Language Processor

You can specify a language processor with the COMPILE command in any of the following three ways:

- Specify a language-name option, such as /MACRO, which invokes the MACRO assembler. Since there are two FORTRAN compilers, see the next section (Choosing the FORTRAN IV or the FORTRAN–77 Compiler) for how to override the default FORTRAN compiler.
- 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.
- 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), RT–11 issues an error message.

Choosing the FORTRAN IV or the FORTRAN–77 Compiler

The meaning of the /FORTRAN option can be changed, depending on the condition set by the SET FORTRA command. By default, or if SET FORTRA F4 has been issued, the COMPILE/FORTRAN command calls the FORTRAN IV compiler. If SET FORTRA F77 has been issued, the COMPILE/FORTRAN command calls the FORTRAN–77 compiler. The FORTRAN–77 compiler under a mapped monitor is F77XM.SAV and under an unmapped monitor is F77.SAV.

The current condition at which FORTRAN is set can be overridden, using the /F4 option to specify FORTRAN IV or /F77 to specify FORTRAN–77. Options supported only under FORTRAN–77 are indicated by *FORTRAN–77 only*. Also, some options, such as /SHOW:val, take different arguments or are otherwise different than that option under FORTRAN IV. Therefore, before using the FORTRAN options, you should see the FORTRAN command description.

Position Dependence of Language Options

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.

Options

/ALLOCATE:size

Used with /LIST or /OBJECT, reserves space on the device for the output file. The argument *size* specifies the number of blocks of space to allocate. The meaningful range for this value is from -1 to 65527. A value of -1 is a special case that creates the largest file possible on the device.

/ALPHABETIZE

Alphabetizes the entries in the symbol table listing. This is useful for program maintenance and debugging.

/CHECK

(FORTRAN–77 only) Checks array references to ensure they are within the specified array address boundaries. The FORTRAN IV functionality of checking is done by the selection of OTS library components, not by compiler action.

/CODE:type

(/FORTRAN IV only) Produces 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 valid 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 functions.

/CONTINUATIONS:value

(FORTRAN–77 only) Specifies the maximum decimal number of continuation lines in the program. The valid range for *value* is 0 through 99; the default is 19.

/CROSSREFERENCE[:type[:type...]]

(DIBOL or MACRO) Generates 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 optional arguments. 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 description for a summary of valid arguments and their meaning.

/DIAGNOSE

(FORTRAN IV only) Helps 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

Invokes the DIBOL language processor to compile the associated files.

/DISABLE:type[:type...]

(MACRO only) Specifies values that override the .DSABL directive in the source file. See the /DISABLE description of the MACRO command for a summary of the *type* arguments and their meanings. See the *PDP–11 MACRO–11 Language Reference Manual* for a description of the directive and a list of all valid types.

/ENABLE:type[:type...]

(MACRO only) Specifies values that override the .ENABL directive in the source file. See the /DISABLE description of the MACRO command for a summary of the *type* arguments and their meanings. See the *PDP–11 MACRO–11 Language Reference Manual* for a description of the directive and a list of all valid types.

/EXTEND

(FORTRAN IV or FORTRAN–77) Used with /FORTRAN, changes the right margin for source input lines. With FORTRAN IV, /EXTEND extends the right margin from column 72 to column 80. With FORTRAN–77, /EXTEND extends the right margin of input lines from 72 to 132 columns.

/F4

Overrides any SET FORTRA conditional and calls the FORTRAN IV compiler (the default).

/F77

Overrides any SET FORTRA conditional and calls the appropriate FORTRAN–77 compiler.

/FORTRAN

Invokes the FORTRAN language processor to compile the associated files.

By default, or if SET FORTRA F4 has been issued, the COMPILE/FORTRAN command calls the FORTRAN IV compiler. If SET FORTRA F77 has been issued, the COMPILE/FORTRAN command calls the FORTRAN–77 compiler. The FORTRAN–77 compiler under a mapped monitor is F77XM.SAV and under an unmapped monitor is F77.SAV. The FORTRAN IV compiler is FORTRA.SAV. See the FORTRAN command for more information.

/HEADER

(FORTRAN IV only) Includes in the printout a list of options that are currently in effect.

/I4

(FORTRAN IV or FORTRAN–77) Allocates two words for the default integer data type (FORTRAN IV uses only one-word integers) so that it takes the same physical space as real variables. With FORTRAN–77, /I4 allocates and uses two words for the default integer and logical variable length.

/LIBRARY

(MACRO only) Identifies 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
```

/[NO]LINENUMBERS

(DIBOL or FORTRAN) Includes internal sequence numbers in the executable program. These numbers are especially useful in debugging programs. This is the default operation.

/NOLINENUMBERS, used with /DIBOL or /FORTRAN, 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 DIBOL or FORTRAN error messages are difficult to interpret.

/NOLINENUMBERS, with FORTRAN–77, requests no traceback information.

/LIST[:filespec]

You must specify this option to produce a compilation or assembly listing.

The /LIST option has the following 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 COMPILE command, the system generates a listing that prints on the line printer.
- If you specify /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. For example, 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 DU1:

  ```
  .COMPILE/LIST:DU1: 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 and file type.

  The following command compiles A.FOR and B.FOR together, producing files A.OBJ and AANDB.OUT on device DK:

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

- If you specify /LIST after the file specification to which it applies, you produce a listing file with the same name as a particular input file.

  The following command compiles A.DBL and B.DBL together, producing files DK:A.OBJ and DU1:B.LST:

```
.COMPILE/DIBOL A+B/LIST:DU1:
```

- 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
  ```

  The preceding commands generate A.OBJ and B.LST on device DK as output files.

  Remember that file options apply only to the file (or group of files separated by plus signs) that 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.

/LOG

(DIBOL only) Creates a log of error messages generated by the compiler.

/MACRO

Invokes the MACRO assembler to assemble the associated files.

/[NO]OBJECT[:filespec]

/OBJECT specifies a file name or device for the object file. Note that any time you type a colon after the /OBJECT option (/OBJECT:), you must specify a device or a file specification after the colon.

Because the COMPILE command creates object files by default, the following three commands have the same meaning:

```
.COMPILE/FORTRAN A
.COMPILE/FORTRAN/OBJECT A
.COMPILE/FORTRAN A/OBJECT
```

These 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 DU1:

```
.COMPILE/OBJECT:DU1: (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 suppresses the 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.

The following command uses /NOOBJECT as a file option. In this command, 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
```

The next command uses /NOOBJECT as a command option. In this command, all object files are suppressed:

```
.COMPILE/NOOBJECT A,B,C
```

/ONDEBUG

(DIBOL or FORTRAN)

- Used with /DIBOL, includes an expanded symbol table in the object file. You can then use a debugging program to find and correct errors in the object file.
- Used with FORTRAN, includes 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.

/[NO]OPTIMIZE

(FORTRAN–77 only) /OPTIMIZE enables compiler optimization (the default).

/NOOPTIMIZE inhibits compiler optimization.

/PAGE:length

(DIBOL only) Overrides the default listing page length of 66 lines. The meaningful range of values for the decimal argument *length* is 1 to 32768_{10}.

/RECORD:length

(FORTRAN IV or FORTRAN–77) Overrides the default maximum record length for sequentially formatted input and output. The meaningful range for the argument length is from 4 to 4095. For FORTRAN IV, the default maximum record length is usually 132 characters, while for FORTRAN–77, the default length is usually 136 characters.

/[NO]SHOW[:type]

(FORTRAN IV, FORTRAN–77, or MACRO) /SHOW:type, used with /FORTRAN, controls FORTRAN listing format. The argument type represents a code that indicates which listings the compiler is to produce. See the description of the /SHOW option under the FORTRAN command for a table listing the FORTRAN *type* arguments.

Use the /SHOW option with /MACRO to specify any MACRO .LIST directive. See the description of the /SHOW option under the MACRO command for a table listing the MACRO *type* arguments. The *PDP–11 MACRO–11 Language Reference Manual* explains how to use these directives.

/NOSHOW:type, used with MACRO, specifies any MACRO .NLIST directive. See the description of the /SHOW option under the MACRO command for a table listing the MACRO .NLIST directives. The *PDP–11 MACRO–11 Language Reference Manual* explains how to use these directives.

/STATISTICS

(FORTRAN IV or FORTRAN–77) Includes compilation statistics in the listing, such as amount of memory used, amount of time elapsed, and length of the symbol table.

With FORTRAN–77, /STATISTICS includes in the compilation statistics workfile access statistics and the number of pages of dynamic memory available and used.

/[NO]SWAP

(FORTRAN IV or FORTRAN–77) /SWAP permits the USR (User Service Routine) to swap over the FORTRAN program in memory. This is the default operation.

/NOSWAP 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 calls (see the *RT–11 System Subroutine Library Manual*). If the program frequently updates or creates a large number of files, making the USR resident can improve program execution. However, the cost for making the USR resident is 2K words of memory.

/TABLES

(DIBOL only) Generates a symbol table and label table as part of the listing. This information is useful for program maintenance and debugging. The system does not generate a listing by default. You must also specify /LIST in the command line to produce an assembly listing.

/TRACE:type

(FORTRAN–77 only) Enables the FORTRAN–77 /S:xxx option. This controls the generation of code used by the OTS during error traceback. See the /TRACE option description of the FORTRAN command for the *type* arguments.

/UNITS:value

(FORTRAN IV or FORTRAN–77) Overrides the default number of logical units (6) to be open at one time. The *value* argument is a decimal number. With FORTRAN IV, the maximum number you can specify for *value* is 16. With FORTRAN–77, the maximum is 99.

/[NO]VECTORS

(FORTRAN IV only) /VECTORS directs the FORTRAN compiler to use tables to access multidimensional arrays. This is the default mode of operation.

/NOVECTORS directs the FORTRAN compiler to use multiplication operations to access multidimensional arrays.

/[NO]WARNINGS

(DIBOL, FORTRAN IV, or FORTRAN–77) /WARNINGS includes 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, used with /DIBOL or /FORTRAN, suppresses 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

(FORTRAN–77 only) Produces a 132-column compiler listing.

/WORKFILE:value

(FORTRAN–77 only) Sets the workfile size. The default is 128_{10} blocks. By default, *value* is octal; include a decimal point after *value* to specify a decimal radix.

COPY

(PIP, DUP, and FILEX)

Transfers:

- One file to another file
- A number of files to a single file by concatenation
- Files from a large volume to several smaller volumes
- The bootstrap code on a volume to the boot blocks of that volume
- The contents of a volume to a file and vice versa
- The contents of a device to another device

The following three tables respectively summarize the three types of COPY commands: those that run PIP (file commands), DUP (device commands), and FILEX (interchange commands).

Format

COPY *input-file[s] output-file*

COPY File Operations

Command Options	Input-File Options	Output-File Options
/ASCII /BINARY **/IMAGE**	/POSITION:value	/POSITION:value
[/BEFORE[:date] /SINCE[:date]] /DATE[:date] /NEWFILES		
/CONCATENATE		
/DELETE		
/EXCLUDE		
/IGNORE		
/INFORMATION		
/[NO]LOG		
/MULTIVOLUME		
/PREDELETE		
/[NO]PROTECTION		
/[NO]QUERY		
/[NO]**REPLACE**		
/SETDATE[:date]		
/SLOWLY		
/SYSTEM		
/VERIFY		
/WAIT		

COPY Device Operations

Command Options	Input-File Options	Output-File Options
/BOOT[:dev] /WAIT /DEVICE [/FILES /WAIT] **/IMAGE** /VERIFY	/END:value /START:value	/ALLOCATE:size /START:value

COPY Interchange Operations

Command Options	Input-File Options	Output-File Options
/ASCII **/IMAGE** /PACKED /QUERY /WAIT	/DOS /OWNER:[nnn,nnn] /INTERCHANGE /TOPS	/DOS /INTERCHANGE[:size]

Parameters

input-file[s]

Specifies the data to copy. The input file specification can be a device name, if you use the /DEVICE option. Otherwise, you can specify as many as six file specifications for input. Specifying a wildcarded device name without the /DEVICE option is equivalent to specifying all the files on a device minus the boot blocks.

Normally, commas separate the input files if you individually 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.

output-file

Specifies the device or file to receive the data. You can specify only one output device or file.

Description

You should be aware of the protection, wildcard, system-file, bad-block, and date features of the COPY command.

Protection Features

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 and type already exists on the output volume.

When you copy files, the protection status of the output file will be the same as the protection status of the input file, unless you use the /PROTECTION or /NOPROTECTION option.

Using Wildcards

You can use wildcards in the input or output file specification of the command. However:

- The output file specification cannot contain embedded wildcards.
- 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.

Copying SYS Files and Files with Bad Blocks

RT–11 has a special way of handling system (SYS) files and files that cover bad blocks (BAD files).

- RT–11 requires you to use the /SYSTEM option to copy system files when you use wildcards in the input file type or when you use the /EXCLUDE option. You cannot copy system files simply by placing wildcards in file specifications. However, you do not have to specify the /SYSTEM option, if you specify a SYS file type.
- To copy a BAD file, you must specify it by explicitly giving its file type. (You can use wildcards when specifying the file name.) Since BAD files cover bad blocks on a device, you usually do not need to copy, delete, or otherwise manipulate these files.

  With RT–11 file-structured device copy operations, the bad-block replacement table on the RL01, RL02, RK06, or RK07 output device is automatically preserved.

Specifying a Date in a COPY Option

Some COPY options accept a date as an argument. The syntax for specifying the date is:

[dd][:mmm][:yy]

where:

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

The default value for the date is the current system date. If you omit any of these values (dd, mmm, or yy), the system uses the values from the current system date. For example, if you specify only the year 90 and the current system date is May 4, 1991, the system uses the date 4:MAY:90. If the current 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 at the beginning of each month with the DATE command. If you fail to set the date at the beginning of each month, the system displays -*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.)

Note that references to unit numbers for TT are currently ignored. For example, COPY TT1: is treated as COPY TT0:

Options

/ALLOCATE:size

Specify this option after the output file specification to reserve space on the device for the output file. /ALLOCATE:size requires the DUP utility. The *size* argument specifies the number of blocks of space to allocate. The meaningful range for this value is from -1 to 65527. A value of -1 is a special case that creates the largest file possible on the device.

/ASCII

Copies files in ASCII mode, ignoring and eliminating nulls (0_8) and rubout (177_8) characters. It forces the high bit of each byte to zero and treats CTRL/Z (32_8) 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 /ASCII option cannot be used with /VERIFY. The following command copies a FORTRAN source program from DU0 to DU1, giving it a new name, and reserving 50 blocks of space for it:

```
.COPY/ASCII DU0:MATRIX.FOR DU1:TEST.FOR/ALLOCATE:50
```

/COPY/ASCII/CONCATENATE can be used to take ASCII files and compress them. For example, if you combined three 3-block files, your combined file could be less than 9 blocks, since the unused space in each of those files is moved to the end of the single combined file.

NOTE

Do not use the COPY/ASCII command to copy JOU files. Copy them in IMAGE mode (the default).

/BEFORE[:date]

Copies all files on a device created before a specified date. The following command copies only those MAC files on DK created before February 4, 1990:

```
.COPY/BEFORE:4:FEB:90 *.MAC DU0:*.MAC
 Files copied:
DK:A.MAC  to DU0:A.MAC
DK:B.MAC  to DU0:B.MAC
DK:C.MAC  to DU0:C.MAC
```

/BINARY

Copies formatted binary files, such as OBJ files produced by the assembler or the FORTRAN compiler, and LDA files produced by the linker. RT–11 verifies checksums and displays a warning if a checksum error occurs. If this happens, the COPY operation does not complete. You cannot use the /BINARY option with /VERIFY. The following command copies a binary file from DK to DU1:

```
.COPY/BINARY ANALYZ.OBJ DU1:*.*
```

/COPY/BINARY/CONCATENATE can be used to take BINARY files and compress them. For example, if you combined three 3-block files, your combined file could be less than 9 blocks, since the unused space in each of those files is moved to the end of the single combined file.

Note that you cannot copy library files with the /BINARY option because they contain formatted binary data and unformatted binary data. Copy them in IMAGE mode.

/BOOT[:dev]

Copies bootstrap information from 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. /BOOT requires the DUP utility. The optional argument *dev* specifies the name of the handler to be used on the target system device.

Note:

- You can combine /BOOT only with the /WAIT option.
- Your input and output volume must be the same.
- You can name your monitor file any name you wish; the default file type is SYS.

When you perform this operation, you must have the correct device handler to go with the volume. For example, to create a bootable DU01 disk, you must have the handler file DU.SYS or DUX.SYS on that DU01.

To create a bootable volume, do the following:

1. Initialize the volume, using the INITIALIZE command. (Note that if the volume is an RK06/07 or an RL01/02, you should also use the /REPLACE option.)

2. Copy the system files you need onto the volume, using the COPY/SYSTEM or SQUEEZE/OUTPUT command.
3. Write the monitor bootstrap onto the volume, using COPY/BOOT.

The following example creates a bootable system diskette with a minimal number of system files that you might need:

```
.INITIALIZE DU1:
DU1:/Initialize; Are you sure? Y
```

```
.COPY SY:(RT11FB,SWAP,DU,SL,LD).SYS DU1:*.*
 Files copied:
DU0:RT11FB.SYS  to DU1:RT11FB.SYS
DU0:SWAP.SYS    to DU1:SWAP.SYS
DU0:DU.SYS      to DU1:DU.SYS
DU0:SL.SYS      to DU1:SL.SYS
DU0:LD.SYS      to DU1:LD.SYS
```

```
.COPY SY:(PIP,DUP,DIR,RESORC).SAV DU1:*.*
 Files copied:
DU0:PIP.SAV     to DU1:PIP.SAV
DU0:DUP.SYS     to DU1:DUP.SYS
DU0:DIR.SAV     to DU1:DIR.SAV
DU0:RESORC.SAV  to DU1:RESORC.SAV
```

```
.COPY/BOOT DU1:RT11FB.SYS DU1:
```

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.

/CONCATENATE

Combines several input files into a single output file. This option is 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 DU1 into a file called MERGE.FOR on DU0:

```
.COPY/CONCATENATE DU1:*.FOR DU0:MERGE.FOR
 Files copied:
DU1:A.FOR  to DU0:MERGE.FOR
DU1:B.FOR  to DU0:MERGE.FOR
DU1:C.FOR  to DU0:MERGE.FOR
```

Wildcards are invalid in the output file specification.

/DATE[:date]

Copies only those files with a certain creation date. If you do not specify a date, the current system date is used. The following command copies all MAC files created on February 20, 1990 from DU0 to DU1:

```
.COPY/DATE:20:FEB:90 DU0:*.MAC DU1:*.*
 Files copied:
DU0:A.MAC  to DU1:A.MAC
DU0:B.MAC  to DU1:B.MAC
DU0:C.MAC  to DU1:C.MAC
```

/DELETE

Deletes the input file after it has been copied. The COPY/DELETE operation does not ask you for confirmation before it executes. You must use /QUERY for this function. If the input specification and output specification are the same, the file is not deleted. The following command copies JSPROG.SAV to DU1, then deletes it from device DK:

```
.COPY/DELETE JSPROG.SAV DU1:JSPROG.SAV
```

/DEVICE

Copies block for block the image of one device to another, and copies all the 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. /DEVICE requires the DUP utility.

You can copy disks that are not in RT–11 format if they have no bad blocks. When copying RT–11 disks, you should ensure the integrity of the results by making sure the disks involved contain no bad blocks. If RT–11 encounters a bad block during the COPY/DEVICE operation, it displays 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 may copy the entire directory of the input volume, but not every file on the input volume. When you copy a larger device to a smaller one, you are asked to confirm the COPY operation. If you also use the /START and /END options with the input specification, the confirmation is requested only if the number of blocks to be copied is greater than the area on the output volume defined by the /START option and the end of the output 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 512-byte blocks. Because magtape is not file structured, there is room for only one disk image on a magtape. When you use the /DEVICE option with magtape, you must also use the /FILES option with the magtape input or output specification.

The following command copies an image of DU0 to DU1:

```
.COPY/DEVICE DU0: DU1:
DU1:/Copy; Are you sure? Y
```

Respond to the query message by typing Y and 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 RX01 and RX02 diskettes. However, this restriction has no impact on any COPY operations if your diskette was supplied by Digital.

The /DEVICE and /POSITION options are mutually exclusive.

/DOS

Transfers files between RSTS/E or DOS–11 format and RT–11 format. /DOS requires the FILEX utility. The option must appear in the command line after the file to which it applies. Valid DOS input devices are DECtape and RK05; the only valid DOS output device is DECtape. The only other options allowed with /DOS are /ASCII, /BINARY, /IMAGE, /OWNER:[nnn,nnn], and /WAIT (using two device drives).

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:value

Used with /START:value and /DEVICE to specify the last block of the volume you are copying. /END:value requires the DUP utility.

The /END:value notation must follow the input file specification. The *value* argument represents a decimal block number. The following example copies blocks 0 to 500 from DU0 to DU1, starting at block 501, in a file named ADAM.MAC:

```
.COPY/DEVICE/FILES DU0:/START:0/END:500  DU1:ADAM.MAC/START:501
```

/EXCLUDE

Copies all the files on a device except the ones you specify. The following command copies all files from DU0 to DU1 except OBJ and SAV files:

```
.COPY/EXCLUDE DU0:(*.OBJ,*.SAV) DU1:*.*
```

Note that if you are copying system (SYS) files, using the /EXCLUDE option, you must also use the /SYSTEM option.

/FILES

Used with /DEVICE to copy a volume to a file on another volume or vice versa. /FILES requires the DUP utility.

If you use a magtape in the operation, you must specify a file name and the /FILES option with the magtape. Do not include wildcards in either the input or output specification when you use the /FILES option.

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 command copies diskette DU1 to DU2 as the file FLOPPY.BAK:

```
.COPY/DEVICE/FILES DU1:  DU2:FLOPPY.BAK
```

The next command copies the file MYWORK.BAK to DU1:

```
.COPY/DEVICE/FILES  MYWORK.BAK DU1:
```

/IGNORE

Ignores errors during a COPY operation. /IGNORE forces a single-block data transfer, which you can invoke at any 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 is displayed on the terminal, but the COPY operation continues. This option is invalid with /DOS, /TOPS, and /INTERCHANGE.

When coupled with /DEVICE, /IGNORE causes any errors returned by a bad block on the input or output device to be ignored. The bad block on the device that returns the error and a corresponding block on the other device are not copied. An error message displays which device (input or output) contains the bad block and the bad-block number.

/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. Note, however, that you should not transfer memory-image files to the printer or terminal.

Copying files, using the default (COPY/IMAGE), always correctly copies any RT–11 formatted file. An advantage to using /IMAGE is that it is faster than /ASCII and /BINARY.

The following command copies a text file to a diskette for storage:

```
.COPY/IMAGE LETTER.SAV DU1:*.*
```

/INFORMATION

Changes the severity level of the error message that displays when not all of the input files you specified are found. If you do not use /INFORMATION, RT–11 displays an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, RT–11 displays an informational message to tell you which files it cannot find, but execution continues.

The following command copies input files FILE1.TXT and FILE3.TXT to DU1. However, since RT–11 is unable to find DU0:FILE2.TXT, RT–11 displays a message to inform you that.

```
.COPY/INFORMATION DU0:(FILE1,FILE2,FILE3).TXT  DU1:*.*
?PIP-I-File not found DU0:FILE2.TXT
```

/INTERCHANGE[:size]

Transfers data in interchange format between interchange diskettes that are compatible with IBM 3741 format and RT–11 block-replaceable devices or the terminal. This option requires the FILEX utility.

/INTERCHANGE 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).

If you use the /WAIT option with /INTERCHANGE, you must use two device drives for the operation.

The following command transfers the RT–11 file PROG.MAC from device DK to device DX1 in interchange format, giving it the name PROG.MA. The record length is set to 128_{10} bytes.

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

/[NO]LOG

/LOG lists on the terminal the names of the files that were copied by the current command. Normally, RT–11 displays a log only if there is a wildcard in the file specification. If you specify /QUERY, RT–11 displays 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 DU1:FILE.MAC  DU0:FILE.MAC
 Files copied:
DU1:FILE.MAC  to DU0:FILE.MAC
```

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

/MULTIVOLUME

Copies files from an input volume to one or more output volumes. This option is useful when you are copying several files from a large input volume to a smaller output volume and you are not sure all the files will fit on one output volume.

When you use this option, RT–11 copies files to the output volume until RT–11 finds a file that will not fit. RT–11 continues to search that file's directory segment, copying all files from that segment that will fit onto the output volume. When no more files from that segment will fit on the output volume, RT–11 prompts you to mount the next output volume and displays the *Continue?* message.

Mount another output volume of the same type and type Y. RT–11 begins the COPY operation with the first file that did not fit on the previous output volume. If you type N or CTRL/C, the operation is not completed and the monitor prompt (.) appears.

RT–11 continues to copy files from that directory segment until no more files from that segment will fit on the output volume or until all files from that directory segment have been copied. When all files from that segment have been copied, RT–11 begins copying files from the next directory segment. File copying continues in this fashion until all the specified input files have been copied.

The following example shows all files on DU0 being copied to several diskettes:

```
.COPY/MULTIVOLUME DU0:*.* DU1:

(Log of files copied)

Mount next output volume in DU1:;  Continue? Y

(Log of files copied)

Mount next output volume in DU1:;  Continue? Y

(Log of files copied)

Mount next output volume in DU1:;  Continue? Y

(Log of files copied)

Mount next output volume in DU1:;  Continue? Y
```

The /MULTIVOLUME option is not valid when you are copying from magtape.

/NEWFILES

Copies 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 *.* DU1:*.*
 Files copied:
DK:A.FOR  to DU1:A.FOR
DK:B.FOR  to DU1:B.FOR
DK:C.FOR  to DU1: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. This option requires the FILEX utility.

Note that the square brackets are part of the UIC; you must type them. The initial default for the UIC is [1,1].

/PACKED

Copies files in DECsystem–10, DOS, or interchange mode. This option requires the FILEX utility.

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:value

Use this option when you copy files to or from magtape. The /POSITION:value 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 *value* argument has the same effect as setting *value* equal to 0 (*value* is interpreted as a decimal number). Since this option applies to the device and not to the files, you can specify one /POSITION:value option for the output file and one for the input file.

The /POSITION and /DEVICE options are mutually exclusive.

The following table lists the operation initiated by each *value* specified with the /POSITION:value option in a magtape read (copy from tape) operation.

Magtape Read Operations Specified with the /POSITION:value Option

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

The following table lists the operation initiated by each *value* specified with the /POSITION:value option in a magtape write (copy to tape) operation.

Magtape Write Operations Specified with the /POSITION:value Option

Value	Operation
0	The tape rewinds before the handler copies each file. A warning message displays 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.
A positive integer	The handler goes to file-sequence number *value* 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.

Value	Operation
	If the handler finds the sequence number *value*, it creates a new logical end-of-tape. If there are any files with a sequence number greater than *value*, they are lost.
-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.
-2	The tape rewinds between each copy operation. The handler enters the file you specify at the 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).

See the PIP utility chapter in the *RT–11 System Utilities Manual* for more detailed information about operations involving magtape.

/PREDELETE

Deletes a file on the output device that has the same file name and type as a file you copy to that device, before the copy occurs. Normally RT–11 deletes a file of the same file name and type after the copy operation successfully completes.

This option is useful for operations involving devices that have limited space, such as diskettes.

Warning: 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.

/PREDELETE is invalid with magtape as output.

/[NO]PROTECTION

/PROTECTION gives an output file protected status so that it cannot be deleted.

Note that if a file is protected, you cannot perform any operations on the file that result in its deletion. You can copy a protected file to another volume, change its name, or write to it. However, you cannot delete a protected file; you must first change its protection status by using the /NOPROTECTION option.

If during a copy operation neither the /PROTECTION nor the /NOPROTECTION option is specified, the output file retains the protection status of the input file.

/NOPROTECTION enables an output file for deletion. When you use the /NOPROTECTION option during a COPY operation, the resulting output file is enabled for deletion.

Files that have been assigned as logical disks are protected. You should not use this option to remove protection from an active logical-disk file.

/[NO]QUERY

/QUERY makes RT–11 request confirmation before it performs the operation. /QUERY is 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.

If you specify /QUERY in a COPY command line that also contains a wildcard in the file specification, the confirmation messages that display on the terminal replace the log messages that would normally appear. Respond to a query message by typing Y (or any string that begins with Y) and RETURN. RT–11 interprets any other response to mean NO, and it does not copy the file.

The following example copies three of the four MAC files stored on DK to DU1:

```
.COPY/QUERY DK:*.MAC DU1:*.*
 Files copied:
DK:A.MAC       to DU1:A.MAC  ? Y
DK:B.MAC       to DU1:B.MAC  ? Y
DK:C.MAC       to DU1:C.MAC  ? N
DK:DEMOF1.MAC  to DU1:DEMOF1.MAC? Y
```

/NOQUERY suppresses the confirmation message that RT–11 displays 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 when you use /NOQUERY.

/[NO]REPLACE

/REPLACE 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, RT–11 deletes the duplicate file after the copy operation successfully completes.

/NOREPLACE 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[:date]

Puts the date you specify on all files it copies. If you specify no date, RT–11 uses the current system date. If the current system date is not set, RT–11 places zeros in the directory entry-date position. Normally, RT–11 preserves the existing file-creation date when it copies a file.

This option is invalid for magtape operations; RT–11 always uses the current date when copying to magtape, and always uses the magtape file's creation date when copying from magtape.

/SINCE:[date]

Copies all files on a specified device that were created on or after a specified date.

The following command copies only those MAC files on DK created on or after February 24, 1991:

```
.COPY/SINCE:24:FEB:91 *.MAC DU0:*.MAC
 Files copied:
DK:A.MAC  to DU0:A.MAC
DK:B.MAC  to DU0:B.MAC
DK:C.MAC  to DU0:C.MAC
```

/SLOWLY

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[:value]

Used with the /DEVICE option to specify the starting block and, with /END:value, to specify the last block of the disk you are copying. /START requires the DUP utility.

The /START:value notation must follow the input or output file specification. The *value* argument with both /START and /END represents a decimal block number.

You can use /START:value with the output device specification to specify the starting block number for the write operation on the output volume. The following command copies blocks 500 to 550 of DU0 to DU1, starting at block 100:

```
.COPY/DEVICE DU0:/START:500/END:550  DU1:/START:100
```

If you do not supply a value with /START, RT–11 assumes the first block on the volume. If you do not specify a value with /END, RT–11 assumes the last block on the volume. Note that the first block of a file or volume is block 0.

/SYSTEM

Copies system (SYS) files when you use wildcards in an input file type, or you use the /EXCLUDE option. If you omit this option, the SYS files are excluded from these operations and a message is displayed on the terminal to remind you, unless the SYS file type is specified in the wildcard COPY operation. For example, the following command copies all SYS files since the .SYS file type is included in the wildcard operation:

```
COPY *.SYS DU1:*
```

/TOPS

Transfers files on DECsystem–10 DECtape to RT–11 format. This option requires the FILEX utility.

/TOPS 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.

If you use the /WAIT option with /TOPS, you must use two device drives for the operation.

The following command copies in ASCII format all the files named MODULE from the DECsystem–10 DECtape DT0 to the RT–11 device RK0:

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

/VERIFY

Verifies that the output matches the input after a copy operation between RT–11 directory-structured devices. If the two files or devices are different, a message is displayed on the terminal. This option cannot be used with /ASCII or /BINARY.

/WAIT

Copies from one disk to another if your system has only a single-disk drive, if you want to use only one drive unit of a dual-drive system for a copy operation, or if you wish to remove the system volume in order to use the system drive to hold the input or output volume.

To use this option, enter a command string according to the following general format:

.COPY/WAIT input-filespec output-filespec

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

After you enter this command, RT–11 responds by displaying the following prompt:

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

When the input volume is mounted and you are ready to proceed, type Y RETURN in response to the prompt to continue the COPY operatation.

Type N RETURN or press CTRL/C to abort the COPY operation at the *Continue?* prompt. However, if the system volume is not in place and you try to abort the operation, RT–11 prompts you to remount the system volume. In that case, RT–11 does not abort the operation until you have responded with a Y RETURN to the request for mounting the system volume.

The COPY/WAIT operation varies slightly depending on how many and what drives you use. The following two-drive and one-drive COPY/WAIT steps summarize the COPY/WAIT operations.

A Two-Drive COPY/WAIT Operation

If you are using two drives, the COPY/WAIT operation:

1. Prompts you to mount the input volume in one drive and the output volume in the other drive.
2. Does the COPY operation.

3. If one of the drives you used contained the system volume, prompts you to remount the system volume.

A One-Drive COPY/WAIT Operation

If you are using one drive, the COPY/WAIT operation:

1. Prompts you to mount the input volume and then the output volume and then (if necessary) repeats those steps until the operation has transferrred whatever you have asked it to transfer between the two volumes.
2. If the one drive you used contained the system volume, you are prompted to remount the system volume when the COPY operation is complete.

Restriction

The /WAIT option is valid with /INTERCHANGE, /TOPS, and /DOS when you have two device drives available for the operation, and with /DEVICE when the input and output devices are different.

CREATE

(DUP)

Creates or extends a file with a specific name, location, and size on the random-access volume you specify.

Format

CREATE *filespec*

Filespec Options
{ /EXTENSION /START:value /ALLOCATE:size }

Parameter

filespec

Specifies the file you wish to create or extend. You must specify both the file name and type.

Description

When you create a file with the CREATE command, it creates only a directory entry for the file. This command does not store any data in a file.

If you attempt to create a file over a tentative file (one that was opened but never closed) and the foreground job is loaded, RT–11 prompts you to confirm the operation. If you type Y to continue, the tentative file will be written over. Be sure that you do not write over a tentative file being used by the foreground job, since this will corrupt the file and cause unpredictable results.

Options

/ALLOCATE:size

Allocates the number of blocks you specify for the file you are creating; *size* represents a decimal number of blocks. A value of -1 indicates a file of the maximum size available on the volume. A value of 0 indicates a file of 0 blocks; this command can create a file of 0 blocks. If you do not use /ALLOCATE, RT–11 assumes one block.

/EXTENSION:value

Extends an existing file by the number of blocks you specify; *value* is a decimal number of blocks. 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).

Procedure for Extending a File

The following example illustrates the procedure for extending a file with the CREATE command. In this example, BUILD.MAC is extended by 20 blocks.

1. A DIRECTORY/FULL command determines whether there is available space adjacent to BUILD.MAC:

```
.DIRECTORY/FULL DU1:
 09-FEB-90
MYPROG.MAC    36P 19-JAN-90      TM    .MAC    25  27-JAN-90
VTMAC .MAC     7  19-JAN-90      MYPRO2.MAC    41  19-JAN-90
< UNUSED >    25                 TEST  .MAC    67P 19-JAN-90
LELA  .LBM     1  09-FEB-90      BUILD .MAC    80  19-JAN-90
< UNUSED >   604
 7 Files, 257 Blocks
 629 Free blocks
```

2. The CREATE command extends BUILD.MAC by 20 blocks:

```
.CREATE DU1:BUILD.MAC/EXTENSION:20
```

/START:value

Specifies the starting block number of the file you are creating. The *value* argument specifies a decimal block number. If you do not use /START, RT–11 uses the first available space on the volume.

Procedure for Creating a File with CREATE

The following example illustrates the procedure for creating a file with the CREATE command. In this example, SWAP.SYS is restored after having been deleted.

Note that you can restore a deleted file only if it has not been written over by a new file. So, if you should ever have to restore a deleted file, you should do it before you create any new files.

1. A DIRECTORY/DELETED command establishes the starting block numbers of the deleted files on DU1:

```
.DIRECTORY/DELETED DU1:
 09-FEB-90
SWAP  .SYS  28  19-JAN-90  117  EMPTY.FIL 179  31-JAN-90 315
 0 Files, 0 Blocks
 204 Free blocks
```

2. The CREATE command restores SWAP.SYS, starting at block 117, and the /ALLOCATE:size option allocates 28 blocks:

```
.CREATE DU1:SWAP.SYS/START:117/ALLOCATE:28
```

See the *RT–11 Volume and File Formats Manual* for a detailed description of the RT–11 file structure.

D (Deposit)

(KMON)

Deposits octal values in memory, beginning at the location you specify.

Format

D *address= [value[,...value]]*

Parameter

address

Specifies an octal address that, when added to the relocation base value from the Base command (if you used one), provides the actual address where RT–11 must deposit the value(s). A relocation base of 0 is the default, if no previous Base command executed.

The *value* argument specifies the new contents of the address. If you do not specify a value (that is, you specify *D address=,* RETURN), RT–11 assumes a value of 0. If you specify *D address* RETURN, RT–11 does nothing. If you specify more than one value and separate the values by commas, RT–11 deposits the values in sequential locations, beginning at the location you specify.

The D (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 D 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 subsequent locations, if indicated.

You cannot specify an address that references a location outside the area of the background job unless you use customization patch 31 described in the *RT–11 IND Control Files Manual*.

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.

Examples

1. The D command deposits zeroes into locations 300, 302, 304, and 306:

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

2. The B command sets the base address to 0:

```
.B
```

3. The D command deposits 3705 into location 1000:

   ```
   .D 1000=3705
   ```

4. The B command sets the relocation base to 1000:

   ```
   .B 1000
   ```

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

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

DATE

(KMON)

Sets or displays the current system date.

Format

DATE *[dd-mmm-yy]*

Parameter

dd-mmm-yy

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, 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. It is recommended that you enter the system date as soon as you bootstrap your operating system.

Multi-job monitors automatically increment the date at midnight each day. The SB 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.

Examples

1. This example enters the current date:

```
.DATE 18-MAY-91
```

2. This example displays the current system date:

```
.DATE
18-May-91
```

DEASSIGN

(KMON)

Disassociates a logical device name from a physical device name.

Format

DEASSIGN *[logical-device-name]*

Parameter

logical-device-name

Specifies an alphanumeric name, from one to three characters long and followed by an optional colon, that is assigned to a particular device. Note that spaces and tabs are not permitted in the logical device name.

Examples

1. To remove the assignment of a particular logical device name to a particular 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:
   ```

2. If you specify a logical name that is not currently assigned, RT–11 displays an error message, as this example shows:

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

3. To return your system to having no default device for undefined devices or logical device names, use the following command:

   ```
   .DEASSIGN *
   ```

   See the ASSIGN command description for an explanation of how and when to assign a default device.

4. 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
   ```

5. If DK is assigned to a nonsystem device, the following command disassociates DK from DU1 and restores the default association of DK to SY, the system device:

   ```
   .DEASSIGN DK:
   ```

6. You cannot DEASSIGN BA or SY; for example, the following command produces the indicated error message:

   ```
   .DEASSIGN SY:
   ?KMON-F-Invalid command
   ```

DELETE

(PIP, FILEX, and QUEMAN)

Deletes the files you specify.

Format

DELETE *filespec1[,filespec2,...]*

RT–11 Command Options

```
⎧ ⎡ /BEFORE[:date] ⎤ ⎫
⎪ ⎣ /SINCE[:date]  ⎦ ⎪
⎨  /DATE[:date]      ⎬
⎩  /NEWFILES         ⎭

 /ENTRY
 /EXCLUDE
 /INFORMATION
 /LOG
 /POSITION[:sequence-number]
 /[NO]QUERY
 /SYSTEM
 /WAIT
```

Interchange Command Options

```
⎧  /DOS           ⎫
⎪     /WAIT       ⎪
⎨  /INTERCHANGE   ⎬
⎩     /WAIT       ⎭
```

Parameter

filespec

Specifies the file(s) to be deleted. You can specify up to six file specifications; separate them with commas.

Description

If you accidentally delete a file, you may be able to recover the file if you act immediately. See the CREATE command and the Device Utility Program (DUP) chapter in the *RT–11 System Utilities Manual* for a description of how to do this.

Deleting SYS and BAD Files

RT–11 has a special way of handling system (SYS) files and files that cover bad blocks (BAD files) so that you do not delete these files by accident. RT–11 requires you to use the /SYSTEM option when you need to delete system files and you use wildcards in an input file type. However, if you specify SYS as an input file type, you do not have to specify the /SYSTEM option. To delete a BAD file, you must specify it by explicitly giving its file name and file type. However, since BAD files cover bad blocks on a device, you should not delete or otherwise manipulate these files.

Deleting Protected Files

To delete a protected file (a P next to the block size of a file's directory entry denotes protection), you must first remove protection from that file by using the UNPROTECT command, the COPY/NOPROTECTION command, or the RENAME /NOPROTECTION command.

Confirming a Deletion

Another feature of the DELETE command is that, unless you use /LOG or /NOQUERY, RT–11 requests confirmation from you before it deletes a file, if you use wildcards in the input specification. You must respond to the query message by typing Y RETURN to execute the command.

Specifying a Date with a DELETE Command

Some of the DELETE command options accept a date as an argument. The format for specifying the date is:

[dd][:mmm][:yy]

where:

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

The default value for the date is the current system date. If you omit any of these values (dd, mmm, or yy), the system uses the values from the current system date. For example, if you specify only the year 90 and the current system date is May 4, 1991, the system uses the date 4:MAY:90. If the current 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 at the beginning of each month with the DATE command. If you fail to set the date at the beginning of each month, RT–11 displays -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.)

Options

/BEFORE[:date]

Deletes only those files created before a certain date. If you specify no date the current system date is used.

The following command deletes all SAV files on DU1 that were created before March 20, 1991:

```
.DELETE/LOG/BEFORE:20:MAR:91 DU1:*.SAV
 Files deleted:
DY0:A.SAV
DY0:B.SAV
DY0:C.SAV
```

/DATE[:date]

Deletes only those files with a certain creation date. If no date is specified the current system date is used. The following command deletes all MAC files on DK that were created on February 20, 1991:

```
.DELETE/LOG/DATE:20:FEB:91 DK:*.MAC
 Files deleted:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

/DOS

Deletes 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 option except /WAIT in combination with /DOS.

/ENTRY

Deletes a job from the queue. Use /ENTRY when QUEUE is running as a foreground or system job (see the chapter on the Queue Package in the *RT–11 System Utilities Manual*).

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, RT–11 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. However, this command does not purge a print buffer. So, because of the use of buffering, the printing may continue for a while after QUEUE has stopped the job.

/EXCLUDE

Deletes all the files on a device except the ones you specify. The following command, for example, deletes all files from DU1 except SAV files:

```
.DELETE/EXCLUDE DU1:*.SAV
?PIP-W-No .SYS action
 Files deleted:
DU1:ABC.OLD  ? Y
DU1:AAF.OLD  ? Y
DU1:COMB.  ? Y
DU1:MERGE.OLD  ? Y
```

/INFORMATION

Changes the severity level of the error message that displays when not all of the input files you specified are found. If you do not use /INFORMATION, RT–11 displays an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, RT–11 displays an informational message to tell you which files it cannot find, but execution continues.

In the following example, the input files FILE1.TXT and FILE3.TXT are deleted. However, since RT–11 is unable to find DU1:FILE2.TXT, RT–11 displays a message to inform you:

```
.DELETE/INFORMATION DU1:(FILE1,FILE2,FILE3).TXT
?PIP-I-File not found DU1:FILE2.TXT
```

/INTERCHANGE

Deletes from a diskette a file that is in interchange format. /WAIT is the only option you can use with /INTERCHANGE.

/LOG

Lists on the terminal a log of the files that are deleted by the current command. Note that if you specify /LOG, RT–11 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

Deletes 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 BAK files created today:

```
.DELETE/NEWFILES DU1:*.BAK
Files deleted:
DU1:MERGE.BAK ? Y
```

/POSITION[:sequence-number]

Allows you to move a magtape and perform an operation at the point you specify. Omitting the *sequence-number* argument has the same effect as setting *sequence-number* equal to 0.

The *sequence-number* is interpreted as the decimal sequence-number of files on a magtape. The following table lists the operation initiated by each value of the *sequence-number* specified with the /POSITION:sequence-number option.

Magtape DELETE Operations Specified with the /POSITION:sequence-number Option

Sequence-Number	Operation
0	The magtape rewinds and RT–11 searches for the file you specify. If you specify more than one file, or if you use a wildcard in the file specification, the magtape rewinds before each search.

Sequence-Number	Operation
A positive integer	RT–11 starts from the magtape's present position and searches for the file you specify. If RT–11 does not find the file you specify before it reaches the *sequence-numbered* file from its starting position, it deletes the *sequence-numbered* file. **Note:** Deleting a *sequence-numbered* file also deletes any files that follow it. If the starting position is not the beginning of the tape, it is possible that RT–11 will not find the file you specify, even though it does exist on the tape.
A negative integer	The magtape rewinds; then RT–11 follows the procedure outlined in the preceding delete operation.

/[NO]QUERY

/QUERY requests confirmation before RT–11 deletes each file. This option is particularly useful on operations that involve wildcards, when you may not be completely sure which files RT–11 selected for the operation. This is the default mode of operation when you use wildcards in the file specifications. 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 any string beginning with Y) RETURN to initiate execution of a particular operation. RT–11 interprets any other response as NO and does not perform the operation. Operations are done as positive responses are given to queries.

The following example shows querying. Only the file DU1:AAF.MAC is deleted:

```
.DELETE/QUERY DU1:*.*
 Files deleted:
DU1:ABC.MAC  ? N
DU1:AAF.MAC  ? Y
DU1:MERGE.FOR  ? N
```

Pressing CTRL/C terminates the QUERY operation. For example, if you are searching for a file to delete, as soon as the QUERY procedure displays the file name, you can respond positively to that deletion and then press CTRL/C to terminate the operation.

/NOQUERY suppresses the confirmation message RT–11 displays before it deletes each file.

/SINCE[:date]

Deletes only those files created on or after a certain date. If you specify no date, the current system date is used.

The following command deletes all SAV files on DU1 that were created on or after March 20, 1991:

```
.DELETE/LOG/SINCE:20:MAR:91 DU1:*.SAV
 Files deleted:
DU1:A.SAV
DU1:B.SAV
DU1:C.SAV
```

/SYSTEM

Allows you to delete system (SYS) files and use wildcards in an input file type. If you omit this option while using a wildcard for a file type, system files are excluded from the DELETE operation, and the following message is displayed on the terminal:

```
?PIP-W-No .SYS action
```

If you specify SYS as an input file type, you do not need to specify the /SYSTEM option.

/WAIT

Initiates the DELETE operation but then pauses for you to mount the volume that contains the files you want to delete. This option is useful if you have a single-disk system or if you want to use only one drive unit of a dual-drive system.

When RT–11 pauses, it displays *Mount input volume in <device>; Continue?*, where <device> specifies the device into which you mount the volume. Mount the volume and type Y RETURN. Type N RETURN or press CTRL/C to abort the operation and return control to the monitor.

When the operation completes, RT–11 displays the *Continue?* message again. Mount the system volume and type Y RETURN. RT–11 then displays the monitor prompt.

The following example deletes FILE.MAC from an RL02 disk:

```
.DELETE/WAIT DL0:FILE.MAC
Mount input volume in DL0:; Continue? Y
Dl0:FILE.MAC? Y
Mount system volume in Dl0:; Continue? Y
```

DIBOL

(DIBOL)

Invokes the DIBOL compiler to compile one or more source programs.

Format

DIBOL *filespec[s]*

Command Options

/ALPHABETIZE
/BUFFERING
/CROSSREFERENCE
/[NO]**LINENUMBERS**
  /ALLOCATE:size
/LOG
/[NO]**OBJECT**[:filespec]
  /ALLOCATE:size
/ONDEBUG
/PAGE:value
/TABLES
/[NO]**WARNINGS**

Parameter

filespec[s]

Specifies one or more files to be included in the compilation. If you omit a file type for an input file, RT–11 assumes DBL. Unless you specify otherwise, RT–11 creates an object file with the same name as the first input file and gives it an OBJ file type.

Output default file types are LST for listing files and OBJ for object files.

Compiling Multiple Source Files

- To compile multiple source files *into a single object file*, separate the files with plus (+) signs in the command line. RT–11 creates an object file with the same name as the first input file and gives it an OBJ file type, unless you specify a different name and type.
- To compile multiple source files *in independent compilations*, separate the files with commas (,) in the command line. This generates a corresponding object file for each input file or for each set of input files.

Position Dependence of Language Options

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.

The *DIBOL–11 Language Reference Manual* contains more detailed information about using DIBOL.

Options

/ALLOCATE:size

Used with the /LIST or /OBJECT option 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 65527. A value of -1 is a special case that creates the largest file possible on the device.

/ALPHABETIZE

Alphabetizes entries in the symbol and label tables. This is useful for program maintenance and debugging.

/BUFFERING

Disables double buffering.

/CROSSREFERENCE

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:

- Symbol cross-reference table
- Label cross-reference table
- External subroutine cross-reference table
- COMMON cross-reference table

Note that RT–11 does not generate a listing by default. You must also specify /LIST in the command line to get a cross-reference listing.

/[NO]LINENUMBERS

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

/NOLINENUMBERS 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]

Produces a DIBOL compilation listing. You must specify this option if you want the 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 DIBOL command.

For example, the following command produces a line printer listing after compiling a DIBOL source file:

```
.DIBOL/LIST MYPROG
```

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

```
.DIBOL MYPROG/LIST
```

You can specify a device name with the /LIST option. To do so, use the following format:

```
/LIST:device-name
```

If the device is a file-structured device, RT–11 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 a listing file called A.LST on DU1:

```
.DIBOL/LIST:DU1: 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:DU1:
```

The command shown above compiles A.DBL and B.DBL together, producing files DK:A.OBJ and DU1: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.

/LOG

Creates a log of error messages generated by the compiler.

/[NO]OBJECT[:filespec]

/OBJECT specifies 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 DU1:

```
.DIBOL/OBJECT:DU1: 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 suppresses the 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, RT–11 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

Includes an expanded symbol table in the object file. You can then use a debugging program to find and correct errors in the object file.

/PAGE:value

Overrides the default listing page length of 66 lines. The meaningful range of values for the decimal *value* is 1 to 32768.

/TABLES

Generates a symbol table and label table as part of the listing. This information is useful for program maintenance and debugging. Note that RT–11 does not generate a listing by default. You must also specify /LIST in the command line to produce an assembly listing.

/[NO]WARNINGS

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

/NOWARNINGS suppresses warning messages during compilation.

DIFFERENCES

(BINCOM and SRCCOM)

Compares two files and lists the differences between them.

Format

DIFFERENCES *1st-filespec[s] 2nd-filespec[s]*

Command Options for Comparing Binary Files

```
                ┌ /ALWAYS                 ┐
                │ /BYTES                  │
                │ /DEVICE                 │
    /BINARY     │ /END[:value]            │
                │ /QUIET                  │
                │ /SIPP:filespec          │
                │    [/ALLOCATE:size]     │
                └ /START[:value]          ┘
 ⎧ /OUTPUT:filespec ⎡ /ALLOCATE:size ⎤ ⎫
 ⎨                  ⎣ /ALWAYS        ⎦ ⎬
 ⎪ /PRINTER                            ⎪
 ⎩ /TERMINAL                           ⎭
```

Command Options for Comparing Text Files

```
    /BLANKLINES
    /CASE[:option]
    /CHANGEBAR
    /[NO]COMMENTS
    /FORMFEED
    /MATCH[:value]
 ⎧ /OUTPUT:filespec ⎡ /ALLOCATE:size ⎤ ⎫
 ⎨                  ⎣ /ALWAYS        ⎦ ⎬
 ⎪ /PRINTER                            ⎪
 ⎩ /TERMINAL                           ⎭
    /SLP:filespec ⎡ /ALLOCATE:size ⎤
                  ⎣ /AUDITTRAIL    ⎦

    /[NO]SPACES
    /[NO]TRIM
```

Parameters

1st-filespec[s]

Specifies the first file to be compared.

2nd-filespec[s]

Specifies the second file to be compared.

The default output device for the differences listing is the terminal.

Description

You can send the results of all the comparisons to a file on a volume rather than to the terminal by specifying a file name with the /OUTPUT option.

When comparing text files, the default file type input files is MAC; when comparing binary files, there is no default input file type. The default file type for the differences file is DIF.

Wildcards in Either Input File Specification

You can use wildcards in either input file specification to perform multiple source file and binary file comparisons. When you use wildcards, RT–11 displays which files are being compared before it lists the differences. The DIFFERENCES command allows no implicit wildcards.

A different type of comparison is performed depending on whether you use wildcards in only one or in both of the input file specifications. If you use wildcards in only one of the input file specifications, the system compares the file you specify without any wildcards to all variations of the file specification with wildcards. The wildcard represents the part of the file specification to be varied. You can use this method to compare one file to several other files. For example, when the following command line is executed, RT–11 compares the file TEST1.MAC on device DU0 to all files on device DU1 with the file name TEST2:

```
.DIFFERENCES/MATCH:1/OUTPUT:TEST.DIF DU0:TEST1.MAC DU1:TEST2.*
```

Wildcards in Both Input Specifications

If you use wildcards in both input file specifications, the wildcards represent the part of a file specification you want to be the same in both files being compared. You can use this method to compare several pairs of files; if an input file is compared, it is compared to at most one other input file. For example, when the following command line is executed, the system compares pairs of files; the first input file in each pair has the file name PROG1, and the second has the file name PROG2. The file type of both files in each pair must match:

```
.DIFFERENCES/BINARY DU0:PROG1.* DU1:PROG2.*
```

RT–11 searches for the first file on DU0 with the file name PROG1, and takes note of its file type. Then, RT–11 searches DU1 for a file with the file name PROG2 and the same file type as PROG1. If a match is found, RT–11 compares the two files and lists the differences on the terminal (or sends the differences to an output file if one is specified). RT–11 then searches DU0 for more files with the file name PROG1 and DU1 for PROG2 files with matching file types.

Some Uses of the DIFFERENCES Command

- Comparing two similar versions of a file

  The DIFFERENCES command is 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.

- Creating command files to install patches

  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:filespec and /SIPP:filespec options are designed especially for this purpose. The default file type for the output file created by /SLP is SLP, while the default file type for the output file created by SIPP is COM. You cannot use wildcards when creating SLP or SIPP command files.

Comparing Text Files

During a text comparison, RT–11 compares the two specified files, character for character, and lists any lines that contain differences. By default, the listing is displayed on the terminal screen. You can redirect the listing from the terminal to a file by using the /OUTPUT:*filespec* option, or to the printer by using the /PRINTER option.

Blank lines and the RETURN character are ignored; but all other characters, including line feeds, tabs, and spaces are compared. By default, even differences between uppercase and lowercase characters are listed.

If you compare two files that are identical, RT–11 does not create a listing but displays the following message on the terminal screen:

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

If you compare two files that are different, RT–11 produces a listing of the differences and displays the following message on the terminal screen:

```
?SRCCOM-W-Files are different
```

If you use wildcards in the command line, RT–11 always displays the specifications of the files being compared regardless of whether or not there are differences.

Two Example Files

To understand how to interpret the output listing, first look at the following two sample FORTRAN text files: EXAMP.ONE and EXAMP.TWO. Notice the two differences between the files:

- In line 7, the first file has *go to 10*, while the second file has *go to 100*.

- In line 14, the first file has the variable *radamg*, while the second file has the variable *radang*.

Example 1: FORTRAN File with Errors (EXAMP.ONE)

```
        real function ASIND( x)
        real x
c
c       This FORTRAN callable function returns the ARCSINE
c       of  a specified value as an angle in degrees.
c
        if (ABS( x) .lt. 1.0) go to 10
        ASIND = x * 90.0
        return
c
c       Use trigonometric identity to calculate ARCSINE of X.
c       Then convert radians to degrees.
c
  100   radamg = ATAN( x / SQRT( 1.0-x**2))
        ASIND = radang * 57.29577951
        return
c
        end
```

Example 2: FORTRAN File without Errors (EXAMP.TWO)

```
        real function ASIND( x)
        real x
c
c       This FORTRAN callable function returns the ARCSINE
c       of  a specified value as an angle in degrees.
c
        if (ABS( x) .lt. 1.0) go to 100
        ASIND = x * 90.0
        return
c
c       Use trigonometric identity to calculate ARCSINE of X.
c       Then convert radians to degrees.
c
  100   radang = ATAN( x / SQRT( 1.0-x**2))
        ASIND = radang * 57.29577951
        return
c
        end
```

Comparing the Example Files

See the chapter on the BINCOM utility in the *RT–11 System Utilities Manual* for an example DIFFERENCES listing between two binary files. The following command compares the two text files EXAMP.ONE and EXAMP.TWO and creates the file EXAMP.DIF containing a listing of any differences:

```
.DIFFERENCES/OUTPUT:EXAMP EXAMP.ONE EXAMP.TWO
```

Since the files are different, RT–11 creates the differences file EXAMP.DIF and displays the following message on your screen:

```
?SRCCOM-W-Files are different
```

The following is the contents of the differences file EXAMP.DIF:

```
1) DK:EXAMP.ONE
2) DK:EXAMP.TWO
**********
1)1             if (ABS( x) .lt. 1.0) go to 10
1)              ASIND = x * 90.0
****
2)1             if (ABS( x) .lt. 1.0) go to 100
2)              ASIND = x * 90.0
**********
1)1       100   radamg = ATAN( x / SQRT( 1.0-x**2))
1)              ASIND = radang * 57.29577951
****
2)1       100   radang = ATAN( x / SQRT( 1.0-x**2))
2)              ASIND = radang * 57.29577951
**********)
```

The first two lines identify the two files being compared. Each file name and the device on which the file resides are printed; for example:

```
1) DK:EXAMP.ONE
2) DK:EXAMP.TWO
```

The numbers at the left margin have the form *n)m*, where *n* specifies the files (either 1 or 2) and *m* specifies the page of that file on which the specific line is located. In this case, both files have only one page of output.

RT–11 displays 10 asterisks both before and after a section showing one or more differences between two files. In addition, within each section, a line of four asterisks separates the two files being compared, thus dividing each difference section into two subsections; for example:

```
**********
1)1             if (ABS( x) .lt. 1.0) go to 10
1)              ASIND = x * 90.0
****
2)1             if (ABS( x) .lt. 1.0) go to 100
2)              ASIND = x * 90.0
**********
```

Each difference section ends with a matching line, used as a reference to identify the location of the differing lines; for example:

```
ASIND = x * 90.0
```

and

```
ASIND = radang * 57.29577951
```

Options

/ALLOCATE:size

Used with /OUTPUT, /SLP, or /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 65527. A value of -1 is a special case that creates the largest file possible on the device.

/ALWAYS

Used with /BINARY, /SIPP:filespec, or /OUTPUT:filespec to create 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 DIFFERENCES command, it applies to all output files.

/AUDITTRAIL

Used 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 1st-file so that it matches 2nd-file.

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, RT–11 displays 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

Compares two binary files and lists the differences between them. This option is useful for comparing memory-image 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, RT–11 displays output at the terminal according to the following general format:

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

where:

`bbbbbb` specifies the octal block number of the block that contains the difference.

`ooo`	specifies the octal offset within the block that contains the difference.
`ffffff`	specifies the value in the first file you are comparing.
`ssssss`	specifies the value in the second file you are comparing.
`xxxxxx`	specifies a bit-wise exclusive OR of the two values in the input files.

If you use the /OUTPUT:filespec option with /BINARY, RT–11 stores the DIFFERENCES listing in the file you specify (if there are any differences found), instead of displaying the differences at the terminal.

The DIFFERENCES/BINARY command on magtapes is valid only with tapes having 512-byte blocks.

/BLANKLINES

Includes blank lines in the file comparison. Normally, RT–11 disregards blank lines.

/BYTES

Used with the /BINARY option, lists the differences byte by byte.

/CASE[:option]

Controls case sensitivity in a file comparison. The two options are EXA for case-sensitive comparisons and GEN for case-insensitive comparisons.

/CASE[:EXA], the default, causes a case-sensitive file comparison. The upper- and lowercase of the same character are treated as a difference.

/CASE:GEN causes a case-insensitive file comparison. The upper- and lowercase of the same character are not treated as a difference.

/CHANGEBAR

Creates an output file that contains the 2nd-file with a changebar character next to the lines in the 2nd-file that differ from the 1st-file. RT–11 inserts a vertical bar next to each line that has been added to the 2nd-file, and a bullet (lowercase letter *o*) next to each line that has been deleted.

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

The following example command creates a listing of RTLIB.MAC with a changebar or bullet character at the left margin of each line that is different from RTLIB.BAK:

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

To change the characters SRRCOM uses to indicate insertions and deletions in DIFFERENCES listings, see customization patch 6.1 in the *RT–11 Installation Guide*.

/[NO]COMMENTS

/COMMENTS includes in the file comparison all assembly-language comments found in the two files. (Comments are preceded by a semicolon on the same line.) This is the default operation.

/NOCOMMENTS excludes 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 contents but different formats.

/DEVICE

Used with the /BINARY option to compare two entire volumes starting with block 0. If one input volume is longer than the other, RT–11 displays the message *?BINCOM-W-Device is longer <device>*. RT–11 displays the message *?BINCOM-W-Devices are different* only if differences are found before the point where one input volume ends and the longer one continues.

/END[:value]

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

/FORMFEED

Includes form feeds in the output listing. Normally, RT–11 compares form feeds but does not include them in the output listing.

/MATCH[:value]

Specifies the number of lines from each file that must agree to constitute a match. The *value* argument is an integer in the range 1–200. The default value is 3. This option is invalid with /BINARY.

/OUTPUT:filespec

Specifies a device and file name for the output listing file. Normally, the listing appears on the terminal. If you omit the file type for the listing file, RT–11 uses DIF. Note that RT–11 creates this file only if differences are found. Use the /ALWAYS option if you want RT–11 to create an output file regardless of whether any differences are found.

/PRINTER

Prints a listing of differences on the printer. Normally, the listing appears on the terminal.

/QUIET

Used with the /BINARY option to suppress the display of the differences at the terminal and displays the message *?BINCOM-W-Files are different* or *?BINCOM-W-Devices are different*, if applicable.

/SIPP:filespec

Used with the /BINARY option to produce 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 the 1st-file so it matches the 2nd-file. The default file type for the SIPP file is COM.

The following example 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 the Save Image Patch Program (SIPP)) utility chapter in the *RT–11 System Utilities Manual*.

/SLP[:filespec]

Creates a command file that, when run with the source language patch utility (SLP), patches the 1st-file to match the 2nd-file. The default file type is SLP. If you do not supply a file specification with /SLP, RT–11 displays the command file at the terminal. The following sample creates the command file PATCH.SLP. PATCH.SLP can be used as input to the SLP program to patch RTLIB.BAK so that it matches RTLIB.MAC:

```
.DIFFERENCES/SLP:PATCH RTLIB.BAK RTLIB.MAC
```

To execute the command file you create with /SLP, see the Source Language Patch Program (SLP) chapter in the *RT–11 System Utilities Manual*.

You cannot use wildcards in a command line with the /SLP option.

/[NO]SPACES

/SPACES, the default, includes spaces and tabs in the file comparison. This is useful when you must pay careful attention to spacing in text files.

/NOSPACES excludes spaces and tabs from the file comparison. This is useful when you are comparing two programs with similar contents but different formats.

/START[:value]

Used with the /BINARY option to specify the starting block number of the file comparison, where *value* represents the octal starting block number. With no value specified for /START, the starting number defaults to the first block in the file.

/TERMINAL

Displays the list of differences on the terminal. This is the default operation.

/[NO]TRIM

/TRIM ignores tabs and spaces that appear at the ends of source lines. This is the default setting.

/NOTRIM includes in the comparison spaces and tabs that appear at the ends of source lines.

DIRECTORY

(DIR)

Lists information you request about a device, a file, or a group of files.

The DIRECTORY command runs the DIR utility for RT–11 directories and the FILEX utility for interchange directories.

Format

DIRECTORY *filespec1, filespec2, ...*

Command Options for RT–11 Directories

```
/BEGIN

/BADBLOCKS

[ /END:value     ]
[ /FILES         ]
[ /START:value   ]
[ /WAIT          ]

{ /ALPHABETIZE          }
{    /REVERSE           }
{ /ORDER[:category]     }
{    /REVERSE           }
{ /POSITION             }
{ /SORT[:category]      }
{    /REVERSE           }

{ [ /BEFORE[:date] ]    }
{ [ /SINCE[:date]  ]    }
{ /DATE[:date]          }
{ /NEWFILES             }

/BLOCKS
/BRIEF
/COLUMNS
/DELETED
/EXCLUDE
/FAST
/FREE
/FULL
/OCTAL
/[NO]PROTECTION
/SUMMARY
/VOLUMEID[:ONLY]

{ /OUTPUT:filespec      }
{    /ALLOCATE:size     }
{ /PRINTER              }
{ /TERMINAL             }
```

Command Options for Interchange Directories

```
 /BRIEF
{/DOS                }
{   /OWNER:[nnn,nnn] }
{   /WAIT            }
{/INTERCHANGE        }
{   /WAIT            }
{/TOPS               }
{   /WAIT            }
 /FAST
{/OUTPUT:filespec    }
{   /ALLOCATE:size   }
{/PRINTER            }
{/TERMINAL           }
```

Parameter

filespec[s]

Specifies the device, file, or group of files whose directory information you request. Using commas to separate the file specifications, you can specify up to six file specifications; but using wildcards in the file specifications, you can specify as many files as there are on a disk.

The default parameter (without specifying a file) is all the files on device DK.

Description

The DIRECTORY command lists the information you request about the files stored on a device. For example, you can list the number, names, size, protection status, and creation dates of files stored on a device; and you can list these files in several ways from alphabetical to chronological.

Organization of Directory Listings

Normally, the DIRECTORY command displays 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.

Specifying a Date with a DIRECTORY Option

Some DIRECTORY options accept a date or part of a date as an argument. The format for specifying the date is:

[dd][:mmm][:yy]

where:

dd specifies the day (a decimal integer in the range 1–31).

mmm specifies the first three characters of the name of the month.

yy specifies the year (a decimal integer in the range 73–99).

The default value for the date is the current system date. If you omit any of these values (dd, mmm, or yy), RT–11 uses the values from the current system date. For example, if you specify only the year 90 and the current system date is May 4, 1991, RT–11 uses the date 4:MAY:90. If the current date is not set, RT–11 considers it 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 at the beginning of each month with the DATE command. If you fail to set the date at the beginning of each month, RT–11 displays -*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.)

Options

/ALLOCATE:size

Reserves 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 65527. A value of -1 is a special case that creates the largest file possible on the device.

/ALPHABETIZE

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 volumes (disks and diskettes) have bad blocks, or they develop bad blocks as a result of use and age. Use the /BADBLOCKS option to scan a volume and locate bad blocks on it. RT–11 displays the absolute block number of these blocks on the volumes that return hardware errors when RT–11 tries to read them. This procedure does not destroy data that is already stored on the volume. Remember that block numbers are listed in both octal and decimal, and the first block on a volume is block 0.

If a volume has no bad blocks, an informational message is displayed on the terminal:

```
.DIRECTORY/BADBLOCKS DU1:
?DUP-I-No bad blocks detected DU1:
```

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]

Displays a directory of files created before the date you specify. The following command lists on the terminal all files stored on device DU1 created before February 1991:

```
.DIRECTORY/BEFORE:1:FEB:91 DU1:
  14-Feb-91
MYPROG.MAC   36P 19-Nov-90  TM    .MAC   25  27-Nov-90
BUILD .MAC  100  19-Nov-90
  3 Files, 161 Blocks
  625 Free blocks
```

/BEGIN

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 SRCCOM.SAV on device DU1 and all the files that follow it in the directory:

```
.DIRECTORY DU1:SRCCOM.SAV/BEGIN
  10-Mar-91
SRCCOM.SAV  26P 13-Feb-91   DIR   .SAV  20P 13-Feb-91
SLP   .SAV  13P 13-Feb-91   BINCOM.SAV  25P 13-Jan-91
SIPP  .SAV  21P 13-Feb-91   DUMP  .SAV  10P 13-Jan-91
  6 Files, 115 Blocks
  671 Free blocks
```

/BLOCKS

Displays 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 DU1, including the starting block numbers of files:

```
.DIRECTORY/BLOCKS DU1:
 15-Feb-91
RT11XM.SYS   123P 31-Oct-90    14      SWAP  .SYS    28P 31-Oct-90   137
DU    .SYS    10P 31-Oct-90   165      SL    .SYS    17P 31-Oct-90   175
LD    .SYS    11P 31-Oct-90   192      PIP   .SAV    30P 31-Oct-90   203
DUP   .SAV    52P 31-Oct-90   233      DIR   .SAV    19P 31-Oct-90   285
RESORC.SAV    32P 31-Oct-90   304      MEMO1 .TXT     2  15-Feb-91   336
MEMO3 .TXT     2  15-Feb-91   340
 11 Files, 326 Blocks
 460 Free blocks
```

/BRIEF

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 DU1:
 15-Feb-91
RT11XM.SYS    SWAP  .SYS    DU    .SYS    SL    .SYS    LD    .SYS
PIP   .SAV    DUP   .SAV    DIR   .SAV    RESORC.SAV    MEMO1 .TXT
MEMO3 .TXT
 11 Files, 326 Blocks
 460 Free blocks
```

/COLUMNS:value

Lists a directory in a specific number of columns. The *value* argument specifies an integer in the range 1–9. Normally, RT–11 uses two columns for regular listings and five columns for brief listings. The following example lists the directory information for device DU1 in one column:

```
.DIRECTORY/COLUMNS:1 DU1:
 15-Feb-91
RT11XM.SYS   123P 31-Oct-90
SWAP  .SYS    28P 31-Oct-90
DU    .SYS    10P 31-Oct-90
SL    .SYS    17P 31-Oct-90
LD    .SYS    11P 31-Oct-90
PIP   .SAV    30P 31-Oct-90
DUP   .SAV    52P 31-Oct-90
DIR   .SAV    19P 31-Oct-90
RESORC.SAV    32P 31-Oct-90
MEMO1 .TXT     2  15-Feb-91
MEMO3 .TXT     2  15-Feb-91
 11 Files, 326 Blocks
 460 Free blocks
```

/DATE[:date]

Includes in the directory listing only those files with a certain creation date. The default is the current date. The following command lists all the files on device DU1 that were created on February 15, 1991:

```
.DIRECTORY/DATE:15:FEB:91 DU1:
 15-Feb-91
MEMO1 .TXT     2  15-Feb-91      MEMO3 .TXT     2  15-Feb-91
 2 Files, 4 Blocks
 460 Free blocks
```

/DELETED

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 to rename the area.

The following command lists files on device DU1 that have been deleted:

```
.DIRECTORY/DELETED DU1:
 15-Feb-91
MEMO2 .TXT     2  15-Feb-91   338      MEMO3 .TXT   458  15-Feb-91   342
 0 Files, 0 Blocks
 460 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

Lists the directory of a device that is in RSTS/E or DOS–11 format. The only other options valid with /DOS are /BRIEF, /FAST, /OWNER, and /WAIT. The valid devices are DECtape (RSTS/E and DOS) and RK05 (DOS).

/END:value

Use with /START:value and /BADBLOCKS to specify the last block number of a bad-block scan. If you do not specify /END:value, RT–11 scans to the last block on the volume.

/EXCLUDE

Lists a directory of all the files on a device except those files you specify. The following example lists all files on DU1 except the SAV and SYS files:

```
.DIRECTORY/EXCLUDE DU1:(*.SAV,*.SYS)
 15-Feb-91
MEMO1 .TXT     2  15-Feb-91      MEMO3 .TXT     2  15-Feb-91
 2 Files, 4 Blocks
 460 Free blocks
```

/FAST

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 list the file names of bad blocks. If RT–11 does not find any bad blocks, it displays only the heading, as the following example shows:

```
.DIRECTORY/BADBLOCKS/FILES DU1:
?DUP-I-No bad blocks detected DU1:
```

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.

/FREE

Lists a directory of unused areas and the size of each. This example lists the unused areas on device DK:

```
.DIRECTORY/FREE
 15-Feb-91
< UNUSED >     2                 < UNUSED >   458
 0 Files, 0 Blocks
 460 Free blocks
```

/FULL

Lists the entire directory, including unused areas and their sizes in blocks$_{10}$.

The following example lists the entire directory for device DU1:

```
.DIRECTORY/FULL DU1:
 15-Feb-91
RT11XM.SYS   123P 31-Oct-90      SWAP  .SYS    28P 31-Oct-90
DU    .SYS    10P 31-Oct-90      SL    .SYS    17P 31-Oct-90
LD    .SYS    11P 31-Oct-90      PIP   .SAV    30P 31-Oct-90
DUP   .SAV    52P 31-Oct-90      DIR   .SAV    19P 31-Oct-90
RESORC.SAV    32P 31-Oct-90      MEMO1 .TXT     2  15-Feb-91
< UNUSED >     2                 MEMO3 .TXT     2  15-Feb-91
< UNUSED >   458
 11 Files, 326 Blocks
 460 Free blocks
```

/INTERCHANGE

Lists the directory of a diskette that is in interchange format. The only other options valid with /INTERCHANGE are /BRIEF, /FAST, /VOLUMEID, and /WAIT.

/NEWFILES

Includes in the directory listing only those files 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 created on 15 February 1991:

```
.DIRECTORY/NEWFILES DU1:
 15-Feb-91
MEMO1 .TXT     2  15-Feb-91      MEMO3 .TXT     2  15-Feb-91
 2 Files, 4 Blocks
 460 Free blocks
```

/OCTAL

Lists the sizes (and starting block numbers if you also use /BLOCKS) in octal. If the device you specify is a magtape, RT–11 displays the sequence numbers in octal. The following example shows an octal listing of device DU1:

```
.DIRECTORY/OCTAL DU1:
 15-Feb-91 Octal
RT11XM.SYS    173P 31-Oct-90      SWAP  .SYS     34P 31-Oct-90
DU    .SYS     12P 31-Oct-90      SL    .SYS     21P 31-Oct-90
LD    .SYS     13P 31-Oct-90      PIP   .SAV     36P 31-Oct-90
DUP   .SAV     64P 31-Oct-90      DIR   .SAV     23P 31-Oct-90
RESORC.SAV     40P 31-Oct-90      MEMO1 .TXT      2  15-Feb-91
MEMO3 .TXT      2  15-Feb-91
 11 Files,    506 Blocks
 714 Free blocks
```

/ORDER[:category]

Sorts the directory of a device according to the category you specify. The following table summarizes the categories and their functions.

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). This is the default.
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 DU1, according to each of the categories:

```
.DIRECTORY/ORDER:DATE DU1:
 15-Feb-91
DIR   .SAV    19P 31-Oct-90      RT11XM.SYS   123P 31-Oct-90
DU    .SYS    10P 31-Oct-90      SL    .SYS    17P 31-Oct-90
DUP   .SAV    52P 31-Oct-90      SWAP  .SYS    28P 31-Oct-90
LD    .SYS    11P 31-Oct-90      MEMO1 .TXT     2  15-Feb-91
PIP   .SAV    30P 31-Oct-90      MEMO3 .TXT     2  15-Feb-91
RESORC.SAV    32P 31-Oct-90
 11 Files, 326 Blocks
 460 Free blocks
```

```
.DIRECTORY/ORDER:NAME DU1:
 15-Feb-91
DIR   .SAV    19P 31-Oct-90      PIP   .SAV    30P 31-Oct-90
DU    .SYS    10P 31-Oct-90      RESORC.SAV    32P 31-Oct-90
DUP   .SAV    52P 31-Oct-90      RT11XM.SYS   123P 31-Oct-90
LD    .SYS    11P 31-Oct-90      SL    .SYS    17P 31-Oct-90
MEMO1 .TXT     2  15-Feb-91      SWAP  .SYS    28P 31-Oct-90
MEMO3 .TXT     2  15-Feb-91
 11 Files, 326 Blocks
 460 Free blocks
```

```
.DIRECTORY/ORDER:POSITION DU1:
 15-Feb-91
RT11XM.SYS   123P 31-Oct-90      DUP   .SAV    52P 31-Oct-90
SWAP  .SYS    28P 31-Oct-90      DIR   .SAV    19P 31-Oct-90
DU    .SYS    10P 31-Oct-90      RESORC.SAV    32P 31-Oct-90
SL    .SYS    17P 31-Oct-90      MEMO1 .TXT     2  15-Feb-91
LD    .SYS    11P 31-Oct-90      MEMO3 .TXT     2  15-Feb-91
PIP   .SAV    30P 31-Oct-90
 11 Files, 326 Blocks
 460 Free blocks
```

```
.DIRECTORY/ORDER:SIZE DU1:
 15-Feb-91
MEMO1 .TXT     2  15-Feb-91      SWAP  .SYS    28P 31-Oct-90
MEMO3 .TXT     2  15-Feb-91      PIP   .SAV    30P 31-Oct-90
DU    .SYS    10P 31-Oct-90      RESORC.SAV    32P 31-Oct-90
LD    .SYS    11P 31-Oct-90      DUP   .SAV    52P 31-Oct-90
SL    .SYS    17P 31-Oct-90      RT11XM.SYS   123P 31-Oct-90
DIR   .SAV    19P 31-Oct-90
 11 Files, 326 Blocks
 460 Free blocks
```

```
.DIRECTORY/ORDER:TYPE DU1:
 15-Feb-91
DIR   .SAV    19P 31-Oct-90      RT11XM.SYS   123P 31-Oct-90
DUP   .SAV    52P 31-Oct-90      SL    .SYS    17P 31-Oct-90
PIP   .SAV    30P 31-Oct-90      SWAP  .SYS    28P 31-Oct-90
RESORC.SAV    32P 31-Oct-90      MEMO1 .TXT     2  15-Feb-91
DU    .SYS    10P 31-Oct-90      MEMO3 .TXT     2  15-Feb-91
LD    .SYS    11P 31-Oct-90
 11 Files, 326 Blocks
 460 Free blocks
```

/OUTPUT:filespec

Specifies a device and file name for the output listing file. Normally, the directory listing appears on the terminal. If you omit the file type for the listing file, RT–11 uses DIR.

/OWNER:[nnn,nnn]

Use this option with /DOS to specify a user identification code (UIC). Note that the set of square brackets (immediately surrounding the UIC) are part of the UIC; you must type them.

/POSITION

Lists the file sequence numbers of files stored on a magtape.

/PRINTER

Displays the directory listing on the printer. The default output device is the terminal. Note that the /PRINTER option does use the transparent spooler and will not use the QUEUE program, if available, to queue the directory listing.

/[NO]PROTECTION

/PROTECTION includes in the directory listing only those files on the specified volume that are protected against deletion. The following command lists only those files on DK that are protected:

```
.DIRECTORY/ORDER:SIZE/REVERSE/PROTECTION
 15-Feb-91
RT11XM.SYS   123P 31-Oct-90      DIR   .SAV    19P 31-Oct-90
DUP   .SAV    52P 31-Oct-90      SL    .SYS    17P 31-Oct-90
RESORC.SAV    32P 31-Oct-90      LD    .SYS    11P 31-Oct-90
PIP   .SAV    30P 31-Oct-90      DU    .SYS    10P 31-Oct-90
SWAP  .SYS    28P 31-Oct-90
 9 Files, 322 Blocks
 460 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 PROTECT, RENAME/PROTECTION, and COPY /PROTECTION commands).

/NOPROTECTION includes in the directory listing only those files on the specified volume that are not protected against deletion.

/REVERSE

Lists a directory in the reverse order of the sort you specify with /ALPHABETIZE, /ORDER, or /SORT. The following example sorts the directory of DU1 and lists it in reverse order by size:

```
.DIRECTORY/ORDER:SIZE/REVERSE DU1:
 15-Feb-91
RT11XM.SYS   123P 31-Oct-90      SL    .SYS    17P 31-Oct-90
DUP   .SAV    52P 31-Oct-90      LD    .SYS    11P 31-Oct-90
RESORC.SAV    32P 31-Oct-90      DU    .SYS    10P 31-Oct-90
PIP   .SAV    30P 31-Oct-90      MEMO1 .TXT     2  15-Feb-91
SWAP  .SYS    28P 31-Oct-90      MEMO3 .TXT     2  15-Feb-91
DIR   .SAV    19P 31-Oct-90
 11 Files, 326 Blocks
 460 Free blocks
```

/SINCE[:date]

Lists a directory of all files on a specified volume created on or after a specified date. The following command lists only those files on DK created on or after August 13, 1990:

```
.DIRECTORY/SINCE:13:AUG:90
 15-Feb-91
LETTR1.TXT   123P 31-Oct-90      LETTR2.TXT    28P 22-Nov-90
 2 Files, 151 Blocks
 635 Free blocks
```

/SORT[:category]

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

/START:value

Use this option with the /BADBLOCKS option to specify the starting block, and optionally the last block if you use /END:value, of the bad-block scan. The *value* argument specifies a block number in decimal. If you do not supply a value with /START, RT–11 scans from the first block on the volume. If you do not specify /END:value, RT–11 scans to the end of the volume.

/SUMMARY

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, rather in 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-Mar-91

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

Lists directory information on the terminal. This is the default operation.

/TOPS

Lists the directory of a DECtape that is in DECsystem–10 format. The only other options valid with /TOPS are /BRIEF, /FAST, and /INTERCHANGE.

/VOLUMEID[:ONLY]

/VOLUMEID displays the volume ID and owner name along with the directory listing of the storage volume. If you include the optional argument, ONLY, RT–11 displays only the volume ID and owner name.

You can use /VOLUMEID[:ONLY] with /INTERCHANGE to display the volume identification of the specified interchange diskette.

The following example displays the volume ID of volume DU1:

```
.DIRECTORY/VOLUMEID DU1:
 15-Feb-91
 Volume ID: RT11A
 Owner    : Harry
RT11XM.SYS   123P 31-Oct-90      SWAP  .SYS    28P 31-Oct-90
DU    .SYS    10P 31-Oct-90      SL    .SYS    17P 31-Oct-90
LD    .SYS    11P 31-Oct-90      PIP   .SAV    30P 31-Oct-90
DUP   .SAV    52P 31-Oct-90      DIR   .SAV    19P 31-Oct-90
RESORC.SAV    32P 31-Oct-90      MEMO1 .TXT     2  15-Feb-91
MEMO3 .TXT     2  15-Feb-91
 11 Files, 326 Blocks
 460 Free blocks
```

/WAIT

Use with the /BADBLOCKS option when you want RT–11 to initiate a bad-block scan but first to pause for you to mount the input volume. This option is useful if you have a single-disk system.

When you use this option, and the system volume is mounted, RT–11 initiates the operation you specify, then displays *Mount input volume in <device>; Continue?*. The <device> prompt specifies the device into which you mount the volume.

To continue the operation, mount your input volume and type Y RETURN. Type N RETURN or press CTRL/C to abort the operation and return control to the monitor.

The following sample performs a bad-block scan on an RL02 disk:

```
DIRECTORY/WAIT/BADBLOCKS DL0:
Mount input volume in DL0: Continue? Y
?DUP-I-No bad blocks detected DL0:
Mount system volume in DL0: Continue? Y
```

DISMOUNT

(KMON)

Disassociates a logical-disk unit from a file.

Format

DISMOUNT *logical-disk-unit*

Parameter

logical-disk-unit

Specifies the logical-disk unit that you want to free from its current assignment.

Description

Specify a logical-disk unit number in the form LDn, where n is an integer in the range 0–7, unless you have done a system generation to enable extended device-unit support. If you have extended device-unit support, logical disks have a range L00–L37.

If the logical disk has been assigned a logical device name, you can free the logical-disk unit by specifying the logical device name.

Examples

1. This command frees logical-disk unit 3 (LD3) from its current file assignment:

   ```
   .DISMOUNT LD3:
   ```

2. These two commands show another way of freeing logical-disk unit 3, once it has been assigned the logical device name DAT:

   ```
   .ASSIGN LD3: DAT
   .DISMOUNT DAT
   ```

DUMP

(DUMP)

Displays on the terminal or printer, or writes to a file all or any contiguous part of a file in octal words, octal bytes, ASCII characters, or Radix–50 characters. It is useful for examining directories and files that contain binary data.

Format

DUMP *filespec*

Command Options

/[NO]**ASCII**
/BYTES
/END:value
/FOREIGN
/IGNORE
/ONLY:value

{ /OUTPUT:filespec
   /ALLOCATE:size
/**PRINTER**
/TERMINAL }

/RAD50
/START:value
/**WORDS**

Parameter

filespec

Specifies the device or file you want to examine.

Description

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, RT–11 uses DMP.

NOTE
The DUMP operation does not print data from track 0
of RX01 abd RX02 diskettes.

Block Numbers

- Some options (/ONLY, /START, and /END) accept a block number as an argument.
- All block numbers are in *octal*, and 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.

Magtapes

RT–11 handles operations involving magtape 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, RT–11 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 nonfile-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 RT–11 encounters them on the tape.

HOW TO INTERPRET A *DUMP* LISTING

To understand how DUMP translates binary code, look at the following one-sentence contents of the file FOX.TXT:

```
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
```

The next two example file listings are the first 64 bytes of two different dumps of the preceding file. The 448 bytes of the two dump listings that are not shown list zeros in the rest of the 512 bytes of each file to show that they contain no information—the smallest unit of information RT–11 deals with on a disk is 1 block (512 bytes). When you create a file with KED/KEX, the editor allocates a minimum of 1 block for your file, even if it contains only a few bytes of information.

The first listing is the default listing, without options.

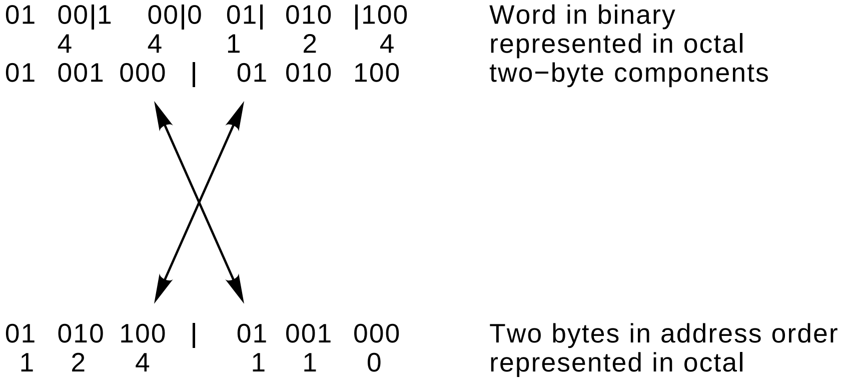
- The first line of the listing contains the input-file specification.
- The second line specifies the input-file block number at which the listing starts.
- The left column of numbers with slashes is the *octal* byte offset from the beginning of the block. Each row across represents 16 bytes or 8 words of binary information; the 17th$_{10}$ byte is at offset 20, the 33rd byte is at offset 40 and so forth.
- The eight columns following the byte offsets contain eight words in octal code.
- The ASCII equivalent of the eight words is displayed in the column to the right of the octal words.

First Listing

```
FOX.TXT
BLOCK NUMBER  000000
000/ 044124 020105 052521 041511 020113 051102 053517 020116 *THE QUICK BROWN *
020/ 047506 020130 052512 050115 042105 047440 042526 020122 *FOX JUMPED OVER *
040/ 044124 020105 040514 054532 042040 043517 000056 000000 *THE LAZY DOG....*
060/ 000000 000000 000000 000000 000000 000000 000000 000000 *................*
```

The second listing includes the two options /WORDS (specifying octal words) and /BYTES (specifying octal bytes). If you do not include the /WORDS option along with the /BYTES option, the listing will not contain words in octal code.

Note also the relationship of the bytes to the words. For example, the first octal word is 044124. That word is divided into a left byte represented by the octal number 124 and a right byte represented by the octal number 110. However, the bytes are displayed in address order; the low-order byte of each word is displayed before the high-order byte. See the following diagram:



The *RT–11 Quick Reference Manual* has a reference section table listing the left/right byte equivalents for each of the octal numbers from 000 to 377.

The ASCII equivalent of each byte is placed below that byte.

Note the dots in the listing. DUMP uses a dot to represent not only a period but also nonprinting codes, such as those for control characters.

Second Listing

```
DK:FOX.TXT
BLOCK NUMBER  000000
000/ 044124  020105  052521  041511  020113  051102  053517  020116
     124 110 105 040 121 125 111 103 113 040 102 122 117 127 116 040
     T   H   E       Q   U   I   C   K       B   R   O   W   N
020/ 047506  020130  052512  050115  042105  047440  042526  020122
     106 117 130 040 112 125 115 120 105 104 040 117 126 105 122 040
     F   O   X       J   U   M   P   E   D       O   V   E   R
040/ 044124  020105  040514  054532  042040  043517  000056  000000
     124 110 105 040 114 101 132 131 040 104 117 107 056 000 000 000
     T   H   E       L   A   Z   Y       D   O   G   .   .   .   .
060/ 000000  000000  000000  000000  000000  000000  000000  000000
     000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
```

HOW TO INTERPRET A DUMP OF A DIRECTORY

One reason for examining volumes is to check the information stored in directories. To understand how to interpret a dump listing of a directory, note the following directory of an RX50 diskette:

```
 13-Feb-91
MEMO1 .TXT     1  13-Feb-91      MEMO2 .TXT     6  13-Feb-91
 2 Files, 7 Blocks
 779 Free blocks
```

The preceding directory listing contains two files. If you examine that directory with the command DUMP/NOASCII/RAD50/ONLY:6, you get the following directory listing. Note the input file specification at the start of directory dump listing DU:/N /X/O:6:

DU: Is the device containing the volume with the directory to be examined.

/N Specifies that ASCII output be suppressed. Since ASCII binary code is not used to store information in RT–11 directories, ASCII translations of directory information would produce useless information.

/X Specifies Radix–50 output since RT–11 uses Radix–50 code to store information in directories. This is a code that is more compact than ASCII and can store three characters in a binary word (rather than two). In the following listing, the letters and numbers beneath the octal words are the Radix–50 equivalents of those words. If you look carefully at the Radix–50 equivalents, you can see (in groups of two and three alphanumeric characters) the names of the files listed in the preceding directory.

/O:6 Specifies the listing contain only the information in block 6. RT–11 directories on random-access devices always begin in block 6. So, if you want a dump of a directory on a random-access device, begin with block 6; that is, specify /S:6 (for *start at block 6*).

In the following example, the option /O (for /ONLY) is the letter *O*. And only a listing of block 6 is requested.

Note that only the first 96 bytes of the 512-byte block of the dump listing are shown in the following example. Since the listing is of a directory containing only two files, the rest of the listing is of unused bytes.

Sample Directory Listing

```
DZ:/N/X/O:6
BLOCK NUMBER  000006
000/ 000004 000000 000001 000000 000016 002000 051025 061230
       D             A             N     YX    MEM    O1
020/ 100324 000001 000000 004663 002000 051025 061300 100324
     TXT      A          AVC     YX    MEM    O2     TXT
040/ 000006 000000 004663 001000 000325 063471 023364 001413
       F           AVC     L2     EM    PTY    FIL     SS
060/ 000000 004663 004000 000000 000000 000000 000000 000000
            AVC    AKH
100/ 000000 000000 000000 000000 000000 000000 000000 000000

120/ 000000 000000 000000 000000 000000 000000 000000 000000
```

See the *RT–11 System Utilities Manual* for more examples of using DUMP.

Options

/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 65527. A value of -1 is a special case that creates the largest file possible on the device.

/[NO]ASCII

/ASCII displays 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 suppresses 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

Displays information in octal bytes. When you specify /BYTES, RT–11 does not display words unless you also use /WORDS.

/END:value

Specifies an ending block number for the dump. RT–11 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

Dumps a magtape that is not RT–11 file-structured.

/IGNORE

Ignores 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:value

Dumps only the block you specify.

/OUTPUT:filespec

Specifies a device and file name for the output listing file. Normally, the listing appears on the printer. If you omit the file type for the listing file, RT–11 uses DMP.

/PRINTER

Displays the output listing on the printer. This is the default operation.

/RAD50

Displays the Radix–50 equivalent of each octal word that is dumped.

/START:value

Specifies a starting block number for the dump. RT–11 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

Displays the output listing on the terminal. Normally, the listing appears on the printer.

/WORDS

Displays information in octal words. This is the default operation.

E (Examine)

(KMON)

Displays in octal the contents of an address on the terminal.

Format

E *address1[-address2]*

Parameter

address

Specifies an octal address that, when added to the relocation base value from the B (Base) command, provides the actual address that RT–11 examines.

Description

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 E (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, RT–11 decreases it by one.

If you specify more than one address (in the form address1-address2), RT–11 displays 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, RT–11 displays the contents of relative location 0.

Note that you cannot examine addresses outside the background. However, you can use the Examine and Deposit commands above a background job if you apply customization patch 31 in the *RT–11 Installation Guide*.

Examples

1. This command displays the contents of location 1000, assuming the relocation base is 0:

   ```
   .E 1000
   127401
   ```

2. This command sets the relocation base to 1000:

   ```
   .B 1000
   ```

3. This command displays the contents of locations 2000 (offset of 1000 from last B command) through 2005:

   ```
   .E 1001-1005
   127401 007624 127400
   ```

EDIT

(EDIT, KED, KEX, TECO)

Runs a text editor for creating or modifying ASCII files.

Format

EDIT *filespec*

```
Command Options                                                  Filespec Options

{ /EDIT                                                      }   /ALLOCATE:size
{                    [ /[NO]COMMAND[:init-filespec] ]        }
{ { /KED }           [ /JOURNAL[:jour-filespec]     ]        }
{ { /KEX }           [ /[NO]QUERY                   ]        }
{                    [ /RECOVER[:jour-filespec]     ]        }
{ /TECO                                                      }
{    /EXECUTE:filespec                                       }
{ /CREATE                    }
{ /INSPECT (or /READONLY)    }
{ /OUTPUT:filespec           }
{    /ALLOCATE:size          }
```

Parameter

filespec

The text (ASCII) file you want to create or edit.

Description

You can invoke any one of the following four editors with the EDIT command:

EDIT
KED
KEX
TECO

EDIT is a **line** editor, used for hard-copy terminals. For a description of EDIT, see the EDIT chapter in the *RT–11 System Utilities Manual*.

KED and KEX are keypad (**screen**) editors. These two editors, derived from the same source files, have the same functionality; they differ in that KED is designed to be used with an unmapped monitor, while KEX is designed to be used with a mapped monitor. For a complete description, see the *PDP–11 Keypad Editor User's Guide*.

TECO is not distributed or supported by Digital; many of the options described here do not apply to the TECO editor.

Default Editor

KED is the default editor for unmapped monitors, and KEX is the default editor for mapped monitors. However, you can use the SET EDIT command to set a default editor (EDIT, KED, KEX) so that when you issue the EDIT command, you invoke that editor (see SET EDIT).

File Backup

When you edit a preexisting file, the editor saves the original file by changing its file type to BAK (for backup), when you exit from the editor.

Protected Files

To perform any edit operations that modify a protected file, you must UNPROTECT that file. (See the UNPROTECT command, the COPY/NOPROTECTION command, or the RENAME/NOPROTECTION command.)

Some Major Features of KED/KEX

- Screen editing with all the standard features of a screen editor. This means you can see at one time as many lines in a file as will fit on a screen, adjust the line width of your text, quickly search for information through an entire file, and so on.
- An initialization-file capability, allowing you to preset some functions of the editor.
- A journal-file option, allowing you to recover your work in the event of an unexpected power-down.
- An ability to create macros allowing you to make your own editor commands.

For a full description of KED/KEX see the *PDP–11 Keypad Editor User's Guide*.

Running KEX as a System Job

Using the SRUN command, you can run KEX as a system job on a mapped monitor. Doing so allows you to keep a file open for editing while doing other work at your terminal. The system job feature also allows you to edit more than one file at a time (depending on how much available memory your computer has). See the system job feature chapter in the *Introduction to RT–11*.

Options

/ALLOCATE:size

Used with /OUTPUT or after the file specification, reserves space on the device for the output file. This option is position dependent; it qualifies a file-specification.

The argument size represents the number of blocks of space to allocate. The meaningful range for this value is from -1 to 65527. A value of -1 is a special case that creates the largest file possible on the device. The following command creates a file on device DU1 and allocates all the space available on the disk for that file:

```
.EDIT/CREATE DU1:PROG1.MAC/ALLOCATE:-1
```

/[NO]COMMAND[:init-filespec]

(KED/KEX only) /COMMAND, the default, executes the initialization file KEDINI.KED or the specified initialization file.

By default, KED/KEX looks for the initialization file KEDINI.KED on DK and, if not found, on SY, before beginning a work session. If found, KED/KEX executes KEDINI.KED. If it is not found, KED/KEX does not report an error. KEDINI.KED is not distributed; you create KEDINI.KED if you want a default KED/KEX initialization file.

/NOCOMMAND suppresses initialization file processing.

/CREATE

Creates a new file when the specified input file is not found.

The following example creates a file called NEWFIL.TXT on device DK:

```
.EDIT/CREATE NEWFIL.TXT
```

/EDIT

Invokes the EDIT line editor. Use this option for editing files on hard-copy terminals.

/EXECUTE:filespec

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

/INSPECT

Opens a file with read-only access. All edit operations are allowed except any that would modify the file:

```
.EDIT/INSPECT NEWFIL.TXT
```

/JOURNAL[:jour-filespec]

(KED/KEX only) Produces a journal file that will let you recover your work after a system crash or similar malfunction. The default file name of the journal file is the name of your output file with a file type of JOU. However, you can specify a name and file type for the journal file; if you specify only a name, the file has the default file type of JOU. The following example command creates the journal file PROG1.JOU:

```
.EDIT/JOURNAL PROG1.FOR
```

Once you create a journal file, if your work session is terminated by an error or malfunction, you can use the /RECOVER option to recover all or most of your work.

The default device for a journal file is the DK device or a device you specify on the command line. See customization patch 65 in the *RT–11 Installation Guide* to change the default device for the journal file.

Note:

- If you create a journal file and you successfully save your editing session, the journal file is automatically deleted when you exit the editor.
- RT–11 renames a superseded journal file by giving it a JBK file type (the journal-file equivalent of a BAK file type). This could happen if you create a journal file during a recovery operation and your work session is again terminated by some error or malfunction. In this case, the second journal file, which might have recorded new work in addition to recovering the old work, takes on the JOU file type while the first journal file takes on the JBK file type.

/KED

Invokes the KED keypad editor, the default editor for unmapped monitors:

```
.EDIT/KED PROG1.FOR
```

/KEX

Invokes the KEX keypad editor, the default editor for mapped monitors:

```
.EDIT/KEX PROG1.FOR
```

/OUTPUT:filespec

Specifies a new file to contain the output of your work session, leaving the input file unchanged:

```
.EDIT/OUTPUT:PROG2.FOR PROG1.FOR
```

/[NO]QUERY

/QUERY, the default, requests for confirmation before carrying out some commands, such as creating a new file.

/NOQUERY suppresses any queries that the editor can generate and executes your command without waiting for confirmation. This includes queries generated by commands in an initalization file or any queries the editor usually makes when you create or edit a file:

```
.EDIT/NOQUERY MEMO20.TXT
```

/READONLY

Opens a file as read only. This is the same as the /INSPECT option.

/RECOVER[:jour-filespec]

Produces a text file containing the work in your editing session that was terminated without an EXIT or QUIT command.

If you do not specify a journal file for the /RECOVER command to use, RT–11 looks for the default journal-file specification of the file you specify in your EDIT command. That is, RT–11 looks for a file having a JOU file type and the same name as the file you specify with the EDIT command. For example, in executing the following command, RT–11 looks for the journal file MEMO20.JOU:

```
.EDIT/RECOVER MEMO20.TXT
```

You can use the /RECOVER option with the /JOURNAL option to both recover work from an editing session and to produce another journal file. This second journal file contains both your recovered work and any additional work you might do. The following command recovers the file MEMO20.TXT by using the journal file MYMEMO.REC. This command also saves a record of the action in a new journal file called MEMO20.JOU:

```
.EDIT/JOURNAL/RECOVER:MYMEMO.REC MEMO20.TXT
```

/TECO

Invokes the TECO editor. (TECO is not distributed or supported by Digital.)

EXECUTE

(DIBOL, CREF, F77, F77XM, FORTRA, LINK, and MACRO)

Invokes one or more language processors, compiles or assembles source files, and then links and runs the resulting modules.

Format

EXECUTE *filespec[s]*

Command Options	Filespec Options
/BOTTOM:value /DEBUG[:filespec] /DUPLICATE /EXECUTE[:filespec] /ALLOCATE:size /GLOBAL /LINKLIBRARY:filespec /LIST[:filespec] /ALLOCATE:size /MAP[:filespec] /ALLOCATE:size /WIDE /OBJECT[:filespec] /ALLOCATE:size /PROMPT /[NO]**RUN**	[/LIBRARY[:filespec]]

/DIBOL [
/ALPHABETIZE
/BUFFERING
/CROSSREFERENCE
/[NO]**LINENUMBERS**
/LOG
/ONDEBUG
/PAGE:value
/TABLES
/[NO]**WARNINGS**]

/FORTRAN [
/CHECK
/CODE:type
/CONTINUATIONS:value
/DIAGNOSE
/EXTEND
/**F4** or /F77
/HEADER
/I4
/[NO]**LINENUMBERS**
/ONDEBUG
/[NO]**OPTIMIZE**
/RECORD:length
/SHOW[:TYPE]
/STATISTICS
/[NO]**SWAP**
/TRACE:type
/UNITS:value
/[NO]**VECTORS**
/[**NO**]**WARNINGS**
/WIDE
/WORKFILE]

Command Options	Filespec Options
{ /MACRO [/CROSSREFERENCE[:type[:type...]] /DISABLE:type[:type...] /ENABLE:type[:type...] /[NO]SHOW:type[:type...]] }	

Parameter

filespec[s]

Specifies one or more files to be included in the assembly or compilation.

Compiling Multiple Source Files

You can compile up to six source files into a single object file or into multiple object files.

- To compile multiple source files *into a single object file*, separate the files with plus (+) signs in the command line. RT–11 creates an object file with the same name as the first input file and gives it an OBJ file type, unless you specify a different name and type.
- To compile multiple source files *in independent compilations*, separate the files with commas (,) in the command line. This generates a corresponding object file for each input file or for each set of input files.

Description

The EXECUTE command runs MACRO, FORTRA, F77, F77XM, DIBOL, LINK, and CREF.

The following are the default input and output file types.

Default File Types

File Type	Description
DBL	DIBOL source file
FOR	FORTRAN source file
LST	Listing file
MAC	MACRO source file
MLB	MACRO macro-library file
OBJ	Object file

Choosing a Language Processor

You can specify a language processor with the EXECUTE command in any of the following three ways:

- Specify a language-name option, such as /MACRO, which invokes the MACRO assembler. Since there are two FORTRAN compilers, see the next section for how to override the default FORTRAN compiler.
- 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.
- 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), RT–11 issues an error message.

Choosing the FORTRAN IV or the FORTRAN–77 Compiler

The meaning of the /FORTRAN option can be changed, depending on the condition set by the SET FORTRA command. By default, or if SET FORTRA F4 has been issued, the EXECUTE/FORTRAN command calls the FORTRAN IV compiler. If SET FORTRA F77 has been issued, the EXECUTE/FORTRAN command calls the FORTRAN–77 compiler. The FORTRAN–77 compiler under a mapped monitor is F77XM.SAV and under an unmapped monitor is F77.SAV.

The current condition at which FORTRAN is set can be overridden, using the /F4 option to specify FORTRAN IV or /F77 to specify FORTRAN–77. Options supported only under FORTRAN–77 are indicated by *FORTRAN–77 only*. Also, some options, such as /SHOW:type, take different arguments or are otherwise different than that option under FORTRAN IV. Therefore, before using the FORTRAN options you should see the FORTRAN command description.

Position Dependence of Language Options

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.

Options

/ALLOCATE:size

Used with /EXECUTE, /LIST, /MAP, or /OBJECT, reserves 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 65527. A value of -1 is a special case that creates the largest file possible on the device.

/ALPHABETIZE

(DIBOL only) Alphabetizes the entries in the symbol table listing. This is useful for program maintenance and debugging.

/BOTTOM:value

Specifies the lowest address to be used by the relocatable code in the load module. The *value* argument specifies a six-digit, unsigned, even octal number. If you do not use this option, RT–11 positions the load module so that the lowest address is location 1000_8. This option is invalid for foreground links.

/Buffering

(Dibol only) Disables double buffering.

/CHECK

(FORTRAN–77 only) Checks array references to ensure they are within the specified array address boundaries. The FORTRAN IV functionality of checking is done by the selection of OTS library components, not by compiler action.

/CODE:type

(FORTRAN IV only) Produces 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 valid 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 functions.

/CONTINUATIONS:value

(FORTRAN–77 only) Specifies the maximum decimal number of continuation lines in the program. The valid range for *value* is 0 through 99; the default is 19.

/CROSSREFERENCE[:type[:type...]]

(DIBOL and MACRO) Generates 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 optional arguments. The argument type represents a one-character code that indicates which sections of the cross-reference listing the assembler should include. See the description of the /CROSSREFER-ENCE option under the MACRO command for a summary of the valid arguments and their meaning.

/DEBUG[:filespec]

Links ODT (On-Line Debugging Technique, described in the *RT–11 System Utilities Manual*) 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

(/FORTRAN IV only) Helps 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

Invokes the DIBOL language processor to compile the associated files.

/DISABLE:type[:type...]

(MACRO only) Specifies arguments for a .DSABL directive. See the description of the /DISABLE option under the MACRO command for a summary listing of the *type* arguments and their meanings. See the *PDP–11 MACRO–11 Language Reference Manual* for a description of the directive and a list of all valid types.

/DUPLICATE

Places duplicate copies of a library module in each overlay segment that references the module. This option is useful in reducing the size of the root segment of your program. When you have entered the complete EXECUTE command, RT–11 prompts you for the names of the global symbols in the library module you want to duplicate. The prompt is:

```
Duplicate symbol?
```

Respond by typing the name of each global symbol you want to duplicate. Terminate each response with RETURN. Press RETURN after the last global symbol you want to duplicate.

See the LINK utility in the *RT–11 System Utilities Manual* for more information on duplicating library modules.

/ENABLE:type[:type...]

(MACRO only) Specifies arguments for an .ENABL directive. See the description of /DISABLE option under the MACRO command for a summary of the *type* arguments and their meanings. See the *PDP–11 MACRO–11 Language Reference Manual* for a description of the directive and a list of all valid types.

/EXECUTE[:filespec]

Specifies a file name or device for the executable file. Note that any time you type a colon after the /EXECUTE option (/EXECUTE:), you must specify a device or a file specification after the colon.

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 depending on whether it follows the command or the file specification. The following command creates an executable file called PROG1.SAV on device DU1:

```
.EXECUTE/EXECUTE:DU1:  PROG1,PROG2
```

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

```
.EXECUTE RTN1,RTN2,MYPROG/EXECUTE
```

/EXTEND

(FORTRAN IV or FORTRAN–77)

- Used with /FORTRAN IV, changes the right margin for source input lines from column 72 to column 80.
- Used with /FORTRAN–77, changes the right margin for source input lines from column 72 to column 132.

/F4

Overrides any SET FORTRA conditional and calls the FORTRAN IV compiler (the default).

/F77

Overrides any SET FORTRA conditional and calls the appropriate FORTRAN–77 compiler.

/FORTRAN

Invokes the FORTRAN language processor to compile the associated files.

By default, or if SET FORTRA F4 has been issued, the COMPILE/FORTRAN command calls the FORTRAN IV compiler. If SET FORTRA F77 has been issued, the COMPILE/FORTRAN command calls the FORTRAN–77 compiler. The FORTRAN–77 compiler under a mapped monitor is F77XM.SAV and under an unmapped monitor is F77.SAV. The FORTRAN IV compiler is FORTRA.SAV. See the FORTRAN command for more information.

/GLOBAL

Generates a global symbol cross-reference section in the load map. The global symbols are listed alphabetically. Each module in which a symbol is referenced or defined is listed in alphabetical order after the global symbol. A number sign (#) after a module name indicates that the global symbol is defined in that module. A plus sign (+) after a module name indicates that the module is from a library.

See the description of the LINK utility in the *RT–11 System Utilities Manual* for a load-map example that includes a global symbol cross-reference table, and for a more detailed description of how to interpret a load map.

Note that RT–11 does not generate a load map by default. You must also specify /MAP in the command line to get a cross-reference section. The following command produces a map listing file, MYPROG.MAP, that contains a global symbol cross-reference section:

```
.EXECUTE/GLOBAL/MAP:DU1:  MYPROG
```

/HEADER

(/FORTRAN IV only) Includes in the printout a list of options currently in effect.

/I4

(FORTRAN IV or FORTRAN–77) Allocates two words for the default integer data type (FORTRAN IV uses only one-word integers) so that it takes the same physical space as real variables. With FORTRAN–77, /I4 allocates and uses two words for the default integer and logical variable length.

/LIBRARY

(MACRO only) Identifies 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 following example, 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.MLB and SYSMAC.SML. It produces C.OBJ and C.LST. RT–11 then links B.OBJ and C.OBJ, resolving undefined references from SYSLIB.OBJ, and produces the executable file B.SAV. Finally, RT–11 loads and executes B.SAV:

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

/[NO]LINENUMBERS

(DIBOL, FORTRAN IV, or FORTRAN–77) /LINENUMBERS includes internal sequence numbers in the executable program. These are useful in debugging programs. This is the default operation.

/NOLINENUMBERS 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 DIBOL or FORTRAN error messages are difficult to interpret.

/LINKLIBRARY:filespec

Includes 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]

Produces a compilation or assembly listing. This is not the default; you must specify this option to get a listing.

The /LIST option has different meanings depending on how you specify it:

- To generate a listing on the printer, specify /LIST without *filespec* as an option of the EXECUTE command.
- To store a listing on a file-structured device, specify /LIST with a device name. RT–11 stores the listing file on that device, assigning it the same name as the input file with a LST file type. For example, the following command creates a listing file called A.LST on DU1:

  ```
  .EXECUTE/LIST:DU1: A.MAC
  ```

- To display a listing on the terminal, specify /LIST with TT; for example:

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

- To override the default file type of LST, specify the /LIST option with a name and file type. For example, the following command compiles A.FOR and B.FOR together, producing files A.OBJ and AANDB.OUT on device DK. The command then links A.OBJ (using SYSLIB.OBJ as needed) and produces A.SAV:

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

- To produce a listing file of the same name as the input file, specify /LIST after the file specification to which it applies. For example, the following command first compiles A.DBL and B.DBL together, producing files DK:A.OBJ and DU1:B.LST. The command then links A.OBJ (using SYSLIB.OBJ as needed) and produces DK:A.SAV:

  ```
  .EXECUTE/DIBOL A+B/LIST:DU1:
  ```

Specifying a file name with a /LIST option following a file specification has the same meaning as specifying a file name with the /LIST option following the command. For example, 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. For example, the following command compiles A.MAC and produces A.OBJ and A.LST:

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

This command also compiles B.FOR, producing B.OBJ. However, it does not produce any listing file for the compilation of B.FOR. After compiling A.MAC and B.FOR, RT–11 links A.OBJ and B.OBJ together, producing A.SAV.

/LOG

(DIBOL only) Creates a log of error messages generated by the compiler.

/MACRO

Invokes the MACRO–11 assembler to assemble associated files.

/MAP[:filespec]

Produces a load map after a link operation. This is not the default; you must specify this option if you want a load map.

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]

Specifies a file name or device for the object file.

Because the EXECUTE command creates object files by default, the following three commands have the same meaning:

```
.EXECUTE/FORTRAN A

.EXECUTE/FORTRAN/OBJECT A

.EXECUTE/FORTRAN A/OBJECT
```

These commands compile A.FOR and produce A.OBJ as output. The /OBJECT option functions like the /LIST option; it can be either a command option 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 DU1:

```
.EXECUTE/OBJECT:DU1: A.MAC,B.MAC
```

Use /OBJECT as a file qualifier 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

(DIBOL, FORTRAN IV, or FORTRAN–77)

- Use /ONDEBUG with /DIBOL to include an expanded 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.

/[NO]OPTIMIZE

(FORTRAN–77 only) /OPTIMIZE enables compiler optimization (the default).

/NOOPTIMIZE inhibits compiler optimization.

/PAGE:value

(DIBOL only) Overrides the default listing page length of 66 lines. The meaningful range of values for the *value* argument is 1 to 32768_{10}.

/PROMPT

Enters additional lines of input for the link operation. RT–11 continues to accept lines of linker input until you enter two slashes (//). See the LINK utility chapter in the *RT–11 System Utilities Manual* for all 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 (Command String Interpreter) conventions, as described in the *RT–11 System Utilities Manual*)

The example that follows uses the /PROMPT option to create an overlay structure for the program COSINE.MAC:

```
.EXECUTE/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 command file. The file LCP.COM contains these lines:

```
A/PROMPT
SUB1/O:1
SUB2/O:1
SUB3,SUB4/O:1
//
```

The following command, using the preceding file, produces an executable file, DK:A.SAV, and a link map on the printer:

```
.EXECUTE/MAP  @LCP
```

/RECORD:length

(FORTRAN IV or FORTRAN–77) Overrides the default maximum record length for sequentially formatted input and output. The meaningful range for *length* is from 4 to 4095.

For FORTRAN IV, the default maximum record length is usually 132 characters, while for FORTRAN–77, the default length is usually 136.

/[NO]RUN

/RUN initiates 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 suppresses execution of your program. The system performs only the compilation and the link.

/[NO]SHOW:type

(FORTRAN IV, FORTRAN–77, or MACRO)

- /SHOW:type, used with /FORTRAN, controls the FORTRAN listing format. The *type* argument specifies a code that indicates which listings the compiler is to produce. See the /SHOW option description under the FORTRAN command for a table listing the FORTRAN *type* arguments.
- /SHOW:type, used with /MACRO, specifies one or more MACRO .LIST directives. See the /SHOW option description under the MACRO command for a table listing the MACRO *type* arguments. The *PDP–11 MACRO–11 Language Reference Manual* explains how to use these directives.

/NOSHOW:type, used with MACRO, specifies any MACRO .NLIST directive. See the /SHOW option description under the MACRO command for a table listing the MACRO .NLIST directives. The *PDP–11 MACRO–11 Language Reference Manual* explains how to use these directives.

/STATISTICS

(FORTRAN IV or FORTRAN–77) Used with /FORTRAN, includes compilation statistics in the listing, such as amount of memory used, amount of time elapsed, and length of the symbol table.

With FORTRAN–77, /STATISTICS includes in the compilation statistics workfile access statistics and the number of pages of dynamic memory available and used.

/[NO]SWAP

(FORTRAN IV or FORTRAN–77) /SWAP permits the USR (User Service Routine) to swap over the FORTRAN program in memory. This is the default operation.

/NOSWAP 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 calls (see the *RT–11 System Subroutine Library Manual*). If the program frequently updates or creates a large number of files, making the USR resident can improve program execution. However, the cost for making the USR resident is 2K words of memory.

/TABLES

(DIBOL only) Generates a symbol table and label table as part of the assembly 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 produce an assembly listing.

/TRACE:type

(FORTRAN–77 only) Enables the FORTRAN–77 /S:xxx option. This controls the generation of code used by the OTS during error traceback. See the /TRACE option description of the FORTRAN command for the *type* arguments.

/UNITS:value

(FORTRAN IV or FORTRAN–77) Overrides the default number of logical units (6) to be open at one time. The *value* argument is a decimal number. With FORTRAN IV, the maximum value you can specify is 16. With FORTRAN–77, the maximum is 99.

/[NO]VECTORS

(FORTRAN IV only) /VECTORS directs the FORTRAN IV compiler to use tables to access multidimensional arrays. This is the default mode of operation.

/NOVECTORS directs the FORTRAN IV compiler to use multiplication operations to access multidimensional arrays.

/[NO]WARNINGS

(DIBOL, FORTRAN IV, or FORTRAN–77) /WARNINGS includes warning messages in 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 suppresses 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

(FORTRAN–77 only) Produces a 132-column compiler listing.

/WORKFILE:value

(FORTRAN–77 only) Sets the workfile size. The default is 128_{10} blocks. By default, *value* is octal; include a decimal point after *value* to specify a decimal radix.

FORMAT

(FORMAT)

- Formats some disks and diskettes by writing headers to each block in a volume.
- Verifies that blocks on disks and diskettes can be read from or written to.
- Replaces bad blocks (ones from/to which you cannot read/write) on a device with good blocks.

Format

FORMAT *device:*

Command Options

/[NO]**QUERY**
/SINGLEDENSITY
/VERIFY[:ONLY]
   /PATTERN:value
/WAIT

Parameter

device:

Specifies the storage volume you want to format and/or verify.

Description

You can FORMAT and /VERIFY the following devices:

DW devices (Devices for Professional 300 Series processors)
RK05
RK06/RK07
RX01 and RX02 on RX02 drives
RX33

You can *only* /VERIFY [:ONLY] the following devices:

RL01/RL02
RX01
RX50

What Formatting Does

Using the FORMAT command to format a storage volume makes that volume usable to the RT–11 operating system. The FORMAT command does this by writing headers for each block on the volume. The header of a block contains data the device controller must use to transfer data to and from that block.

When to Format a Device

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).

What the FORMAT Verification Procedure Does

When RT–11 verifies a volume, it writes a 16-bit pattern in each word on each block in the volume and then reads each pattern. When RT–11 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.

Confirming Your FORMAT Request

When you issue the FORMAT command, RT–11 displays the prompt:

```
<dev:>/FORMAT-Are you sure?
```

The variable <dev:> represents the drive name and unit number of the volume you want to format. Type Y RETURN to continue the format operation. Type N RETURN or press CTRL/C to abort the operation.

Formatting RX01 and RX02 Diskettes

You can format an RX01 or an RX02 diskette only when you have mounted the diskette in a double-density diskette drive unit (RX02). Unless you use the /SINGLEDENSITY option, RT–11 formats diskettes in double-density format. If you attempt to format a diskette in a single-density drive unit (RX01), RT–11 displays an error message.

Formatting RK06 and RK07 Disks

When you format an RK06 or RK07 disk, RT–11 lists the block numbers of all the bad blocks in the manufacturer's bad-block table and in the software bad-block table.

Checking Foreground Before Formatting

If you try to format a volume while a foreground job is loaded, RT–11 displays the message:

```
Foreground Loaded.
<dev:>/FORMAT-Are you sure?
```

Type Y RETURN to continue with the formatting operation. Type N RETURN or press CTRL/C to abort the operation.

NOTE

Although you can format or verify a volume while a foreground job is loaded, Digital does not recommend this. If you format or verify a volume that the foreground job is using, data on the volume will be written over and corrupted, which can cause the foreground job or the system to crash.

Formatting a Volume Containing Protected Files

If you try to format a volume that contains protected files, RT–11 displays the message:

```
Volume contains protected files; Are you sure?
```

Type Y RETURN to continue the formatting operation. Type N RETURN or press CTRL/C to abort the operation.

Formatting RX33 Diskettes

Formatting RX33 diskettes requires an RX33 diskette drive and an RQDX3 controller of at least microcode revision 2.

If you attempt to format an RX33 diskette and your DU controller is not an RQDX3, FORMAT returns the error message, *?FORMAT-F-Device invalid or not supported by FORMAT*. If you attempt to format an RX33 diskette and your RQDX3 controller is below microcode revision 2, FORMAT returns the error message, *?FORMAT-F-Operation unsupported on old micro-code revision*. Contact your local Digital customer services representative to update your RQDX3 controller.

The RQDX3 controller is locked during the formatting and is unavailable to any other device on that controller. Therefore, you cannot perform I/O to any other device on that controller while you are formatting an RX33 diskette.

If you attempt to format an RX50 diskette in an RX33 diskette drive, FORMAT returns the following query *<device> is an RX50 volume/FORMAT-Are you sure?* Do not format single-sided quad-density RX50 diskettes in the RX33 diskette drive. However, you can format double-sided high-density diskettes previously formatted as RX50s in the RX33 diskette drive.

RX33 and RX50 diskettes are physically similar. Use the color of the metal diskette oxide surface to distinguish them from each other. RX33 diskettes have a black oxide surface, while the surface of the RX50 diskette is brown. The difference is apparent when you compare them.

CAUTION

Do not format RX50 diskettes in an RX33 drive.

To format RX50 diskettes as RX33 diskettes is physically possible. However, RX50 diskettes do not contain the same oxide (storage medium) as RX33 diskettes and cannot safely store 1.2 Mbytes of data. RX50 diskettes therefore can randomly lose data if they are formatted as RX33 diskettes.

Formatting Disks on the Professional Series Computer

Digital recommends formatting DW hard disks before they are used. If you boot a Professional Series Computer from a diskette and the computer contains an unformatted hard disk, a hardware diagnostic picture is displayed on the screen. The diagnostic picture displays the number 010030 followed underneath by 000401, indicating an unformatted hard disk. Software cannot be installed on the hard disk.

RT–11 boots from the diskette after displaying the diagnostic picture. Execute the following sequence of commands to format and install the hard disk:

```
.FORMAT DW:
.INSTALL DW:
.LOAD DW:
.FORMAT/VERIFY:ONLY DW:
```

The command FORMAT DW: generates the informational message *?FORMAT-I-Install DW*.

Attempting to verify a hard disk with FORMAT/VERIFY:ONLY before formatting an unformatted hard disk generates the error message *?FORMAT-F-Invalid device for /VERIFY, or not formatted*.

Formatting Extended Device Units

The FORMAT command recognizes device units greater than 7. So, for example, the command FORMAT D10: is valid.

Formatting Based on CSR location

Formatting of devices at nonstandard addresses is supported and will occur automatically, based on the CSR location specified in the device handler. For DU devices, any supported number of controllers is allowed. For all other devices, only a single controller is allowed.

Even though FORMAT accesses the hardware directly for some devices, FORMAT uses the handler file contents to determine the CSR address.

Options

/PATTERN[:value]

Used with /VERIFY[:ONLY], specifies which 16-bit patterns you want RT–11 to use when it verifies the volume. The optional argument value represents an octal integer in the range 0 to 177777 that denotes which patterns you want used.

The following table lists the verification patterns FORMAT uses and the corresponding values for the argument value.

Verification Bit Patterns

Pattern	Bit Set	Value	16-Bit Pattern
1	0	1	000000
2	1	2	177777
3	2	4	163126
4	3	10	125252
5	4	20	052525
6	5	40	007417
7	6	100	021042
8	7	200	104210
9	8	400	155555
10	9	1000	145454
11	10	2000	146314
12	11	4000	162745
13	12	10000	*
14	13	20000	*
15	14	40000	*
16	15	100000	*

*These patterns are reserved for future use. Currently these bit patterns run the default bit pattern (pattern 8).

In /PATTERN:value, the number you specify for value indicates which bit patterns to run during verification. The preceding table gives the equivalent values for each verification bit pattern. To run more than one bit pattern, add together the values for each pattern you select. For example, suppose you want to run bit patterns 1, 3, and 5. The corresponding values are 1, 4, and 20, for a sum of 25. This is the value you would specify with /PATTERN to run all three bit patterns. If you specify /PATTERN:777, patterns 1 through 9 are run during verification. If you do not use the /PATTERN:value option, RT–11 runs only pattern 8.

After it completes verification, RT–11 displays at the terminal each bad block it found during each verification pass. The format of the verification report is:

```
PATTERN  x
----------
nnnnnn
```

In the preceding example, x represents the pattern number, and nnnnnn represents the bad-block number. RT–11 makes a separate verification pass for each pattern it runs and reports on each pass.

The command line that follows verifies an RL02 disk with the 16-bit patterns denoted by the value 25:

```
.FORMAT/VERIFY/PATTERN:25  DL0:
DL0:/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, RT–11 uses pattern 8.

/[NO]QUERY

/QUERY requests confirmation before it performs formatting or verification. Respond to the query message by typing Y (or any string beginning with Y) RETURN to continue the operation. RT–11 interprets any other response to mean NO; it does not perform the specified operation. /QUERY is the default setting.

/NOQUERY suppresses the confirmation message before it performs formatting or verification. When you use this option in the FORMAT command line, RT–11 displays only the pattern numbers it uses (if it performs verification) and the informational messages. The default setting is /QUERY.

/SINGLEDENSITY

Formats an RX02 double-density diskette in single-density format. The following example uses the /SINGLEDENSITY option to format a diskette in RX02 drive unit 1 as a single-density diskette:

```
.FORMAT/SINGLEDENSITY DY1:
DY1:/FORMAT-Are you sure? Y
?FORMAT-I-Formatting complete
```

/VERIFY[:ONLY]

/VERIFY (without the ONLY) formats and verifies that RT–11 can read from or write to blocks on a disk or diskette. During the verification procedure, if RT–11 finds a bad block, it replaces that block.

Bad-block replacement is a technique in which substitute blocks are provided for blocks that have caused a read or write error. The replacement blocks appear to occupy the disk positions of the original blocks, and the disk appears to contain only good blocks.

/VERIFY writes a bit pattern to each block on a volume and then reads that pattern. After the verification procedure is complete, RT–11 displays at the terminal the block number of each block to which it could not write or from which it could not read.

/VERIFY:ONLY *only* verifies a volume and replaces bad blocks it cannot read; it does not format that volume; but its verification procedure is destructive of information already on the volume.

The following command formats and verifies an RX33 diskette in drive unit 1:

```
.FORMAT/VERIFY DU1:
DU2:/FORMAT-Are you sure? Y
?FORMAT-I-Formatting complete
PATTERN  8
?FORMAT-I-Verification complete
```

The next command only verifies an RL02 disk in drive unit 2:

```
.FORMAT/VERIFY:ONLY DL2:
DY0:/VERIFY-Are you sure? Y
PATTERN  8
?FORMAT-I-Verification complete
```

/WAIT

Initiates the formatting operation, then pauses before formatting begins to wait for you to change volumes. The /WAIT option is useful for single-drive systems.

After RT–11 accepts your command line, it pauses and displays the message *Continue?*. At this time, you can exchange volumes. When the new disk is loaded, type Y RETURN to resume the operation. If you type N RETURN or press CTRL/C, the operation is not performed and control returns to the monitor.

When formatting completes, RT–11 pauses again while you remount the system volume. Mount the system volume and type Y RETURN to terminate the formatting operation. If you type any other response, RT–11 prompts you to mount the system volume until you type Y. RT–11 then displays the monitor prompt.

The following command uses the /WAIT option to format an RX33 diskette:

```
.FORMAT/WAIT DU1:
DU1:/FORMAT-Are you sure? Y
Mount input volume in <device>; Continue? Y
?FORMAT-I-Formatting complete
Mount system volume in <device>; Continue? Y
```

FORTRAN

(FORTRA.SAV, F77.SAV, F77XM.SAV)

Invokes a FORTRAN (FORTRAN IV or FORTRAN–77) compiler to compile one or more source programs.

The default FORTRAN compiler is the FORTRAN IV compiler.

- To invoke the **FORTRAN–77** compiler with the FORTRAN command, you must do *one* of the following:
  - — Issue the SET FORTRA F77 command before you issue the FORTRAN command.
  - — Specify the /F77 option with the FORTRAN command.
- To invoke the **FORTRAN IV** compiler with the FORTRAN command, either you must have not issued a SET FORTRA F77 command or you must do *one* of the following:
  - — Issue the SET FORTRA F4 command before you issue the FORTRAN command.
  - — Specify the /F4 option with the FORTRAN command.

Format

FORTRAN *filespec[s]*

FORTRAN IV Command Options

- /CODE:type
- /DIAGNOSE
- /EXTEND
- **/F4**
- /HEADER
- /I4
- /[NO]**LINENUMBERS**
- /LIST[:filespec]
  - /ALLOCATE:size
- /[NO]OBJECT[:filespec]
  - /ALLOCATE:size
- /ONDEBUG
- /RECORD:length
- /SHOW[:type]
- /STATISTICS
- /[NO]**SWAP**
- /UNITS:value
- /[NO]**VECTORS**
- **/[NO]WARNINGS**

FORTRAN–77 Command Options

- /CHECK
- /CONTINUATIONS
- /EXTEND
- /F77
- /I4
- /[NO]**LINENUMBERS**
- /LIST[:filespec]
  - /ALLOCATE:size
- /[NO]OBJECT[:filespec]
  - /ALLOCATE:size
- /ONDEBUG
- /[NO]**OPTIMIZE**
- /RECORD:length
- /SHOW[:type]
- /STATISTICS
- /[NO]**SWAP**
- /TRACE:type
- /UNITS:value
- /[**NO**]**WARNINGS**
- /WIDE
- /WORKFILE:value

Parameter

filespec[s]

Specifies one or more files to be included in the compilation.

The default input file type for a source file is FOR. The default output file type for a listing file is LST and for an object file is OBJ.

Compiling Multiple Source Files

- To compile multiple source files *into a single object file*, separate the files with plus (+) signs in the command line. RT–11 creates an object file with the same name as the first input file and gives it an OBJ file type, unless you specify a different name and type.
- To compile multiple source files *in independent compilations*, separate the files with commas (,) in the command line. This generates a corresponding object file for each input file or for each set of input files.

Description

Language options are position dependent—that is, they have different meanings depending on where you place them in the command line. Options that follow the FORTRAN command 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.

Differences Between FORTRAN IV Options and FORTRAN–77 Options

- Options supported only under FORTRAN IV are indicated by the phrase *(FORTRAN IV only)* at the beginning of their descriptions.
- Options supported only under FORTRAN–77 are indicated by the phrase *(FORTRAN–77 only)* at the beginning of their descriptions.
- Options supported by both FORTRAN compilers have no restrictions at the beginning of their descriptions.

In addition, some options, such as /SHOW:type, take different arguments or are in some other way not the same, depending on the FORTRAN compiler. For further information, see the appropriate FORTRAN user guide.

Two FORTRAN–77 Compilers—One FORTRAN IV Compiler

There are two FORTRAN–77 compilers; your monitor determines which one you need. If you have invoked FORTRAN–77 and you have a mapped monitor, the FORTRAN command calls the F77XM.SAV monitor; if you have invoked FORTRAN–77 and you have an unmapped monitor, the FORTRAN command calls the F77.SAV monitor.

There is only one FORTRAN IV compiler, FORTRA.SAV.

Options

/ALLOCATE:size

Used with /LIST or /OBJECT, reserves space on the device for the output file. The argument *size* specifies the number of blocks of space to allocate. The meaningful range for this value is from -1 to 65527 decimal blocks. A value of -1 is a special case that creates the largest file possible on the device.

If you do not specify /ALLOCATE, RT–11 uses its standard algorithm for allocating volume space for files.

/CHECK

(FORTRAN–77 only) Checks array references to ensure they are within the specified array address boundaries. The FORTRAN IV functionality of checking is done by the selection of OTS library components, not by compiler action.

/CODE:type

(FORTRAN IV only) Produces 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 FORTRAN IV User's Guide* for a complete description of the types of code and their function.

/CONTINUATIONS:dvalue

(FORTRAN–77 only) Specifies the maximum decimal number of continuation lines in the program. The valid range for *dvalue* is 0 through 99; the default is 19.

/DIAGNOSE

(FORTRAN IV only) Expands the dump information from a crash to include internal compiler tables and buffers. This can help you analyze an internal compiler error.

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

With FORTRAN IV, extends the right margin for source input lines from 72 to 80 columns

With FORTRAN–77, extends the right margin for source input lines from 72 to 132 columns.

/F4

Overrides any SET FORTRA conditional and calls the FORTRAN IV compiler (the default).

/F77

Overrides any SET FORTRA conditional and calls the appropriate FORTRAN–77 compiler.

/HEADER

(FORTRAN IV only) Includes in the printout a list of options that are currently in effect.

/I4

With FORTRAN IV, allocates two words for the default integer data type (though it uses only one-word integers) so that it takes the same physical space as real variables.

With FORTRAN–77, allocates and uses two words for the default integer and logical variable length.

/[NO]LINENUMBERS

/LINENUMBERS includes internal sequence numbers in the executable program. These are useful in debugging a FORTRAN program. They identify the FORTRAN statements that cause run-time diagnostic error messages. This is the default operation.

/NOLINENUMBERS requests no traceback information and 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]

Sends a compiler listing to the printer or to the file specification. This is not the default; you must specify this option if you want a compilation listing.

The /LIST option has different meanings depending on how you specify it. The following examples illustrate the ways you can use the /LIST option:

- To produce a listing on your printer, qualify the command with the /LIST option; for example:

  ```
  .FORTRAN/LIST MYPROG
  ```

- To produce a listing file, specify the /LIST option after the file you want to compile; for example:

  ```
  .FORTRAN MYPROG/LIST
  ```

- To store a listing on a file-structured device, specify /LIST with a device name. The default listing-file name is the same as the input file with a LST file type.

  The following command displays the listing on the terminal:

  ```
  .FORTRAN/LIST:TT: MYPROG
  ```

The next command creates a listing file called MYPROG.LST on DU1:

```
.FORTRAN/LIST:DU1: MYPROG
```

- To include your own listing-file name, specify the name with the /LIST option. For example, the following command compiles A.FOR and B.FOR together, producing object file A.OBJ and listing file AANDB.OUT on device DK:

  ```
  .FORTRAN/LIST:AANDB.OUT A+B
  ```

- To produce a listing file of the same name as a particular file, specify /LIST after that file. For example, the following command compiles A.FOR and B.FOR together, producing files DK:A.OBJ and DU1:B.LST:

  ```
  .FORTRAN A+B/LIST:DU1:
  ```

/[NO]OBJECT[:filespec]

/OBJECT specifies a file name or device for the object file.

Because FORTRAN creates object files by default, the following three commands have the same meaning:

```
.FORTRAN PROG1.FOR
.FORTRAN/OBJECT PROG1.FOR
.FORTRAN PROG1/OBJECT
```

These commands compile PROG1.FOR and produce PROG1.OBJ as output. Like the /LIST option, the /OBJECT option can be either a command option or a file qualifier.

As a command option, /OBJECT applies across the entire command string. The following command, for example, compiles PROG1.FOR and PROG2.FOR separately, creating object files PROG1.OBJ and PROG2.OBJ on DU1:

```
.FORTRAN/OBJECT:DU1: PROG1,PROG2
```

Use /OBJECT as a file qualifier to create an object file with a specific name or destination. The following command compiles PROG1.FOR and PROG2.FOR together, creating files PROG2.LST and PROG2.OBJ.

```
.FORTRAN PROG1+PROG2/LIST/OBJECT
```

/NOOBJECT suppresses the 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 PROG1.FOR and PROG2.FOR together, producing files PROG1.OBJ and PROG2.LST. It also compiles PROG3.FOR and produces PROG3.LST, but does not produce PROG3.OBJ:

```
.FORTRAN PROG1+PROG2/LIST,PROG3/NOOBJECT/LIST
```

/ONDEBUG

Includes debug lines (those that have a D in column one) in the compilation. So, you do not have to edit the file to include these lines in the compilation or to logically remove them.

With this option you can include messages, flags, and conditional branches to help you trace program execution and find an error.

/[NO]OPTIMIZE

(FORTRAN–77 only) /OPTIMIZE enables compiler optimization (the default).

/NOOPTIMIZE inhibits compiler optimization.

/RECORD:length

Overrides the default maximum record length for sequentially formatted input and output. The meaningful range for *length* is from 4 to 4095.

For FORTRAN IV, the default maximum record length is usually 132 characters, while for FORTRAN–77 the default length is usually 136.

/SHOW[:type]

Specifies listing options. The *type* argument specifies a code indicating the kind of listing the compiler is to produce.

The following table summarizes the FORTRAN IV listing options.

FORTRAN IV Listing Options

Type Code	Option Listing
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

You can combine FORTRAN IV listing options by specifying the sum of their numeric codes. For example, the following two options have the same meaning:

```
/SHOW:7
```

or

```
/SHOW:ALL
```

If you specify no code, the default value is 3, a combination of SRC and MAP.

With FORTRAN–77, the codes for the *type* argument are 0 through 3; the following table describes the meaning of each FORTRAN–77 code.

FORTRAN–77 Listing Options

Type Code	Option Listing
0	Minimal listing; diagnostics and program section summary only
1	Source listing and program section summary
2	Source listing, program section summary, and storage map (the default)
3	Source listing, assembly code, program section summary, and section map

/STATISTICS

Includes compilation statistics in the listing, such as amount of memory used, amount of time elapsed, and length of the symbol table.

With FORTRAN–77, /STATISTICS displays workfile access statistics and the number of pages of dynamic memory available and used.

/[NO]SWAP

/SWAP permits the USR (user service routine) to swap over the FORTRAN program in memory. This is the default operation.

/NOSWAP 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 the *RT–11 System Subroutine Library 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.

/TRACE:type

(FORTRAN–77 only) Enables the FORTRAN–77 /S:xxx option. This controls the generation of code used by OTS during error traceback. The following *type* arguments compile the indicated traceback information.

Type Values

Type Value	Description of Compiled Traceback Information
(omitted)	Same as the /NOLINENUMBERS option
ALL	All source statements, functions, and subroutine entries
LIN	Same as ALL
BLO	Selected source statements, all functions, and all subroutine entries (the default)
NAM	All function and subroutine entries
NON	None compiled

/UNITS:value

Specifies the decimal number of FORTRAN logical units to be open at one time. The default is 6.

With FORTRAN IV, the maximum value you can specify for *value* is 16, while for FORTRAN–77, the maximum value you can specify is 99.

/[NO]VECTORS

(FORTRAN IV only) /VECTORS directs FORTRAN to use tables to access multidimensional arrays. This is the default mode of operation.

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

/[NO]WARNINGS

/WARNINGS includes 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 message displays, for example, if you change an index within a DO loop, or if you specify a variable name longer than six characters.

/NOWARNINGS excludes warning messages in FORTRAN compiler diagnostic error messages. This is the default.

/WIDE

(FORTRAN–77 only) Used with /MAP, produces a 132-column compiler 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.

/WORKFILE:value

(FORTRAN–77 only) Sets the workfile size. By default, *value* is octal; include a decimal point after *value* to specify a decimal number. The default workfile size in decimal blocks is 128.

FRUN

(KMON)

Initiates foreground jobs on multi-job monitors. See also R, RUN, SRUN, V/VRUN.

Format

FRUN *filespec*

Filespec Options
/BUFFER:value /NAME:name /PAUSE /TERMINAL:value

Parameter

filespec

Specifies the program to execute. The default input file type is REL and the default input device is DK.

Description

Because this command runs a foreground job, it is valid for multi-job monitors only. Before you run a foreground job, make sure you load into memory the device handlers your foreground job needs. See the *Introduction to RT–11* for an explanation of how to run a foreground job.

Requirements of a Foreground Job

You should be aware of the following three points on the memory requirements of a foreground job:

- You can run only one foreground job at a time. So, if a foreground job is active when you issue the FRUN command, an error message displays on the terminal.
- If a terminated foreground job is occupying memory, RT–11 reclaims that region for your next FRUN job.
- Use the UNLOAD command:
  - When you want to run a second foreground job and you do not need the handlers the first job used. This command can free the memory space occupied by a terminated foreground job. If any handers were loaded just for the terminated foreground job, they can also be unloaded with the UNLOAD command.

— When you are finished running your foreground job so that the memory reserved for it is freed for your background use.

FRUN Runs REL or SAV Jobs

Note that you can use the FRUN command to run a virtual SAV (image) foreground job on a mapped monitor.

Running a Foreground Job in a Completely Virtual Environment

To run a foreground job in the completely virtual environment, use the SRUN command in the following format:

```
FRUN  SY:VBGEXE.SAV[/NAME:prgnam][/options]
```

If you do not specify /NAME:prgnam, VBGEXE prompts with "Program?". Enter a file specification at the prompt to indicate the file to be run as a completely virtual system job. The default file extension is SAV. If you omit the device specification, VBGEXE tries to find the save-image file on the system device. If it cannot find the file there, it tries to find the save-image file on the default data device.

Options

/BUFFER:value

Reserves more space in memory than the actual program size. The *value* argument specifies the octal number of words in memory to allocate.

Virtual Jobs

If you use /BUFFER for a virtual job linked with the /V option (or /XM), RT–11 ignores /BUFFER because it has already provided a buffer in extended memory.

FORTRAN IV Foreground Jobs

You must use this option to execute a FORTRAN IV foreground job.

The following formula determines the space needed to run a FORTRAN program as a foreground job:

```
n = [[504+(35*N)+(R-136)+A*512]/2]+[10*qcount]+[6*num]+[25*INTSET]+[64+R/2]
```

where:

A Specifies the maximum number of files open at one time. Each file opened as double buffered should be counted as two files.

N Specifies the maximum number_{10} 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` Specifies 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.

`qcount` Specifies queue elements.

`num` Specifies the number of channels.

`INTSET` Specifies the SYSLIB INTSET function.

Include the following optional elements in the formula if you want to use the indicated system subroutine library (SYSLIB) functions:

`[10*qcount]` Specifies space for queue elements, which the IQSET function requires.

`[6*num]` Specifies space for the number of channels, which the ICDFN function requires.

`[25*INTSET]` Specifies space for the number of INTSET calls issued in the program, which the INTSET function requires.

`[64+R/2]` Specifies space for completion routines and a second record buffer. Any functions, including INTSET, that invoke completion routines must include 64_{10} 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_{10} bytes, or the length that was set at FORTRAN installation time.

To specify a decimal number for *n*, include a decimal point (.) after the number you specify. The formula for determining the *n* calculates *n* as a decimal number, so include the decimal point when you specify the number.

Two Reasons for an Error Message

- 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
  ```

- The previous message also appears if there is not enough free memory for the background job or if a completion routine in an unmapped monitor is activated during another completion routine. In the latter case, the job aborts; you should use a multi-job unmapped monitor to run multiple active completion routines.

/NAME:name

Assigns 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.

/PAUSE

Helps you debug a program. When you press RETURN at the end of the command string, RT–11 displays the load address of your program and waits. You can examine or modify the program (by using the ODT utility described in the *RT–11 System Utilities Manual*) before starting execution. You must use the RESUME command to start the foreground job.

The following command loads the program DEMOSP.REL, displays the load address, and waits for a RESUME command to begin execution:

```
.FRUN DEMOSP/PAUSE
Loaded at 127276
.RESUME
```

/TERMINAL:value

(Only valid in a multiterminal system) Assigns a terminal to interact with the foreground job. The *value* argument specifies 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 background job and with the monitor, unless you use the SET TT: CONSOL command to change this.

GET

(KMON)

Loads a memory image file into memory.

Format

GET *filespec*

Parameter

filespec

Specifies the memory image file to be loaded. The default file type is SAV. Note that a magtape is not a block-replaceable device and therefore is not permitted with the GET command.

Description

Use the GET command for a background job only. The GET command is useful when you need to modify or debug a program.

- Use GET with the Base, Deposit, Examine, and START commands to test changes. Use the SAVE command to make these changes permanent, except for a virtual program.
- 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 the *RT–11 System Utilities Manual*), and starts the program using the address of ODT's entry point:

```
.GET DEMOSP

.GET ODT

.START

ODT  V05.6
*
```

Using the GET Command to Overlay a Previous Program

- 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, RT–11 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.

HELP

(HELP)

Displays concise DCL command descriptions with examples.

Format

HELP *[topic [subtopic[:item[:item...]]]]*

Command Options

/TERMINAL
/PRINTER

Parameters

topic

Specifies a subject about which you need information. In the help file supplied with RT–11, the topics are the DCL commands that you can use with RT–11.

You can specify a wildcard (*) in place of a topic. In that case, the HELP command lists all the DCL commands with a brief description of each.

If you do not specify a topic, the HELP command displays a brief description of itself.

subtopic

Specifies a category within a topic. Every complete HELP description of a command is divided into the following four subtopics.

Subtopic	Description
Syntax	Shows how to type a command.
Semantics	Briefly explains the meaning of the words used in the preceding *syntax* section.
Options	Lists and briefly describes a command's options. However, you do not have to use the options subheading. Instead, all you need type is the command with the options you want explained; for example: `.HELP DIRECTORY/ALPHABETIZE/NEWFILES`
Examples	Lists a few examples illustrating how to use the specified command.

item

Specifies one member of the subtopic group. For example, the following gives you a brief explanation of the /ALPHABETIZE option for the DIRECTORY command:

```
.HELP DIRECTORY OPTION:ALPHABETIZE
```

You can specify more than one item in the command line if you separate the items with a colon (:). For example, the following displays explanations for both the ALPHABETIZE and NEWFILES options of the DIRECTORY command:

```
.HELP DIRECTORY OPTION:ALPHABETIZE:NEWFILES
```

Description

The HELP command enables you to access the HELP file HELP.SAV. This file, distributed with RT–11, contains information about the RT–11 DCL 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 Guide*.

Options

/PRINTER

Prints the help information on the printer.

/TERMINAL

Displays the HELP information on the terminal. This is the default. When HELP information is listed on a video terminal and SET TT SCOPE is in effect, the display fills one screen at a time. Type RETURN to view the next screen.

Examples

1. The HELP * command lists and briefly describes all the RT–11 DCL commands:

```
.HELP *

ABORT        Terminates, from the system console, a Foreground job
ASSIGN       Associates a logical device name with a physical device
B            Sets a relocation base
BACKUP       Backup/Restore large files or random access devices
  .
  .
  .
```

2. The HELP DATE command displays a brief description of 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 represents the
       first three characters of the name of the month.  Under
       RTEM-11, the current date cannot be changed.

  OPTIONS
       None

  EXAMPLES
       DATE 12-MAR-90
```

3. The HELP DIRECTORY OPTIONS command lists all the valid options of 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
     .
     .
     .
```

4. The HELP DIRECTORY OPTIONS:BRIEF command describes the /BRIEF option of the DIRECTORY command:

```
.HELP DIRECTORY OPTIONS:BRIEF

  BRIEF
       Lists only file names and file types of files; same as /FAST
```

5. The HELP DIRECTORY/B command lists all the DIRECTORY command options beginning with B:

```
.HELP DIRECTORY/B

BADBLOCKS
     Scans the device for bad blocks and types their octal number
BEFORE[:DD:MMM:YY]
     Lists the files created before the specified date.  If the date
     is omitted, the system date is used.
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
```

If you issue the command HELP DIRECTORY/A, all the DIRECTORY options beginning with the letter A would be displayed. In the same way, you can list DIRECTORY options beginning with other letters of the alphabet.

INITIALIZE

(DUP and FILEX)

Writes an RT–11 empty device directory on the specified volume.

INITIALIZEdevice:)

RT–11 Command Options

/BADBLOCKS[:RETAIN]
/FILE:filespec
/[NO]**QUERY**
/REPLACE[:RETAIN]
/SEGMENTS:value
/VOLUMEID[:ONLY]
/WAIT
/RESTORE

Interchange Command Options

```
⎧ /DOS             ⎫
⎪    /WAIT         ⎪
⎨ /INTERCHANGE     ⎬
⎩    /WAIT         ⎭
   /[NO]QUERY
   /VOLUMEID[:ONLY]
```

Parameter

device:

Specifies the volume you want to initialize.

Description

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. When you use the INITIALIZE command with no options, RT–11 simply creates or clears the specified device directory.

Directory Segments

The default numbers of directory segments for RT–11 directory-structured volumes are listed in the description of the /SEGMENTS option. If any default is too small for your needs, see the *RT–11 Installation Guide* for details on changing this default directory size.

Protected Files

If the volume you are initializing has protected files, RT–11 always requests confirmation as in the following example:

```
.INITIALIZE DU1:
DU1:/Initialize; Are you sure? Y
Volume contains protected files; Are you sure? Y
```

Options

/BADBLOCKS[:RETAIN]

Scans a volume for bad blocks and writes files of type BAD over them. That is, for each bad block RT–11 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 empty files describing the good areas of the disk and FILE.BAD entries to cover any bad blocks. This procedure ensures that RT–11 will not attempt to access these bad blocks during routine operations. If RT–11 finds a bad block in either the boot blocks or the volume directory, it displays an error message and the volume is not usable.

You can use the DIRECTORY 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 DL1 and scans for bad blocks:

```
.INITIALIZE/BADBLOCKS DL1:
DL1:Initialize; Are your sure? Y
```

If you use /BADBLOCKS:RETAIN, RT–11 will retain across initialization all files with a BAD file type that it finds on the volume, giving them the name FILE.BAD. RT–11 does not do a bad-block scan. The advantage in using /BADBLOCKS:RETAIN is that initializing takes less time.

Note that some volumes support bad-block replacement; Digital recommends you use the /REPLACE[:RETAIN] option instead of /BADBLOCKS[:RETAIN] 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 RT–11.

If the volume being initialized contains bad blocks, RT–11 displays the locations of the bad blocks in octal and in decimal, as in the following example:

```
.INITIALIZE/BADBLOCKS DL0:
DL0:/Initialize;  Are you sure?  Y
    Block         Type
000120     80.  Hard
000471    313.  Hard
000521    337.  Hard
?DUP-W-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

Initializes a DECtape for DOS–11 format.

/FILE:filespec

Initializes a magtape and creates a bootable tape. For filespec, substitute the primary boot for your type of tape device. See the *RT–11 Installation Guide* for more information.

The following example creates a bootable magtape on an MS device:

```
.INITIALIZE/FILE:MSBOOT.BOT  MS0:
```

/INTERCHANGE

Initializes 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.

/[NO]QUERY

/QUERY requests confirmation before it initializes a device. Respond by typing Y (or any string beginning with Y) RETURN to initiate execution of the command. RT–11 interprets any other response to mean NO. /QUERY is the default operation.

/NOQUERY suppresses the confirmation message RT–11 displays before it proceeds with the initialization.

/REPLACE[:RETAIN]

If you have an RK06, RK07, RL01, or RL02 disk, use this option to scan a disk for bad blocks. If RT–11 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.

If you use /REPLACE:RETAIN, RT–11 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 RT–11. In this case, move the monitor. Use the DIRECTORY/BADBLOCKS/FILES command to determine which files reside on bad blocks.

With an RK06, RK07, RL01, or RL02 disk, 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, RT–11 can replace only those bad blocks that generate a bad-sector error (BSE). With an RL01 or RL02 disk, RT–11 can replace any kind of bad block. The following paragraphs describe how to designate which blocks to replace on an RK06, RK07, RL01, or RL02 disk.

When you use /REPLACE, RT–11 displays 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, RT–11 prompts you to indicate which blocks you want replaced as follows:

```
?DUP-W-Replacement table overflow DEV:
Type RET, 0, or nnnnnn (RET)
Replace block
```

The variable *nnnnnn* represents the octal number of the block you want RT–11 to replace.

After you enter a block number, RT–11 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 RETURN at any time, RT–11 places all bad blocks you have not entered into the replacement table, starting with the first on the disk, until the table is full. RT–11 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 RETURN in response to the first *Replace block* prompt.

/RESTORE

Uninitializes a volume. That is, you can use this option to restore the directory and files that were present on the volume before the previous initialization. However, you can use /RESTORE only if no files have been transferred to the volume since the last time it was initialized. And, you cannot restore volumes that support bad-block replacement, if bad blocks were found during initialization.

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:value

Specifies directory segments when you initialize a volume. Use this option if you need to change the number of directory segments on a disk.

The number of segments in the directory establishes the number of files that can be stored on a device. RT–11 allows a maximum of 72 files for each directory segment and 31 directory segments for each device.

The *value* argument specifies the number of directory segments. The valid range for *value* is from 1 to 31_{10}. The following table shows the default values of *value* for standard RT–11 devices.

Default Directory Sizes

Device	Decimal Number of Segments in Directory
DW (RD50)	16
DW (RD51)	31
DX (RX01)	1
DY (RX02 single-density)	1
DY (RX02 double-density)	4
DZ (RX50)	4

Device	Decimal Number of Segments in Directory
DL (RL01)	16
DL (RL02)	31
DM (RK06/07)	31
DU (Winchester disks)	31
DU (Diskettes)	1
RK (RK05)	16

The number of default segments in a directory depends on the size of the volume. The next table shows the algorithm RT–11 uses for determining directory segments.

Algorithm for Determining Number of Directory Segments

Device Block Size = s	Default Number of Directory Segments
$s \leq 512_{10}$	1
$512_{10} < s \leq 2048_{10}$	4
$2048_{10} < s \leq 12288_{10}$	16_{10}
$s > 12288_{10}$	31_{10}

/VOLUMEID[:ONLY]

/VOLUMEID writes a volume identification on a device when you initialize it. This identification consists of a volume ID and an owner name. The volume ID can be up to 12 characters long for a block-replaceable device and up to 6 characters long for a magtape or interchange diskette when used with /INTERCHANGE. The owner name can be up to 12 characters long for a block-replaceable device and up to 10 characters long for a magtape.

If you use this option with /INTERCHANGE but you specify no volume ID, the volume ID RT11A is automatically assigned.

The following example initializes volume DU1 and writes a volume identification on it:

```
.INITIALIZE/VOLUMEID DU1:
DU1:/Initialize; Are you sure? Y
 Volume ID?  FORTRAN VOL
 Owner?  AJAX-CORP
```

/VOLUMEID:ONLY writes a new volume identification on a device *without* reinitializing the device. However, this option does not work for magtapes; you cannot change the volume ID of a magtape 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, RT–11 begins the procedure but then pauses and waits for you to mount the volume you want to initialize. When RT–11 pauses, it displays the following prompt at the terminal:

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

The variable <device> is the name of the device into which you mount the volume to be initialized. Mount the input volume and type Y RETURN to continue the initialization operation. Type N RETURN or press CTRL/C to abort the operation and return control to the monitor.

After RT–11 completes the initialization process, RT–11 displays the following message prompting you to mount the system volume:

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

Mount the system volume and type Y RETURN. If you type any other response, RT–11 continues to prompt you to mount the system volume until you type Y RETURN.

INSTALL

(KMON)

Enters the device you specify into the monitor's internal tables.

Format

INSTALL *device[,device,...]*

Parameter

device

Specifies the device or devices to be installed.

Description

The INSTALL command accepts no options.

The INSTALL command records a device handler in the monitor's internal tables so that the handler can be used.

Note, however, that RT–11 attempts to install all, up to 40 handlers, at bootstrap time. So, use the INSTALL command if you put a handler on the system disk after rebooting, or if you rename a handler on the system disk.

If you do an INSTALL of a handler that was not SYSGENed for the monitor, the handler will not be installed. If the handler is installed, the INSTALL command establishes all the information in the monitor internal tables that the monitor needs to quickly reach that handler to load it into memory when the LOAD command or the .FETCH programmed request is issued.

INSTALL puts into the monitor internal tables the location of the handler on disk, its size, its name, its device type, and so on.

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. The newly installed device occupies the first available device slot in the system tables.

Device Handlers You Can Install

The INSTALL command only installs a device handler that meets the following requirements:

- The SYSGENed characteristics of the handler and the SYSGENed characteristics of the monitor match.
- Any other special requirements enforced by the installation code within the handler are true.

Further, when you specify a device name with the INSTALL command, RT–11 searches the system volume for the appropriate corresponding device handler file.

For example, if you want to install the LP device, RT–11 seaches for the file SY:LP.SYS when installing LP on an unmapped monitor, and RT–11 searches for the file SY:LPX.SYS when installing LP on a mapped monitor.

Note that you cannot install devices with the names SY, DK, or BA.

Issuing a REMOVE before an INSTALL

The INSTALL command lets you 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.
- If you COPY or RENAME a handler, you should also do a REMOVE to ensure that the system will use the latest copy of the handler. For example, if you take a handler, such as SDH.SYS, and RENAME it to SD.SYS, do a REMOVE and an INSTALL after you change the handler to ensure that the system uses the new version of the handler:

  ```
  .COPY SDH.SYS SD.SYS
  .UNLOAD SD
  .REMOVE SD
  .INSTALL SD
  ```

  In this situation, ignore any error message you may get from issuing the UNLOAD and REMOVE commands.
- Apply this same rule (using REMOVE and INSTALL) to any newly installed handler. Otherwise, RT–11 may use an old copy of the handler rather than the newly installed one.

Examples

1. The INSTALL LS: command installs the serial line printer into the system tables from the file LSx.SYS. The colon (:) that follows the device handler name is optional:

   ```
   .INSTALL LS:
   ```

2. This command installs the line printer and two device handlers:

   ```
   .INSTALL LP:,DU:,DY:
   ```

   Note that if RT–11 cannot install a device handler, it issues an error message and does not install any other handlers following it in the command line. For example, if DU could not be installed in the preceding command, DY would also not be installed.

LIBRARY

(LIBR)

Lets you create, update, modify, list, and maintain library files.

Format

LIBRARY *library input-files*

```
Command Options                              Input-File Options

⎧  /EXTRACT                           ⎫      { /REPLACE }
⎪                                     ⎪      { /UPDATE  }
⎪ ⎡/CREATE                      ⎤     ⎪
⎪ ⎢/DELETE                      ⎥     ⎪
⎪ ⎢/INSERT                      ⎥     ⎪
⎪ ⎢/LIST[:filespec]             ⎥     ⎪
⎪ ⎢   /ALLOCATE:size            ⎥     ⎪
⎨ ⎢/[NO]OBJECT[:filespec]       ⎥     ⎬
⎪ ⎢   /ALLOCATE:size            ⎥     ⎪
⎪ ⎢/PROMPT                      ⎥     ⎪
⎪ ⎣/REMOVE                      ⎦     ⎪
⎪                                     ⎪
⎪  /MACRO[:value]                     ⎪
⎪     /CREATE                         ⎪
⎩     /PROMPT                         ⎭
```

Parameters

library

Specifies the library file name.

input-files

Specifies 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.

Default File Types

- LST is the default file type for library directory listing files.
- OBJ is the default file type for object libraries and object input files.
- MAC is the default file type for macro input files.
- MLB is the default output file type for macro library files.

Object libraries contain machine-level object modules, and macro libraries contain MACRO source modules. You cannot combine object modules with MACRO modules.

Description

The default operation, if you do not specify an option, is /INSERT.

The LIBRARY command:

- Creates, updates, modifies, lists, and maintains object library files.
- Creates macro library files for use with the MACRO–11 assembler.

Library File Description

A library file is a direct-access file (a file that has a directory) that contains one or more modules of the same type. RT–11 organizes library files so the linker and the MACRO–11 assembler can access them rapidly.

Each object 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.

Each MACRO library file is a file that contains a library header, a library directory, and macro definitions.

Inserting Input Files Into a Library

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. However, once you insert an object module into a library file, you refer to the module by the module name, not by the name of the file of which it was a part. For example, the input file FORT.OBJ may exist on DU1: and can contain an object module called ABC. Once you insert the module into a library, only refer to ABC and the library, and not to 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.

Accessing Library Files

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.

See the Librarian (LIBR) utility chapter in the *RT–11 System Utilities Manual* for further information on object and macro libraries.

Combining LIBRARY Options

You can combine the LIBRARY options with the exceptions of /EXTRACT and /MACRO, which you cannot combine with most of the other functions.

Execution and Prompting Sequence of LIBRARY Options

The following table lists the sequence in which RT–11 executes the LIBRARY options and prompts you for additional information.

Option	Prompt
/CREATE	
/DELETE	Module name?
/REMOVE	Global?
/UPDATE	
/REPLACE	
/INSERT	
/LIST	

The following example combines several options:

```
.LIBRARY/LIST:TT:/REMOVE/INSERT NEWLIB LIB2/REPLACE,LIB3
Global? SQRT
Global?
RT-11 LIBRARIAN V05.6  FRI 10-MAY-91 00:08:37
NEWLIB                 FRI 10-MAY-91 00:08:35

MODULE   GLOBALS   GLOBALS   GLOBALS

         COS       SIN
         DATAN     DATAN2
         ATAN      ATAN2
         DCOS      DSIN
```

This LIBRARY command executes in the following sequence:

1. Removes the global SQRT from the NEWLIB.OBJ library directory.
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.

Creating Files for a Macro Library

When creating files for a macro library, you should assemble a macro file before submitting it to the Librarian (LIBR) utility. The MACRO–11 assembler can pick up some errors that LIBR cannot. For example, assume a macro file TEST.MAC, residing on device DK, that contains macros that will become a macro library TEST.MLB, also residing on device DK. First, issue the command MACRO TEST to check for any errors that come from the assembly. Then, issue the command LIBRARY/MACRO TEST to generate the macro library.

See the *RT–11 System Macro Library Manual* for more information on the system macro library (SYSMAC).

Options

/ALLOCATE:size

Use this option only with /LIST or /OBJECT to reserve space on the device for the output file. The value *size* specifies the number of blocks of space to allocate. The meaningful range for this value is from 1 to 65535. 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

Deletes an object module and all its associated global symbols from a library file directory. Since the module is deleted only from the directory (the object module itself is not deleted), the module and all global symbols that were previously deleted are restored whenever you modify that library, unless you use /DELETE again to delete them. Specify the library name in the command line.

RT–11 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 symbol name.) Follow each module name with RETURN. Enter 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

Extracts an object module from a library and stores 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.

RT–11 prompts you for the name of the object module to be extracted. The prompt is:

```
Global?
```

If you enter a global symbol name and press RETURN, RT–11 extracts the entire module of which that global symbol is a part. To terminate the sequence, press RETURN by itself in response to the prompt. The following example shows how to extract the module ATAN from the library called NEWLIB.OBJ and store it in file ATAN.OBJ on DU1:

```
.LIBRARY/EXTRACT RET
Library? NEWLIB RET
File   ? DU1:ATAN RET
Global ? ATAN RET
Global  ? RET
```

/INSERT

Inserts 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]

Obtains a directory listing of an object library.

The following example obtains a directory listing of OLDLIB.OBJ on the terminal (the printer is the default device):

```
.LIBRARY/LIST:TT: OLDLIB
```

The directory listing displays global symbols. A plus sign (+) in the module column indicates a continued line. See the Librarian (LIBR) utility chapter in the *RT–11 System Utilities Manual* 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 displays 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, using the librarian. You can use EDIT/INSPECT of the resultant library and search for all the lines that contain .MACRO. Note that nested macro definitions do not appear in the directory.

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[:value]

Creates a macro library. The optional *value* argument specifies the number of macro names that can be stored in the directory. The default *value* is 128 (which uses two blocks for the macro-name table).

Note that you can create a macro library, but you cannot list or modify it. To update a macro library, simply edit the ASCII source file and then reprocess the file with the LIBRARY/MACRO command.

It is also possible to retrieve the macro definitions from the library using the SPLIT utility and to recreate at least a partial source file. As an example of doing this, see customization patch 59 in the *RT–11 Installation Guide* which describes how to recreate the file SYSMAC.MAC from the distributed SYSMAC.SML file.

The following example creates a macro library called NEWLIB.MLB 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.

/[NO]OBJECT[:filespec]

/OBJECT creates object library files. This is the default. When you modify an existing library, RT–11, by default, makes the changes to the library you specify by 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 an updated library file and to 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 suppresses the creation of a new object library as a result of a LIBRARY command.

/PROMPT

Specifies that there will be 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 two backslashes (//) as the last input to terminate the input list.

Note that the file specifications you enter after typing the /PROMPT option must conform to Command String Interpreter (CSI) conventions.

The following example creates a macro library called MACLIB.MLB from seven input files:

```
.LIBRARY/MACRO/PROMPT MACLIB A,B,C,D RET
*E,F,G RET
*// RET
```

/REMOVE

Lets you delete a specific global symbol from a library file's directory.

Note: Because 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.

RT–11 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 RETURN. Enter 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 RET
Global? GA RET
Global? GB RET
Global? GC RET
Global? GD RET
Global? RET
```

/REPLACE

Replaces 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:

- The /REPLACE option must follow each file specification that contains a module to be inserted into the library.
- You can use /REPLACE only with modules and never with library files.

/UPDATE

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, RT–11 replaces those library modules. If the modules in the input file do not exist in the library, RT–11 inserts them.

The following example updates the library OLDLIB.OBJ:

```
.LIBRARY OLDLIB FIRST/UPDATE,SECOND/UPDATE
```

Note:

- The /UPDATE option must follow each file specification to which it applies.
- You can use this option only with modules, not files.

LINK

(LINK)

Converts object modules into a format suitable for loading and execution.

Format

LINK *filespecs*

Command Options

```
/ALPHABETIZE
/[NO]BITMAP
/DEBUG[:filespec]
/DUPLICATE
/[NO]EXECUTE[:filespec]
   /ALLOCATE:size
/EXTEND[:value[:type]]
/FILL:value[:type]
/GLOBAL
/IDSPACE
/INCLUDE
/LIBRARY:filespec
/LINKLIBRARY:filespec
/MAP[:filespec] [ /ALLOCATE:size ]
                [ /WIDE          ]
/PROMPT
/ROUND:value[:type]
/SLOWLY
/STACK[:value[:type]]
/SYMBOLTABLE[:filespec]
/TRANSFER[:value]

{ /BOTTOM:value[:type]         }
{ /BOUNDARY[:value[:type]]     }
{ /FOREGROUND[:stacksize]      }
{ /LDA                         }
{ /RUN                         }
{ /TOP:value[:type]            }
{ /XM                          }
{    /LIMIT:value              }
```

Parameter

filespecs

Specifies the object modules to be linked. Each input module should be stored on a random-access device; 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.

File Type	Description
SAV, REL (/FOREGROUND), LDA (/LDA)	Load Module
MAP	Map Output
OBJ	Object Module
STB	Symbol Table File

Separate multiple file specifications with commas. The linker creates an executable file with the same name as the first file in the input list (unless you use /EXECUTE to change it).

Description

The linker can:

- Process a program's object modules and its subroutines to relocate each module and assign absolute addresses.

  RT–11 lets you separately assemble a main program and each of its subroutines without assigning an absolute load address at assembly time.
- Link the modules by correlating global symbols that are defined in one module and referenced in another.
- Create the initial control block for the linked program.
- Create an overlay structure (if you specify the /PROMPT option) and include the necessary overlay commands.
- Allow multiple passes for linking SAV image programs with separate Instruction and Data Space.
- Search first any libraries you specify and then the default system subroutine library SYSLIB.OBJ to locate unresolved global symbols.
- Produce a load map (if you specify /MAP) that shows the layout of the executable module.
- Produce a symbol-table file (if you specify /SYMBOLTABLE) that contains the definition of all the global symbols in the image.

Prompting Sequence for LINK Options

The following table summarizes the LINK prompting sequence for commands that combine two or more LINK options. If you issue the /IDSPACE option, you can get the double prompts for the indicated options.

Option	Prompt
/TRANSFER	Transfer symbol?
/STACK	Stack symbol?
/EXTEND:value[:type]	Extend section? Extend instruction section? Extend data section?
/BOUNDARY:value[:type]	Boundary section? Instruction boundary section? Data boundary section?
/ROUND:value[:type]	Round section? Round instruction section? Round data section?
/INCLUDE	Library search?
/DUPLICATE	Duplicate symbol?

If you combine any of the options listed in this table, LINK prompts you for information in the sequence shown in the table. Note that the *Duplicate symbol?* prompt is always last. This and *Library search?* are the only prompts that accept more than one line as a response. For all the prompts, press RETURN to terminate your response. Terminate your list of responses to the *Library search?* and *Duplicate symbol?* prompts by typing an extra RETURN. Note that if the command lines are in a command file and LINK encounters an end-of-file before all the prompting information has been supplied, it displays the prompt messages on the terminal.

See the LINK utility chapter in the *RT–11 System Utilities Manual* for a more detailed explanation of the RT–11 linker. See the *Introduction to RT–11* for a tutorial explanation of the linker.

Options

/ALLOCATE:size

Used with /EXECUTE or /MAP, reserves space on the device for the output file. The *size* argument specifies the number of blocks of space to allocate. The meaningful range for this value is from -1 to 65527. A value of -1 is a special case that creates the largest file possible on the device.

/ALPHABETIZE

Lists in the load map your program's global symbols in alphabetical order.

/[NO]BITMAP

/BITMAP creates a memory usage bitmap. This is the default setting.

/NOBITMAP suppresses the creation of 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. /NOBITMAP is also used in linking many handlers. /BITMAP is the the default setting.

/NOBITMAP causes the I and D space bitmaps to be suppressed. The absence of /NOBITMAP causes both I and D space bitmaps to be generated. One cannot be generated and the other suppressed.

/BOTTOM:value[:type]

Specifies the lowest address to be used by the relocatable code in the load module. The *value* argument specifies 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_8. This option is invalid for foreground links.

The optional type argument to the value can be DAS or INS and is used only if you also specify the /IDSPACE option. When specified with /IDSPACE:

- /BOTTOM:value:DAS specifies the lowest address to be used by the D-space code in the load module.
- /BOTTOM:value:INS specifies the lowest address to be used by the D-space code in the load module.
- /BOTTOM:value:INS is the default; that is, /BOTTOM:value:INS and /BOTTOM:value have the same effect.

If /BOTTOM is not used to specify a value for either the I-space or D-space code, 1000_8 is used as the default for that space or spaces.

/BOTTOM and /TOP are mutually exclusive options for a particular space. However, you can use /BOTTOM for one data space and /TOP for the other. For example, /BOTTOM:value:DAS and /TOP:value:INS are valid to use together.

/BOUNDARY[:value[:type]]

Starts a specific program section in the root on a particular address boundary. LINK generates a whole-number multiple of the value you specify for the starting address of the program section. If specified, the *value* argument must be a power of 2. LINK extends the size of the previous program section to accommodate the new starting address for the specific section.

The optional type argument to the value can be DAS or INS and is used only if you also specify the /IDSPACE option. When specified with /IDSPACE:

- /BOUNDARY:value:DAS specifies a particular address boundary at which a specified D-space PSECT in the root begins.
- /BOUNDARY:value:INS specifies a particular address boundary at which a specified I-space PSECT in the root begins.
- /BOUNDARY:value:INS is the default; that is, /BOUNDARY:value:INS has the same effect as /BOUNDARY:value.

/BOUNDARY and /TOP are mutually exclusive options for a particular space. However, you can use /BOUNDARY for one data space and /TOP for the other. For example, /BOUNDARY:value:DAS and /TOP:value:INS are valid to use together.

When you have entered the complete LINK command, LINK prompts you for the name of the section whose starting address you need to modify.

- If you do not also use the /IDSPACE option, the prompt is:

  ```
  Boundary section?
  ```

- If you use the /IDSPACE option, the prompt is either one or both of the following, depending on whether one or both types of /BOUNDARY are specified. If both types are specified, the prompts are issued in the following order:

  ```
  Instruction boundary section?
  Data boundary section?
  ```

Respond with the appropriate value and/or program section name(s), and terminate your response with RETURN. The sections specified in answer to these prompts are verified to be I-space or D-space sections, as appropriate. If not, an error message is displayed.

If you do not want to specify a value, respond with only the appropriate program section name. If you want to specify a value, respond in the following format:

value[:type]

where:

value	specifies the address boundary you assign that PSECT.
type	specifies the abbreviation for the PSECT name, the name of the section whose starting address you need to modify. DAS is for the data section and INS is for the instruction section.

Not specifying the value parameter causes LINK to prompt for up to eight separate PSECT boundary addresses. You terminate the prompt sequence by pressing RET with no specified value.

If you do not specify a value, any value that was entered at the /BOUNDARY:value option is used, and prompting stops. If a value is not specified at the command line or the prompt, the default value 1000_8 is used as the boundary address.

/DEBUG[:filespec]

Links a debugger (ODT, VDT, or user supplied) with your program to help you debug it. See the ODT (on-line debugging technique) or the VDT chapter in the *RT–11 System Utilities Manual*. If you supply the name of another debugging program, LINK links the debugger you specify with your program. LINK links the debugger low in memory relative to your program.

/DUPLICATE

Places duplicate copies of a library module in each overlay segment that references the module. This option is useful in reducing the size of the root segment of your program.

When you have entered the complete LINK command, LINK prompts you for the names of the global symbols in the library module you want to duplicate. The prompt is:

```
Duplicate symbol?
```

Respond by typing the name of each global symbol in a module you want to duplicate. Type RETURN after each global symbol. Type RETURN on a line by itself to terminate the list.

See the LINK utility chapter in the *RT–11 System Utilities Manual* for more information on duplicating library modules.

/[NO]EXECUTE[:filespec]

/EXECUTE specifies 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 a different meaning depending on whether it follows the command or follows the file specification. For example, the following command creates an executable file called PROG1.SAV on device DU1:

```
.LINK/EXECUTE:DU1: PROG1,PROG2
```

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

```
.LINK RTN1,RTN2,MYPROG/EXECUTE
```

/NOEXECUTE suppresses the creation of an executable file.

/EXTEND:value[:type]

Extends a program section to a specific octal value *value*. The resultant program section size is equal to or greater than the value you specify, depending on the space the object code requires.

The optional type argument to the value can be DAS or INS and is used only if you also specify the /IDSPACE option. When specified with /IDSPACE:

- /EXTEND:value:DAS specifies the minimum size to allocate to a D-space PSECT that you specify.
- /EXTEND:value:INS specifies the minimum size to allocate to an I-space PSECT that you specify.
- /EXTEND:value:INS is the default; that is, /EXTEND:value:INS and /EXTEND:value have the same effect.

When you have entered the complete LINK command, LINK prompts you for the name of the program section you need to extend:

- If you do not also use the /IDSPACE option, the prompt is:

  ```
  Extend section?
  ```

- If you use the /IDSPACE option, the prompt is either one or both of the following, depending on whether one or both types of /EXTEND are specified. If both types are specified, the prompts are issued in the following order:

  ```
  Extend instruction section?
  Extend data section?
  ```

Respond with the appropriate program section name(s), and terminate your response with RETURN. The sections specified in answer to these prompts are verified to be I-space or D-space sections, as appropriate. If not, an error message is displayed.

/FILL:value[:type]

Initializes unused locations in the load module and places a specific octal value *value* 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.

The optional type argument to the value can be DAS or INS and is used only if you also specify the /IDSPACE option. When specified with /IDSPACE:

- /FILL:value:DAS initializes all the unused D-space locations in the load module with *value*.
- /FILL:value:INS initializes all the unused I-space locations in the load module with *value*.
- /FILL:value:INS is the default; that is, /FILL:value:INS and /FILL:value have the same effect.

/FOREGROUND[:stacksize]

Produces an executable file in relocatable (REL) format for use as a foreground job under a multi-job monitor.

If a REL job has never been run or if the last time the REL job was run, it was run in the background, then that job can be run under a single-job monitor.

This option assigns the default file type REL to the executable file. The *stacksize* argument specifies 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 *stacksize* with a decimal point (stacksize.) to specify a decimal number. The default value is 128_{10} or 200_8 bytes of stack space. Digital recommends that you allocate 256. bytes of stack space when linking a FORTRAN program to run in the foreground.

The /FOREGROUND[:stacksize] option is incompatible with the /IDSPACE option.

/GLOBAL

Generates a global symbol cross-reference section in the load map. The global symbols are listed alphabetically. Each module in which a symbol is referenced or defined is listed in alphabetical order after the global symbol. A number sign (#) after a module name indicates that the global symbol is defined in that module. A plus sign (+) after a module name indicates that the module is from a library. See the LINK utility chapter in the *RT–11 System Utilities Manual* for an example of a load map that includes a global symbol cross-reference table, and for a more detailed description of how to interpret a load map.

Note that LINK does not generate a load map by default. You must also specify /MAP in the command line to get a cross-reference section.

The following command produces a map listing file, MYPROG.MAP, that contains a global symbol cross-reference section:

```
.LINK/GLOBAL/MAP:DU1:  MYPROG
```

When you request a global symbol cross-reference listing with the /GLOBAL option, LINK generates the temporary file DK:CREF.TMP.

If DK is write locked or if it contains insufficient free space for the temporary file, you can designate another device for the file. To designate another device for the temporary file, assign the logical name CF to the device by using the following command:

```
.ASSIGN dev: CF
```

If you have assigned CF to a physical device for the MACRO cross-reference temporary listing file CREF.TMP, that device will also serve as the default device for the LINK global-symbol cross-reference temporary file.

/IDSPACE

Causes LINK to generate an extended SAV image file which separates I and D space; that is, /IDSPACE causes LINK to produce a separated I and D space program. /IDSPACE and /FOREGROUND[:stacksize] are incompatible and generate an error message when used together.

The following options are modified based on the presence of this option:
/BOTTOM:value, /BOUNDARY:value, /EXTEND:value, /FILL:value,
/FOREGROUND[:stacksize], /NOBITMAP, /ROUND:value, /STACK[:value],
/TOP:value, and /TRANSFER[:value]. See the description of each option for how it is modified.

/INCLUDE

Lets you take global symbols from any library and include the object modules the symbols specify in the linked memory image. When you use /INCLUDE, the linker loads modules into the root that are not called by other modules (from a library).

When you have entered the complete LINK command, LINK prompts you for a list of global symbols specifying object modules to include in the load module. The prompt is:

```
Library search?
```

Respond by typing the global symbols specifying the object modules to be included in the load module. Press RETURN after each global symbol, and press RETURN in response to the *Library search?* prompt itself to terminate the list.

/LDA

Produces an executable file in LDA format. The LDA-format file can be output to any device, including those that are not block replaceable. 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:filespec

This option is the same as /LINKLIBRARY. It is included here only for system compatibility.

/LIMIT:value

This option is ignored by LINK, but is included for compatibility with the RSTS /E operating system.

/LINKLIBRARY:filespec

Includes the library file you specify as an object-module library in the linking operation. Because LINK 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]

Produces a load-map listing. This is not the default; you must specify this option to produce a load-map listing.

The optional *filespec* argument specifies where you want the listing placed.

- If you omit this argument, LINK generates a load-map listing on the printer.
- If you specify a device as the argument, LINK creates a load-map file on that device. For example, the following command produces a load map on the terminal:

  ```
  .LINK/MAP:TT:  MYPROG
  ```

- If you specify a file-structured device as the argument, LINK stores the listing file on that device, assigning it the same name as the first input file with a MAP file type. For example, the next command creates a map listing file called MYPROG.MAP on DU1:

  ```
  .LINK/MAP:DU1:  MYPROG
  ```

- If you specify a file name, LINK gives the map listing file that name. In the following command, LINK links PROG1 and PROG2, producing a map listing file called MAP.OUT on device DK:

  ```
  .LINK/MAP:MAP.OUT  PROG1,PROG2
  ```

The contents of the load map vary depending on where you place the /MAP option in the command line:

- If you specify the /MAP option along with the other options immediately following the command, the map applies to the command as a whole and includes all the files specified in the command.
- If you specify /MAP after a file specification, LINK produces a map listing file of only the specified file; and the map file name has the same name as that file. For example, in the following command, LINK links PROG1 and PROG2, producing files PROG2.SAV and PROG2.MAP:

  ```
  .LINK  PROG1,PROG2/EXECUTE/MAP
  ```

/PROMPT

Enters additional lines of input. LINK continues to accept lines of linker input until you enter two slashes (//). The LINK utility chapter in the *RT–11 System Utilities Manual* 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 the description of the Command String Interpreter in the *RT–11 System Utilities Manual*).

The example that follows uses the /PROMPT option to create an overlay structure for the program COSINE.MAC:

```
.LINK/PROMPT COSINE  RET
*TAN/O:1 RET
*COS1/O:1 RET
*SIN3/O:2 RET
*LML3/O:2// RET
```

The /PROMPT option also gives you a convenient way to create an overlaid program from a command file. The file PROMPT.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  @PROMPT
```

/ROUND:value[:type]

Rounds up the section you specify so that the size of the root segment is a whole-number multiple of the *value* you supply. The *value* argument must be a power of 2.

The optional *type* argument to the /ROUND value can be DAS or INS and is used only if you also specify the /IDSPACE option. When specified with /IDSPACE:

- /ROUND:value:DAS specifies the size boundary for the D-space root. This size must be an integer multiple of *value*; that is *value* must be a power of 2. The size of the specified D-space PSECT is rounded up the minimum amount necessary to accomplish this.
- /ROUND:value:INS specifies the size boundary for the I-space root. This size must be an integer multiple of *value*; that is *value* must be a power of 2. The size of the specified I-space PSECT is rounded up the minimum amount necessary to accomplish this.
- /ROUND:value:INS is the default; that is /ROUND:value:INS and /ROUND:value have the same effect.

When you have entered the complete LINK command, LINK prompts you for the name of the program section you need to round up:

- If you do not also use the /IDSPACE option, the prompt is:

  ```
  Round section?
  ```

- If you use the /IDSPACE option, the prompt is either one or both of the following, depending on whether one or both types of /ROUND are specified. If both types are specified, the prompts are issued in the following order:

  ```
  Round instruction section?
  Round data section?
  ```

Respond with the appropriate program section name(s), and terminate your response with RETURN. The sections specified in answer to these prompts are verified to be I-space or D-space sections, as appropriate. If not, an error message is displayed.

/RUN

Initiates 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

Instructs LINK 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.

If you are running under a mapped monitor, and the linker fails through lack of memory area, issue the command SET RUN VBGEXE and retry the LINK command.

/STACK[:value]

Modifies the stack address, location 42, which is the address that contains the value for the stack pointer (SP). When your program executes, the monitor sets

SP to the contents of location 42. The *value* argument is an even, unsigned, six-digit octal number that defines the stack address.

When you have entered the complete LINK command, LINK displays the following prompt message if you did not already specify a value:

```
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 RETURN. If you specify a nonexistent symbol, LINK displays an error message. It then sets the stack address to 1000 (for memory image files) or to the bottom address if you used /BOTTOM.

When used together with the /IDSPACE option, if *value* is not specified, the specified stack symbol is verified to be in D-space. If it is not, an error message is displayed.

/SYMBOLTABLE[:filespec]

Creates a file containing 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 the file type STB.

By default, LINK does not create a symbol-table file.

The following example creates the symbol-table file BTAN.STB:

```
.LINK AOBJ,BOBJ/SYMBOLTABLE:BTAN
```

/TOP:value[:type]

Specifies the highest address to be used by the relocatable code in the load module. The *value* argument specifies an unsigned, even octal number.

The optional type argument to the value can be DAS or INS and is used only if you also specify the /IDSPACE option. When specified with /IDSPACE:

- /TOP:value:DAS specifies the highest address to be used by the D-space code in the load module. The *value* must be even.
- /TOP:value:INS specifies the highest address to be used by the I-space code in the load module. The *value* must be even.
- /TOP:value:INS is the default; that is /TOP:value:INS and /TOP:value have the same effect.

The /BOTTOM and /TOP options are mutually exclusive for a particular space. However, you can use /BOTTOM for one data space and /TOP for the other. For example, /BOTTOM:value:DAS and /TOP:value:INS are valid to use together.

The /BOUNDARY and /TOP options are also mutually exclusive for a particular space. However, you can use /BOUNDARY for one data space and /TOP for the other. For example, /BOUNDARY:value:DAS and /TOP:value:INS are valid to use together.

/TRANSFER[:value]

Lets you specify the transfer (start) address of the load module. The transfer address is the address at which a program starts when you initiate execution with R, RUN, FRUN, SRUN, V, or VRUN.

The *value* argument is an even, unsigned, six-digit octal number that defines the transfer address. If the transfer address you specify is odd, the program does not execute after loading, and control returns to the monitor.

When you have entered the complete LINK command, LINK displays the following prompt message if you did not already specify a value:

```
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 RETURN. If you specify a nonexistent symbol, an error message displays and the linker sets the transfer address to 1 so that LINK cannot execute the program.

When /TRANSFER[:value] is specified with /IDSPACE, if *value* is not specified, the specified transfer symbol is verified to be in I-space. If it is not, an error message is displayed.

/WIDE

Used with /MAP, produces 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.

/XM

Enables special .SETTOP and .LIMIT features provided in a mapped monitor. This option allows a virtual job to map a scratch region in extended memory with the .SETTOP programmed request. See the *RT–11 System Macro Library Manual* and the *RT–11 System Internals Manual* for more details on these special features.

To create an extended memory overlay structure for your program, use the /PROMPT option. You can then specify the overlay structure on subsequent lines, using the LINK (CSI) /V option (see the LINK chapters in the *RT–11 System Utilities Manual*). When you use /V to create an overlay structure, the linker automatically enables the special .SETTOP and .LIMIT features.

LOAD

(KMON)

Copies a device handler from the system device into memory for use with foreground, background, system jobs, or BATCH.

Format

LOAD *device[=jobname][,...device[=jobname]]*

Parameter

device

Specifies the device handler to be made resident in memory.

The optional *jobname* assigns the device to a job. This means that the device is restricted to that job, only that job can use the device. The *jobname* specification is invalid with a single-job monitor.

- To assign the device handler to the background job, assign it the name B:

  ```
  LOAD dev=B
  ```

- To assign the device handler to the foreground job, assign it the name F:

  ```
  LOAD dev=F
  ```

- To assign the device handler for exclusive use by a program, specify the program name. So, to assign the XL communication-port handler to VTCOM and to assign the LS serial-interface printer handler to SPOOL, do the following:

  ```
  LOAD XL=VTCOM
  LOAD LS=SPOOL
  ```

  Note, however, that the program job must be running before you can assign the job name to the device handler.

See the Using the Foreground/Background Monitor chapter and the Using the System Job Feature chapter in the *Introduction to RT–11* for a description of loading and calling foreground, background, and system jobs.

Description

To use a device handler, it must be installed and loaded into memory:

- RT–11 automatically installs most device handlers. See the INSTALL command.
- RT–11 automatically loads the system device handler when you boot your system.

When to Load a Handler

Before you use a device in a foreground or system program, you must first load the device handler since foreground programs cannot fetch. Also, if you have generated a mapped monitor without fetchable handler support or if your handler is not fetchable, you must load the device handler before it can be used. A fetchable handler is one having the .FETCH programmed request enabling the monitor to load that handler. RT–11 distributed monitors come with support for the .FETCH programmed request.

Devices Owned Exclusively by a Job

A device can be owned exclusively by either the foreground, background, or system job. This exclusive ownership prevents other jobs from using the assigned device. In the following example, the XL device is exclusively owned by VTCOM:

```
.LOAD XL=VTCOM
```

Different Units of a Device Owned by Different Jobs

Different units of the same random-access device controller can be owned by different jobs. That is, DL1 can belong to the background job, while DL3 can belong to the foreground or system job. If no ownership is indicated, the device is available for use by any job.

NOTE

If you use the LOAD command to load a device handler, and assign ownership of that handler to a job, all units of that device become assigned to that job. This means no other job can use any unit of that device.

Changing Job Ownership of a Device Handler

To change ownership of a device handler, use another LOAD command with the =*jobname* parameter. The second LOAD command reassigns the ownership. It is not necessary to first unload the device.

However, if you interrupt an operation that involves magtape, you must unload (with the UNLOAD command) and then load the appropriate device handler (MM, MT, or MS). When using these handlers with the FB monitor, this restriction does not apply.

Loading Units of the Same Type As the System Unit

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.

When Booting from a Diskette

If you booted from 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.

Using LOAD with Logical Names

LOAD is valid for use with logical names. For example:

```
.ASSIGN DL: XY
.LOAD XY:=F
```

Finding the Status of Device Handlers

Use the SHOW command to display on the terminal the status of device handlers and device ownership.

MACRO

(MACRO)

Invokes the MACRO assembler to assemble one or more source files.

Format

MACRO *filespec[s]*

Command Options	Input-File Options
/CROSSREFERENCE[:type[:type...]]	/LIBRARY
/DISABLE:type[:type...]	
/ENABLE:type[:type...]	
/LIST[:filespec]	
/ALLOCATE:size	
/[NO]/**OBJECT**[:filespec]	
/ALLOCATE:size	
/[NO]SHOW:type[:type...]	

Parameter

filespec[s]

Specifies one or more files to be included in the assembly. If you omit a file type for an input file, RT–11 assumes MAC for a source program and MAC or MLB for a file used with /LIBRARY. Output default file types are LST for listing files and OBJ for object files.

Assembling Multiple Source Files

- To assemble multiple source files *into a single object file*, separate the files with plus (+) signs in the command line. RT–11 creates an object file with the same name as the first input file and gives it an OBJ file type, unless you specify a different name and type.
- To assemble multiple source files *in independent compilations*, separate the files with commas (,) in the command line. This generates a corresponding object file for each input file or for each set of input files.

Description

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.

RT–11 displays on the terminal the number of errors MACRO detects during an assembly.

The MACRO–11 Assembler Program chapter in the *RT–11 System Utilities Manual* and in the *PDP–11 MACRO Language Reference Manual* contains more detailed information about using MACRO.

Options

/ALLOCATE:size

Used with /LIST or /OBJECT, reserves space on the device for the output file. The *size* argument specifies the number of blocks of space to allocate. The meaningful range for this value is from -1 to 65527. A value of -1 is a special case that creates the largest file possible on the device.

/CROSSREFERENCE[:type[:type...]]

Generates a symbol cross-reference section in the listing. This information is useful for program maintenance and debugging. Note that RT–11 does not generate a listing by default. You must also specify /LIST in the command line to get a cross-reference listing. The optional *type* argument specifies a one-character code that indicates which sections of the cross-reference listing the assembler should include. The following table lists the type arguments with their meanings.

Summary Listing of Cross-Reference Section Types

Type	**Meaning**
C	Control sections (.CSECT symbolic names)
E	Error codes
M	Macro symbolic names
P	Permanent symbols (instructions, directives)
R	Register symbols
S	User-defined symbols
None	Equivalent to :S:M:E

/DISABLE:type[:type...]

Specifies a MACRO .DSABL directive. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all valid types. The following table lists the types with their meanings.

Summary Listing of .DSABL and .ENABL Types

Type	Default	What Is Enabled or Disabled
ABS	Disable	Absolute binary output
AMA	Disable	Assembly of all absolute addresses as relative addresses
CDR	Disable	Treating source columns 73 and greater as comments
DBG	Disable	Generation of internal symbol directory (ISD) records during assembly (See Chapter 2 of the *RT–11 Volume and File Formats Manual* for more information on ISD records.)
FPT	Disable	Floating-point truncation
GBL	Enable	Treating undefined symbols as globals
LC	Enable	Accepting lowercase ASCII input
LCM	Disable	Uppercase and lowercase sensitivity of MACRO–11 conditional assembly directives .IF IDN and .IF DIF
LSB	Disable	Local symbol block
PNC	Enable	Binary output
REG	Enable	Mnemonic definitions of registers

/ENABLE:type[:type...]

Specifies a MACRO .ENABL directive. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all valid types. See the description of the /DISABLE option for a table summarizing the type arguments and meanings.

/LIBRARY

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 following example, 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.MLB, and then to SYSMAC.SML to satisfy .MCALL references. When it assembles file B, MACRO searches MYLIB2.MLB, MYLIB1.MLB, and then SYSMAC.SML, in that order, to satisfy references.

/LIST[:filespec]

Produces a MACRO assembly listing. This is not the default; you must specify this option to produce an assembly listing.

The /LIST option has different meanings depending on where you place it in the command line:

- To produce a listing on the printer, specify /LIST after the command name. For example, the following command produces a printer listing after compiling a MACRO source file:

  ```
  .MACRO/LIST MYPROG
  ```

- To produce a listing on a specific device, specify /LIST with the desired device name. The following command produces a listing file on the terminal:

  ```
  .MACRO/LIST:TT: MYPROG
  ```

  If the device you specify is a file-structured one, RT–11 stores the listing file on that device, assigning it the same name as the input file with a LST file type. The next command creates a listing file called MYPROG.LST on DU1:

  ```
  .MACRO/LIST:DU1: MYPROG
  ```

- To produce a listing file with the same name as a particular input file to which it alone applies, specify /LIST after the file specification to which it applies. The following command assembles A.MAC and B.MAC, producing files DK:A.OBJ, DK:B.OBJ, and a listing file for B only, DU1:B.LST. The command does not produce a listing file for A.MAC:

  ```
  .MACRO A,B/LIST:DU1:
  ```

- To produce a listing file with the same name as a particular input file but applying to all the files assembled, specify /LIST after the file specifications joined by plus (+) signs. The following command assembles A.MAC and B.MAC, producing files DK:A.OBJ and DU1:B.LST:

  ```
  .MACRO A+B/LIST:DU1:
  ```

If the /LIST option contains a name and file type to override the default of LST, RT–11 generates a listing file with that name. The following command, for example, assembles A.MAC and B.MAC together, producing files A.OBJ and AANDB.OUT on device DK:

```
.MACRO/LIST:AANDB.OUT A+B
```

You should be careful when compiling several files separated by commas. For example, in the following case, FILE2.LST is first created when A.MAC is compiled; then, FILE2.LST is replaced when B.MAC is compiled. The second listing file replaces the first, leaving you with only the second listing:

```
.MACRO/LIST:FILE2 A,B
```

/[NO]OBJECT[:filespec]

/OBJECT specifies a file name or device for the object file. Because MACRO creates object files by default, the following three commands have the same meaning:

```
.MACRO A
.MACRO/OBJECT A
.MACRO A/OBJECT
```

These commands assemble A.MAC and produce A.OBJ as output.

The /OBJECT option functions like the /LIST option; it can be either a command option 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 DU1:

  ```
  .MACRO/OBJECT:DU1: A,B
  ```

- As a file option, /OBJECT applies only to the file with which it is specified. Use /OBJECT as a file qualifier 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 suppresses the creation of an object file. As a command option, /NOOBJECT suppresses all object files; as a file qualifier, it suppresses only the object file produced by the related input files. In this command, for example, RT–11 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
```

/[NO]SHOW:type[:type...]

/SHOW:type specifies any MACRO .LIST directive. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives. The following table summarizes the arguments and their meanings. Note that you must explicitly request a listing file by specifying the /LIST option.

.LIST and .NLIST Directive Summary

Argument	Default	Controls
BEX	List	Extended Binary code
BIN	List	Generated binary code
CND	List	Unsatisfied conditionals, .IF and .ENDC statements
COM	List	Comments
LD	Nolist	Listing directives with no arguments
LOC	List	Location counter
MC	List	Macro calls, repeat range expansions
MD	List	Macro definitions, repeat range expansions
ME	Nolist	Macro expansions
MEB	Nolist	Macro expansion binary code
SEQ	List	Source line sequence numbers

Argument	Default	Controls
SRC	List	Source code
SYM	List	Symbol table
TOC	List	Table of contents
TTM	Line printer	Wide or narrow listing format

/NOSHOW:type specifies any MACRO .NLIST directive. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives. The preceding table summarizes the valid arguments and their meanings. Note that you must explicitly request a listing file by specifying the /LIST option.

MOUNT

(LD)

Associates the logical-disk unit (LDn) you specify with the file you specify. This enables you to use a logical disk as if it were a physical disk.

Format

MOUNT *logical-disk-unit filespec [logical-device-name]*

Command Options
/READONLY /[NO]WRITE

Parameters

logical-disk-unit

Specifies the the logical-disk unit you want to mount; for example, LD7 specifies logical-disk unit 7. If you do not specify a number for a logical-disk unit, the number defaults to 0.

The MOUNT command associates a logical-disk file with one of eight logical-disk units (LD0–LD7). If you perform a system generation (SYSGEN), and request extended device-unit support, you can access up to 32 logical-disk units at one time.

filespec

Specifies the device and file being mounted. The default file type for a file mounted as a logical disk is DSK.

logical-device-name

Specifies an optional logical name assigned to the logical disk. The logical device name can be one to three characters long, followed by an optional colon (:). All alphanumeric characters are valid, but the first character must be a letter.

Description

See the *Introduction to RT–11* for a tutorial explanation of how to use the Logical Disk (LD) runnable handler, and see the *RT–11 System Utilities Manual* for a description of that handler.

The file you associate with a logical-disk unit is protected from deletion after you associate it with a unit. The handler for the device containing the file must be loaded. If the handler is not loaded, the MOUNT command loads that handler.

Even if you reboot the computer, an association (but not the logical name assignment) made with the MOUNT command is maintained until you change it. However, the MOUNT information is contained within the image of the LD handler. So, if you explicitly unload the device handler referenced in the *filespec* in the MOUNT command, or if you reboot your processor, the access to the association is lost until you reload the handler. For example, this can happen if you switch between a mapped and an unmapped monitor.

Verifying Logical-Disk Assignments

Issue the SHOW SUBSET command to verify your logical-disk assignments:

```
.SHOW SUBSET
```

Freeing Nested Logical-Disk Units

If you dismount a primary logical-disk file and its unit without first dismounting nested logical-disk units, issue the SET LD CLEAN command to free the nested logical-disk units. See the SET command description for more information on SET LD CLEAN.

The DISMOUNT Command Disassociates a Logical-Disk Association

Use the DISMOUNT command to disassociate a logical disk from a logical-disk unit.

Options

/READONLY

Does not allow WRITE access through LD to any files contained in the specified logical-disk unit.

/[NO]WRITE

/WRITE makes the logical disk write enabled. This is the default.

/NOWRITE makes the logical disk write protected. This option allows you read-only access to the logical disk you specify.

Examples

1. This command example associates logical-disk unit 5 (LD5) with the file DATA.DSK on device DU1:

   ```
   .MOUNT LD5: DU1:DATA
   ```

2. This command example associates LD5 with the file DU1:DATA.DSK, after LD5 has been assigned the logical-device name OUT. When the command is executed, the logical-device name TST is also assigned to LD5:

   ```
   .ASSIGN LD5: OUT
   ```

   ```
   .MOUNT
   Device? OUT
   File? DL0:DATA TST
   ```

Note: Be careful to avoid accidentally destroying files while performing logical-disk assigning. You can assign logical-disk unit numbers to both protected and system (SYS) files, and write to those files.

3. This command example write protects LD0:

```
.MOUNT/NOWRITE LD0: DU1:MYFILE.DSK
```

PRINT

(PIP, QUEMAN, QUEUE, and SPOOL)

Sends the contents of one or more files to the printer.

Format

PRINT *filespec[s]*

Command Options

```
/BEFORE[:date]
/SINCE[:date]
/DATE[:date]
/NEWFILES
/COPIES:value
/DELETE
/[NO]FLAGPAGE:value
/INFORMATION
/[NO]LOG
/NAME:[dev:]jobname
/OUTPUT:dev:[filespec]
/PRINTER
/PROMPT
/QUERY
/WAIT
```

Parameter

filespec[s]

Specifies the file or files to be printed. You can explicitly specify up to six file specifications as input to the PRINT command. RT–11 prints the files in the order in which you specify them in the command line. If you specify more than one file, separate the files by commas.

You can also use wildcards in the file specification. In this case, RT–11 prints the files in the same order as they occur in the directory of the specified volume. If you omit the file type for a file specification, RT–11 assumes LST.

Description

You can use the SPOOL or the QUEUE utility to manage the printing process, freeing your terminal for your use while your file is printing. See the *Introduction to RT–11* for information on using the PRINT command with SPOOL. See the chapters on these utilities (QUEUE or SPOOL) in the *RT–11 System Utilities Manual* for more information.

KMON assigns precedence to SPOOL for any PRINT commands, so be careful if you run both QUEUE and SPOOL.

If QUEUE is not running, some PRINT options are invalid (as noted below). 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 or SPOOL so that another job will not intermix output on that device.

Specifying a Date Argument to Print Options

Some of the options accept a date as an argument. The format for specifying the date is:

[dd][:mmm][:yy]

where:

dd specifies the day (a decimal integer in the range 1–31).

mmm specifies the first three characters of the name of the month.

yy specifies the year (a decimal integer in the range 73–99).

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), RT–11 uses the values from the current system date. For example, if you specify only the year 90 and the current system date is May 4, 1991, RT–11 uses the date 4:MAY:90. If the current 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 at 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.)

Options

/BEFORE[:date]

Prints only those files created before the specified date. If no date is specified, the current system date is used. The following command prints all MAC files on DU0: created before April 21, 1991:

```
.PRINT/BEFORE:21:APR:91 DU0:*.MAC
```

/COPIES:value

Prints more than one copy of the file. The meaningful range of values for the decimal *value* argument is from 1 to 32 (1 is the default). The following command, for example, prints three copies of the file REPORT.LST on the printer:

```
.PRINT/COPIES:3 REPORT
```

/DATE[:date]

Prints only those files with the specified creation date. If no date is specified, the current system date is used. The following command prints all MAC files created on April 21, 1991:

```
.PRINT/DATE:21:APR:91 DK:*.MAC
```

/DELETE

Deletes a file after it is printed. 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 if you want to be asked about each file to be printed and deleted. The following example prints PROG1.BAS on the printer and then deletes it from DU1:

```
.PRINT/DELETE DU1:PROG1.BAS
```

/[NO]FLAGPAGE:value

(Valid only with SPOOL or QUEUE) /FLAGPAGE prints banner pages for each file being printed, where *value* specifies the number of banner pages you want for each file.

- When SPOOL or both SPOOL and QUEUE are running, and you specify a *value* with the /FLAGPAGE:value option, that value overrides any value specified previously by the SET SP FLAG=value command.

  If you do not specify a *value*, the value is set by the SET SP FLAG=value command. The /NOFLAGPAGE option inhibits flag pages under all circumstances.

- When only the QUEUE package is running, the default number of banner pages printed when you use the /FLAGPAGE:value option is determined by the default number of banner pages set with the QUEMAN /P option. If the default set with the /P option is 0, the default for /FLAGPAGE:value is 1. If the QUEMAN /P option is not used, the default is /NOFLAGPAGE.

The banner page 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 the Queue Package chapter in the *RT–11 System Utilities Manual*.

If you are using the SPOOL utility, you can use the /FLAGPAGE:value option to override the default number of banner pages set with the SET SP FLAG=value command.

The following example prints three banner pages for each file in the command line:

```
.PRINT/FLAGPAGE:3 PROG1.MAC,PROG1.LST,PROG1.STB
```

/NOFLAGPAGE suppresses the printing of banner pages. Use this option only if you are running QUEUE. /NOFLAGPAGE is useful if you have previously set QUEMAN's /P option to create banner pages each time a job is output (see preceding note). The default setting is /NOFLAGPAGE, unless you specify otherwise with the QUEMAN /P option.

/INFORMATION

/INFORMATION changes the severity level of the error message that displays when not all the input files you specified are found. If you do not use /INFORMATION, RT–11 displays an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, RT–11 displays an informational message to tell you which files it cannot find, but execution continues.

In the following example, RT–11 prints input files FILE1.TXT and FILE3.TXT. However, since RT–11 is unable to find DL0:FILE2.TXT, RT–11 displays a message to inform you:

```
.PRINT/INFORMATION DL0:(FILE1,FILE2,FILE3).TXT
?PIP-I-File not found DL0:FILE2.TXT
```

/[NO]LOG

/LOG lists on the terminal the names of the files that are printed by the current command. Normally, RT–11 displays 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:
```

/NOLOG prevents a list of the files copied from being displayed on the terminal. You can use this option to suppress the log when you use a wildcard in the file specification.

/NAME:[dev:]jobname

Specifies a job name for the files you want printed. You can use up to six alphanumeric characters for the job name. If you send the files to a mass storage volume, RT–11 copies each input file to a separate output file with the same file name and type as the input file. The job name is used only in the JOBNAME field of the banner page. The handler for the output device must be loaded in memory (see the LOAD command description).

The following example sends the files FILE1.LST, FILE2.LST, and FILE3.LST to files of the same name on DU1:. The job name JOB5 appears in the JOBNAME field of the banner page:

```
.PRINT/NAME:DU1:JOB5 FILE1, FILE2, FILE3
```

The files from this example reside on DU1: as JOB5.JOB.

/NEWFILES

Prints 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 *
Files copied:
DK:OUTFIL.LST   to LP:
DK:REPORT.LST   to LP:
```

/OUTPUT:dev:[filespec]

Specifies the output device for the PRINT command. The default device for *dev* is DK. The default output for the PRINT command without the /OUTPUT option is the printer.

/PRINTER

Forces files to be copied directly to the printer if QUEUE is running. This option causes PIP to copy the file to the printer, which bypasses QUEUE processing, but not SPOOL processing.

/PROMPT

(Valid only with QUEUE) Continues a command string on subsequent lines. When you use /PROMPT, you can enter file specifications on subsequent lines directly to QUEMAN, as described in the chapter on QUEMAN in the *RT–11 System Utilities Manual*. 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

Makes RT–11 request confirmation from you before it performs the operation. /QUERY is useful on operations that involve wildcards, when you may not be sure which files RT–11 selected for an operation.

If you specify /QUERY in a PRINT command line that also contains a wildcard in the file specification, the confirmation messages that display on the terminal replace the log messages that would normally appear.

You must respond to a query message by typing Y (or any string beginning with Y) RETURN to initiate execution of an operation. RT–11 interprets any other response to mean NO; it does not perform the specified operation. The following example uses /QUERY:

```
.PRINT/QUERY *.LST
Files copied:
DK:OUTFIL.LST  to LP:? N
DK:REPORT.LST  to LP:? Y
```

/SINCE[:date]

Prints only those files created on or after the specified date. If no date is specified, the current system date is used.

The following command prints all MAC files on DU0 created on or after April 21, 1991:

```
.PRINT/SINCE:21:APR:91 DU0:*.MAC
```

/WAIT

Useful if you have a single-disk system, initiates the PRINT operation, but then pauses and waits for you to mount the volume that contains the files you want to print.

When RT–11 pauses, it displays *Mount input volume in <device>; Continue?*. Mount the input volume and type Y RETURN to continue the PRINT operation. Type N RETURN or press CTRL/C to abort the operation and return control to the monitor.

After RT–11 completes the PRINT operation, RT–11 displays the following message prompting you to mount the system volume:

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

Mount the system volume and type Y RETURN. If you type any other response, RT–11 continues to prompt you to mount the system volume until you type Y.

The following command line prints ERREX.MAC from DU0:

```
.PRINT/WAIT DU0:ERREX.MAC
Mount input volume in DU0:; Continue? Y
Mount system volume in DU0:; Continue? Y
```

PROTECT

(PIP)

Protects a file so you cannot delete it until you remove the protection (see the UNPROTECT command).

Format

PROTECT *filespec[s]*

Command Options

```
⎧ ⎡ /BEFORE[:date] ⎤ ⎫
⎪ ⎣ /SINCE[:date]  ⎦ ⎪
⎨  /DATE[:date]      ⎬
⎩  /NEWFILES         ⎭
  /EXCLUDE
  /INFORMATION
  /[NO]LOG
  /QUERY
  /SETDATE[:date]
  /SYSTEM
  /WAIT
```

Parameter

filespec[s]

Specifies the file or files you want to protect. You can explicitly specify up to six file specifications if you separate the files with commas. You can also use wildcards in the file specifications.

Description

Some options accept a date as an argument. The format for specifying the date is:

[dd][:mmm][:yy]

where:

dd specifies the day (a decimal integer in the range 1–31).

mmm specifies the first three characters of the name of the month.

yy specifies the year (a decimal integer in the range 73–99).

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), RT–11 uses the values from the current system date. For example, if you specify only the year 90 and the current system date is May 4, 1991, RT–11 uses the date 4:MAY:90. If the current 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 at 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.)

Options

/BEFORE[:date]

Protects only those files created before the specified date. If no date is specified, the current system date is used.

The following command protects all MAC files on DK created before March 20, 1991:

```
.PROTECT/BEFORE:20:MAR:91 *.MAC
Files protected:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

/DATE[:date]

Protects only those files with a certain creation date. If no date is specified, the current system date is used.

The following command protects all MAC files on DK that were created on March 20, 1991:

```
.PROTECT/DATE:20:MAR:91 *.MAC
Files protected:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

/EXCLUDE

Protects all the files on a device except the ones you specify. The following command, for example, protects all files on DU1 except SAV files and SYS files:

```
.PROTECT/EXCLUDE DU1:*.SAV
?PIP-W-No .SYS action
Files protected:
DU1:ABC.OLD
DU1:AAF.OLD
DU1:COMB.
DU1:MERGE.OLD
```

/INFORMATION

Changes the severity level of the error message that displays when not all of the input files you specified are found. If you do not use /INFORMATION, RT–11 displays an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, RT–11 displays an informational message to tell you which files it cannot find, but execution continues.

In the following example, the input files FILE1.TXT and FILE3.TXT are protected. However, since RT–11 is unable to find DL0:FILE2.TXT, RT–11 displays a message to inform you:

```
.PROTECT/INFORMATION DL0:(FILE1,FILE2,FILE3).TXT
?PIP-I-File not found DL0:FILE2.TXT
```

/[NO]LOG

/LOG lists on the terminal a log of the files protected by the current command. This is the default mode of operation when you use wildcards in the file specification. Note that if you specify /LOG, RT–11 does not ask you for confirmation before execution proceeds. Use both /LOG and /QUERY to invoke logging and querying.

/NOLOG prevents a list of files being protected from printing on the terminal.

/NEWFILES

Protects only the files that have the current system date. The following example protects the files created today:

```
.PROTECT/NEWFILES DU1:*.BAK
Files protected:
DU1:MERGE.BAK
```

/QUERY

Requests confirmation from the system before it protects each file. This option is particularly useful on operations that involve wildcards, when you may not be completely sure which files the system will select for the operation. Note that specifying /LOG eliminates the automatic query; you must specify /QUERY with /LOG to retain the query function. Respond to a query message by typing Y (or any response beginning with Y) RETURN to initiate execution of an operation. RT–11 interprets any other response to mean NO; it does not perform the specified operation.

The following example shows querying. Only the file DU1:AAF.MAC is protected:

```
.PROTECT/QUERY DU1:*.*
Files protected:
DU1:ABC.MAC   ? N
DU1:AAF.MAC   ? Y
DU1:MERGE.FOR ? N
```

/SETDATE[:date]

Causes RT–11 to put the date you specify on all files it protects. If you specify no date, the current system date is used. If the current system date is not set, the system places zeros in the directory entry date position. Normally, RT–11 preserves the existing file creation date when it protects a file.

The following example protects three files and changes their dates to the current system date:

```
.PROTECT/SETDATE DU1:*.FOR
Files protected:
DU1:ABC.FOR
DU1:AAF.FOR
DU1:MERGE.FOR
```

/SINCE[:date]

Protects only those files created on or after the specified date. If no date is specified, the current system date is used.

The following command protects all MAC files on DU1 that were created on or after April 21, 1991:

```
.PROTECT/SINCE:21:APR:91 DU1:*.MAC
Files protected:
DU1:A.MAC
DU1:B.MAC
DU1:C.MAC
```

/SYSTEM

Use this option if you need to protect system (SYS) files and you use wildcards in the file type. If you omit this option, RT–11 files are excluded from the preceding type of PROTECT operation and a message is displayed on the terminal to remind you of this. However, if you specify SYS files in your PROTECT operation, you do not need to use the /SYSTEM option.

This example protects all files on DU1 with the file name MM, including SYS files:

```
.PROTECT/SYSTEM DU1:MM.*
Files protected:
DU1:MM.MAC
DU1:MM.OBJ
DU1:MM.SAV
DU1:MM.SYS
```

If you specify a SYS file type, you do not need to specify the /SYSTEM option.

/WAIT

Initiates the PROTECT operation but then pauses for you to mount the volume that contains the files you want to protect. This option is useful if you have a single-disk system.

When RT–11 pauses, it displays *Mount input volume in <device>; Continue?*, where <device> specifies the device into which you mount the volume. Mount the volume and type Y RETURN. Type N RETURN or press CTRL/C to abort the operation and return control to the monitor.

When the operation completes, RT–11 displays the *Continue?* message again. Mount the system volume and type Y RETURN. If you type any other response, RT–11 prompts you to mount the system volume until you type Y. RT–11 then displays the monitor prompt.

The following example protects the file FILE.MAC on an RL02 disk:

```
.PROTECT/WAIT DL0:FILE.MAC
Mount input volume in DL0:; Continue? Y
Mount system volume in DL0:; Continue? Y
```

R

(KMON)

Loads and runs a program from the system volume or from the specified file-structured device. See also FRUN, RUN, SRUN, and V/VRUN.

Format

R *filespec*

Parameter

filespec

Specifies the program to be executed. The default device is SY, and the default file type is SAV.

Description

The R command is unlike the RUN command in one way: the default location for the program run by the R command is the SY device, while the default location for the program run by the RUN command is the DK device. Use the R command only with background jobs, including privileged jobs, under a mapped monitor. (Use FRUN to execute a foreground job under an unmapped multi-job monitor or a mapped monitor.)

The R and the RUN commands are alike in two ways:

- They use the bitmap in the core control block (CCB) to load jobs.
- With each command, you can specify any random-access device as the location of the utility to be run.

The following command loads and executes MYPROG.SAV from device SY:

```
.R MYPROG
```

You can use the R command to execute a background virtual job under a mapped monitor. The R command:

- Creates a virtual memory partition for the job.
- Creates a region 0 and window 0 definition block.
- Sets up the user-mapping registers.

REENTER

(KMON)

Starts the program at its reentry address (the start address minus 2).

Format

REENTER

Description

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 issued two CTRL/Cs to interrupt those programs.

If you issue the REENTER command and it is not valid, the message *?KMON-F-Invalid command* is displayed. You must start that program with an R or RUN command. Note that if SET EXIT NOSWAP is in effect, you may be unable to reenter the program.

In the following example, the directory program (DIR) lists the directory of DK: on the line printer. Pressing CTRL/C twice interrupts the listing and returns you to the monitor. REENTER starts DIR at its reentry address, and DIR prompts for a line of input:

```
.R DIR RET
*LP:=DK:*.*
CTRL/C
CTRL/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 recommences execution.

REMOVE

(KMON)

Removes a device name from the monitor's system tables and, under a mapped monitor, can be used to remove a global region.

Format

$$\textbf{REMOVE} \quad \left\{ \begin{array}{l} \textit{xx:[,xx:...]} \\ \textit{yy[,yy...]} \\ \textit{region[,region...]} \end{array} \right\}$$

Parameters

xx:

Specifies a device name to be removed from the system tables. Any global region attached to that device is automatically eliminated. If no device name is found, the monitor returns an error message.

yy

Specifies a device name or global region. If a device name is specified, that device name is removed from the system tables. If not a device name, the monitor looks for a global region of that (2-character) name, and if found, eliminates it. If the device name or global region is not found, the monitor returns an error message. (mapped monitors only)

region

Specifies a 1-to-6-character global region. If found, that global region is eliminated. If the global region is not found, the monitor returns an error message. (mapped monitors only)

Description

You can use the REMOVE command to remove a device name from the system tables or to remove a global region or both.

Removing a Device Name from the System Tables

Using the REMOVE command does not change the monitor disk image; it only modifies the system tables of the monitor currently in memory. 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 command (STRTxx.COM) file.

You cannot remove a loaded device or any of the following handlers:

SY (the handler for the system device)
BA (the BATCH handler)
MQ (the message queue handler)
TT (the terminal handler)

If you attempt to remove a device that does not exist in the running monitor's system table, RT–11 displays an error message.

Use the SHOW DEVICES command to display on the terminal a list of devices that are available on your system.

Issuing a REMOVE Before an INSTALL

- 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.
- If you COPY or RENAME a handler, you should also issue a REMOVE to ensure that the system will use the latest copy of the handler. For example, if you take a handler, such as SDH.SYS, and RENAME it to SD.SYS, issue a REMOVE and an INSTALL after you change the handler to ensure that the system uses the new version of the handler:

  ```
  .COPY SDH.SYS SD.SYS
  .UNLOAD SD
  .REMOVE SD
  .INSTALL SD
  ```

  In this situation, ignore any error message you may get from issuing the UNLOAD and REMOVE commands.
- Apply this same rule (using REMOVE and INSTALL) to any newly installed handler. Otherwise, RT–11 may use an old copy of the handler rather than the newly installed one.

The following command removes the line printer handler and the logical-disk handler from the system:

```
.REMOVE  LP:,LD:
```

Removing a Global Region from Extended Memory

When RT–11 is running under a mapped monitor, you can use the REMOVE command to remove a global region from extended memory and to return the memory allocated to that global region to the free memory list.

Some regions are not removable; for example:

```
IOPAGE
KERNEL
MCA
```

Any attempt to eliminate a global region attached to a job returns an error message.

RENAME

(PIP)

Assigns a new name to an existing file.

Format

RENAME *in-filespec[s]* *out-filespec*

Command Options

```
/BEFORE[:date]
/SINCE[:date]
/DATE[:date]
/NEWFILES
/INFORMATION
/[NO]LOG
/[NO]PROTECTION
/QUERY
/[NO]REPLACE
/SETDATE[:date]
/SYSTEM
/WAIT
```

Parameters

in-filespec[s]

Specifies the file specifications or combination of file specifications to be renamed.

out-filespec

Specifies the new name or acts as a model for the new names if it is a wildcard specification. For example, the following command tells PIP how to rename MAC files. The command changes the MAC extension to BAK but leaves the file names the same:

```
.RENAME *.MAC *.BAK
```

Description

You can specify up to six input file specifications but only one output file.

Features of the RENAME Command

- 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, RT–11 deletes the existing file, unless you use the /NOREPLACE option to prevent this.

- RT–11 requires you to use the /SYSTEM option to rename system files when you use a wildcard in a file type. However, if you specify a SYS file type, you do not need to specify the /SYSTEM option 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.
- 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.

Specifying a Date as an Argument to an Option

Some options accept a date as an argument. The format for specifying the date is:

[dd][:mmm][:yy]

where:

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

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), RT–11 uses the values from the current system date. For example, if you specify only the year 90 and the current system date is May 4, 1991, RT–11 uses the date 4:MAY:90. If the current 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 at 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.)

Options

/BEFORE[:date]

Renames only those files created before the specified date. If no date is specified, the current system date is used.

The following command renames all MAC files on DU1 created before April 21, 1991:

```
.RENAME/BEFORE:21:APR:91 DU1:*.MAC DU1:*.BAK
Files renamed:
DU1:A.MAC to DU1:A.BAK
DU1:B.MAC to DU1:B.BAK
DU1:C.MAC to DU1:C.BAK
```

/DATE[:date]

Renames only those files with a certain creation date. If no date is specified, the current system date is used.

The following command renames all MAC files created on March 20, 1991 to BAK files:

```
.RENAME/DATE:20:MAR:91 DK:*.MAC *.BAK
 Files renamed:
DK:A.MAC  to DK:A.BAK
DK:B.MAC  to DK:B.BAK
DK:C.MAC  to DK:C.BAK
```

/INFORMATION

Changes the severity level of the error message that displays when not all of the input files you specified are found. If you do not use /INFORMATION, RT–11 displays an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, RT–11 displays an informational message to tell you which files it cannot find, but execution continues.

In the following example, the input files FILE1.TXT and FILE3.TXT are renamed. However, since RT–11 is unable to find DL0:FILE2.TXT, RT–11 displays a message to inform you:

```
.RENAME/INFORMATION DL0:(FILE1,FILE2,FILE3).TXT
?PIP-I-File not found DL0:FILE2.TXT
```

/[NO]LOG

/LOG lists on the terminal the files that were renamed by the current command. Normally, RT–11 displays 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 DU0:(A*.MAC  *.FOR)
Files renamed:
DU0:ABC.MAC    to DU0:ABC.FOR
DU0:AAF.MAC    to DU0:AAF.FOR
```

/NOLOG prevents a list of the files that are renamed from appearing on the terminal.

/NEWFILES

Renames only those files that have the current date. This is a convenient way to access all new files after a session at the computer.

/[NO]PROTECTION

/PROTECTION protects a file 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 unprotects a file.

/QUERY

Causes RT–11 to request confirmation before it performs the operation. /QUERY is useful on operations that involve wildcards, when you may not be sure which files RT–11 selected for the operation.

You must respond to a query message by typing Y (or any string beginning with Y) RETURN to initiate execution of an operation. RT–11 interprets any other response to mean NO; it does not perform the specified operation. The following example demonstrates querying:

```
.RENAME/QUERY DU0:(PIP1.SAV PIP.SAV)
Files renamed:
DU0:PIP1.SAV   to  DU0:PIP.SAV   ? Y
```

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 display on the terminal replace the log messages that would normally appear.

/[NO]REPLACE

/REPLACE 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, RT–11 deletes that duplicate file when it performs the RENAME operation.

/NOREPLACE 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 DU0:TEST.SAV DU0:DUP.SAV
?PIP-W-Output file found, no operation performed DU0:TEST.SAV
```

/SETDATE[:date]

Causes RT–11 to put the date you specify on all files it renames. If you specify no date, the current system date is used. If the current system date is not set, the system places zeros in the directory entry date position. Normally, RT–11 preserves the existing file creation date when it renames a file.

The following example renames files and changes their dates to the current system date:

```
.RENAME/SETDATE DU1:(*.FOR *.OLD)
Files renamed:
DU1:ABC.FOR    to DU1:ABC.OLD
DU1:AAF.FOR    to DU1:AAF.OLD
DU1:MERGE.FOR  to DU1:MERGE.OLD
```

/SINCE[:date]

Renames all files on a specified device created on or after a specified date.

The following command renames only those MAC files on DK created on or after February 24, 1991:

```
.RENAME/SINCE:24:FEB:91 *.MAC *.BAK
Files copied:
DK:A.MAC  to DK:A.BAK
DK:B.MAC  to DK:B.BAK
DK:C.MAC  to DK:C.BAK
```

/SYSTEM

Includes SYS files in a wildcard RENAME operation. Use this option if you need to rename system (SYS) files and you use wildcards in the input file type. If you omit this option in the preceding type of operation, SYS files are excluded from the RENAME operation and a message is displayed on the terminal to remind you of this. However, if you specify a SYS file type, you do not need to specify the /SYSTEM option to rename system files.

This example renames all files on DU1: with the file name MM, including SYS files, to MX files:

```
.RENAME/SYSTEM DU1:MM.* DU1:MX.*
 Files renamed:
DU1:MM.MAC  to DU1:MX.MAC
DU1:MM.OBJ  to DU1:MX.OBJ
DU1:MM.SAV  to DU1:MX.SAV
DU1:MM.SYS  to DU1:MX.SYS
```

If you specify a SYS file type, you do not need to specify the /SYSTEM option to rename system files.

/WAIT

Initiates the RENAME operation but then pauses and waits for you to mount the volume that contains the files you want to rename. The /WAIT option is useful if you have a single-disk system.

When RT–11 pauses, it displays *Mount input volume in <device>; Continue?*. Mount the input volume and type Y RETURN to continue the RENAME operation. Type N RETURN or press CTRL/C to abort the RENAME operation and return control to the monitor.

After RT–11 completes the RENAME operation, RT–11 displays the following message prompting you to mount the system volume:

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

Mount the system volume and type Y RETURN. If you type any other response, RT–11 prompts you to mount the system volume until you type Y.

The following command line renames PRIAM.TXT to NESTOR.TXT. PRIAM.TXT is on a DU0 disk:

```
.RENAME/WAIT/NOLOG SY:(PRIAM.TXT NESTOR.TXT)
Mount input volume in DU0:; Continue? Y
Mount system volume in DU0:; Continue? Y
```

RESET

(KMON)

Initializes several background system tables and does a general cleanup of the background area.

Format

RESET

Description

The RESET command accepts no options or arguments.

RESET causes RT–11 to:

- Purge all open input/output channels.
- Reset the terminal ring buffers and command buffers.
- Initialize the user program memory area.
- Reset CTRL/O (this enables terminal output).
- Clear locations 40–53.
- Reset the KMON (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 foreground or system jobs.

The following example uses the RESET command before running a program:

```
.RESET
.R MYPROG
```

RESUME

(KMON)

Continues execution of a foreground or system job from the point at which a SUSPEND command was issued.

Format

RESUME *[jobname]*

Parameter

jobname

If you have system-job support enabled on your monitor, *jobname* specifies the name of the foreground or system job you wish to resume. This name can be a logical job name.

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 since there is only one foreground job.

Description

When you issue the RESUME command, the foreground or system job enters any completion routines that were scheduled while the job was suspended. The RESUME command is not valid with single-job monitors.

Examples

1. This command resumes execution of the foreground job that is currently suspended:

```
.RESUME
```

2. This 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 job that you loaded with FRUN or SRUN using /PAUSE.

RUN

(KMON)

Loads and runs a program from the default device DK, or from the specified file-structured device. See also FRUN, R, SRUN, and V/VRUN.

Format

```
RUN   filespec   [ { input-list,   [output-list] } ]
                 [ { argument                    } ]
```

Parameters

filespec

Specifies the program to be executed. RT–11 assumes a SAV file type for the executable file, which can reside on any RT–11 block-replaceable device. The default device is DK.

argument

Specifies a single argument in CSI (Command String Interpreter) that can be passed to the program to be run.

input-list

Specifies a list of input file specifications in CSI or CCL format that can be passed to the program to be run.

output-list

Specifies a list of output file specifications in CSI CCL format that can be passed to the program to be run.

Description

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 an unmapped multi-job monitor or a mapped 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.

Optional CSI Parameters

You can use the RUN command to specify a CSI (Command String Interpreter) formatted input line for a user or a system utility program. To do so, specify the needed argument or argument list as parameters of the RUN command. RT–11 automatically converts the argument(s) you specify into a format that CSI accepts. See the *RT–11 System Utilities Manual* for a description of the CSI command format.

Examples

1. This command executes MYPROG.SAV, which is stored on device DU1:

   ```
   .RUN DU1:MYPROG
   ```

2. This command executes the directory program (DIR) to obtain a complete listing of the directory on DU1. The command then prints the listing on the printer. *DU1:*.** is the input parameter and *LP:/E* is the output parameter of the directory program:

   ```
   .RUN DIR DU1:*.* LP:/E
   .
   ```

   The previous command has the same effect as the following:

   ```
   .RUN DIR
   *LP:/E=DU1:*.*
   * CTRL/C
   .
   ```

3. This command displays an alphabetical listing of the directory on DK. The filespec specifies the DIR program, and the argument specifies the /A option:

   ```
   .RUN DIR /A
   ```

   Note: 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

(KMON)

Writes memory areas in memory-image format to the file and device that you specify.

Format

SAVE *filespec* *[memory-locations]*

Parameters

filespec

Specifies the file to be saved on a block-replaceable device. If you do not specify a file type, RT–11 uses SAV.

memory-locations

Specifies memory locations to be saved. Specify these in the following format:

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_{10}. If the location(s) you specify make a block that is less than 256 words, RT–11 saves additional words to make a 256-word block.

RT–11 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).

To change the contents of the header block of the save image, use the Deposit command to alter them and the SAVE command to save the correct values.

Area	**Location**
Start address	40
Stack	42
JSW	44
USR address	46
High address	50
Fill characters	56

If you change the values of these locations, you must reset the locations to their default values. For more information concerning these locations, see the *RT–11 Volume and File Formats 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.

Examples

1. 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
   ```

2. 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
   ```

SET

(KMON, device handler)

Defines or changes characteristics of device or pseudodevice handlers and the values of some system parameters.

Format

SET *entity* *condition[,condition,...]*

Parameters

entity

Specifies a physical-device name, a pseudodevice name, or a system parameter.

- Device name

  Specifies the device handler (physical or pseudo) whose characteristics you want to modify. See Table 2 for the physical and pseudodevice handlers operated on by this command. See Table 3–1 for a list of the standard RT–11 permanent device names. The colon (:) after each device name is optional.

- System parameter

  Specifies a system parameter in the resident monitor that you may need to modify. This allows you to control certain monitor features, such as error handling or program swapping. The SET commands are valid for all monitors unless otherwise specified.

  The SET commands for a monitor, such as SET USR NOSWAP, set or clear status words in the memory copy of the resident monitor.

condition

Specifies how you want the *entity* (device handler or system parameter) modified.

If you set more than one condition in a single SET command, separate the conditions with commas; for example, the following command turns on the single-line command editor, sets it to keypad editing functionality, and lets you use the RECALL command with a mapped monitor:

```
SET SL RECALL,KED,ON
```

Description

SET commands let you enable, disable, and change such features as:

- Command-line editor (SET SL)
- Default editor (SET EDIT)
- Error handling (SET ERROR)

- Program swapping on exit (SET EXIT)
- IND and KMON handling of command and control files (SET KMON)

How SET Commands Work

SET commands modify monitor parameters and device-handler parameters. The monitor SET commands affect only the copy of the monitor in memory while most of the handler SET commands affect only the copy of the handler on the system disk. The copy of the handler in memory is generally not affected. This means, for a handler SET command to take effect, you usually have to issue the SET command, unload the handler (if it is loaded), and then load the new "set" modified copy of the handler. When a SET command modifies the memory copy of the handler, it is noted in that SET command description.

With the exception of the SET TT, SET USR, and SET system-parameter commands, the SET command locates the device-handler file *SY:device.SYS* and permanently modifies it.

Specifying a Device Handler

Specify a device handler with the SET command as follows:

```
SET XXn: condition
```

Where *XX* is the standard two-letter mnemonic for the device handler name, *n* is the unit number (0–7), and *condition* is how you want the handler modified.

Specifying an Extended-Unit Device Handler

Specify an extended-unit device handler (one for which you have generated extended device-unit support) in the format:

```
SET Xnn: condition
```

where *X* is a one-letter mnemonic (A–Z), *nn* is an octal unit number (0–77), and *condition* is how you want the handler modified. The format X: is valid and means X0:.

D is the one-letter name for the DU (MSCP) device handler with extended device-unit support. *L* is the one-letter name for the LD (logical disk) device handler with that support (see SET DUx and SET LDx).

You can optionally use two-letter physical device names for units 0 through 7 of an extended device-unit handler; for example: DU7. Therefore, you do not have to modify any existing command files or procedures that refer to only the first 8 units of an extended device-unit handler. You must, however, specify units 10 through 77_8 only by the 1-letter convention and octal number (for example: DU10).

Ensuring a System Set the Way You Want It

SET commands permanently modify device handlers (except where noted); this means that the conditions remain set even across a reboot.

However, 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 and to ensure you always have the settings you want when you start your computer, include the appropriate SET commands in your system's STRTxx.COM file.

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 a mapped monitor, any device handlers modified will be of the form %%X.SYS.

Remember: if a handler (except for TT or the handler specified in SET dd: [NO]WRITE commands) 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.

Summary of SET Command Options

Table 2 lists all the SET command options with their types and a brief explanation of what each one is. The type is indicated by a T for TEMPORARY and a P for PERMANENT. The temporary type lasts until you turn off your computer. The permanent type remains when you reboot your computer.

The SET command descriptions following the table define each command separately.

Table 2: SET Options

Option	Type	Parameter the Option Sets
CLI	T	Command-Line Interpreter (DCL, CCL, UCL, UCF)
CLOCK	T	Frequency of system clock
DL:	P	DL disk handler (RL01/RL02)
DM:	P	DM disk handler (RK06/RK07)
DU[n]:	P	DU handler (MSCP disk or diskette)
DW:	P	DW disk handler (RDxx)
DX[n]:	P	DX diskette handler (RX01)
DY[n]:	P	DY diskette handler (RX02)
DZ[n]:	P	DZ diskette handler (RX50 for the Professional 300 series processor)
EDIT	T	Default editor
EL:	T	Error Logger under the single-job monitors
ERROR	T	Severity level of failure that aborts a command file
EXIT	T	Use of SWAP.SYS
FORTRA	T	Default FORTRAN compiler

Table 2 (Cont.): SET Options

Option	Type	Parameter the Option Sets
KMON	T	Command or control-file processing in response to the @ command.
LD[n]:	P	LD (logical-disk) subsetting handler
LP:	P	LP (line-printer) handler
LS:	P	LS (serial-printer) handler
MODE	T	FBMON$ bit in configuration offset of monitor
MM:	P	MM magtape handler (TJU16/TU45)
MS:	P	MS magtape handler (TK25/TS11/TSV05/TSU05/TU80)
MT:	P	MT magtape handler (TM11/TMA11/TS03/TE16)
MU:	P	MU magtape handler (TMSCP devices)
NC:	P	Ethernet handler (Professional 300 series processors)
NL:	P	NL (null) handler
NQ:	P	Ethernet handler (Q-bus processors)
NU:	P	Ethernet handler (UNIBUS processors)
RK:	P	Disk handler (RK05)
RUN	T	Use of the completely virtual environment
SD:	P	Symbolic Debugger handler
SL:	P	SL (single-line) command editor
SP:	P	SP (SPOOL) handler
ST:	P	Pseudodevice ST (Symbol Table) handler for use with SD:
TERMINAL *or* TT:	T	TT (Terminal) handler
UB:	P	UB (UNIBUS mapping register) handler
USR	T	USR (user service routine) in the monitor
VM:	P	VM (virtual memory) handler
WILD	T	Monitor recognition of wildcards in a file specification
XC:	P	Communication port handler for the Professional 300 series processor
XL:	P	Communication port handler for PDP–11 processors

SET CLI

(KMON)

Enables or disables one or more command-line interpreters.

Format

SET CLI *interpreter1,interpreter2,...*

Command Summary

SET CLI
[NO]**CCL**
[NO]**DCL**
[**NO**]**UCF**
[NO]**UCL**

Description

CLI means *Command-Line Interpreter*. The SET CLI command enables/disables four command-line interpreters: CCL, DCL, UCF, and UCL.

- CCL is the *Concise Command Language* interpreter. This interpreter allows you to run utility programs and pass them command strings on a single line. See the *RT–11 System Utilities Manual* for a description of this interpreter.
- DCL is the *Digital Command-Line* interpreter that understands the commands in this manual. See Part I of this manual for a description of DCL.
- UCF is the *User Command First* interpreter through which you can change or redefine DCL or CCL commands. See the *Introduction to RT–11* for a description of how to use the UCL and UCF interpreters.
- UCL is the *User Command Linkage* interpreter through which you can define your own commands.

The CSI command-string interpreter is not affected by the SET CLI command.

The Order of Command Processing

The RT–11 monitor passes commands to the interpreters in the order described below. Once the appropriate interpreter translates a command, the monitor executes it. If no interpreter can translate a command, you will get an error message.

- If the UCF command interpreter has been enabled, it is the first interpreter to look at a command. If the UCF interpreter translates the command, the monitor executes it; if the interpreter cannot translate the command, the monitor passes the command to the DCL interpreter (if it is enabled).

  Since the UCF interpreter looks at all commands first, UCF command processing slows down the processing of all commands that are not UCF commands.

- If you have not enabled the UCF interpreter, the DCL command interpreter is the first interpreter to look at a command (if it is enabled).

  If the DCL interpreter translates the command, the monitor executes it; if the interpreter cannot translate the command, the monitor passes the command to the CCL interpreter (if it is enabled).

- The CCL command interpreter looks at a command only if the DCL interpreter cannot translate it.

  If the CCL interpreter translates the command, the monitor executes it; if the interpreter cannot translate the command, the monitor passes the command to the UCL interpreter (if it is enabled).

- The UCL command interpreter is always passed a command last when no other interpreter can translate the command. This means that UCL commands never slow down the interpretation of DCL or CCL commands.

  If the UCL interpreter translates the command, the monitor executes it; if the interpreter cannot translate the command, the UCL interpreter displays an error message.

Using the UCL and UCF Interpreters

The DCL and CCL interpreters are always present in the operating system, but to use the UCL and UCF interpreters, you must follow the procedures described in the Defining Your Own Commands chapter in the *Introduction to RT–11*.

Displaying the Status of Your Command-Line Interpreters

You can display the status of CLI by issuing the SHOW CONFIGURATION command. A line in the display shows the status of all CLI options.

Enabling/Disabling Interpreters

Once you include the UCL and UCF interpreters in your operating system, you can then use the SET CLI command to enable or disable them. You might want to enable or disable one or another interpreter to adapt your system to a particular set of commands and to remove the system overhead for the interpreter(s) you do not need.

Making a SET CLI Command Automatic

The SET CLI command affects your running operating system's memory only; it does not affect the image of the operating system on file on your system device. This means that when you reboot your operating system, the command-line interpreter reverts to its defaults.

To make a SET CLI command automatic (that is, active when you reboot), you can make it the last part of your system startup procedure in your STRTxx.COM file.

Note

- The SET CLI command has no provision to safeguard against turning off all the command-line interpreters that it controls (DCL, CCL, UCL, and UCF).
- If you accidentally disable all your command-line interpreters, reboot your system to reenable them.
- If you disable DCL, you can no longer use any DCL command except RUN and R. This means, to reboot your system, you must turn it off and turn it on again or reboot it in some other way appropriate for your processor, since the BOOT command will no longer function. Another way to boot some processors is the following:
  - — Press Break. The system will display a number such as 144356 and prompt you with an @ (the at sign).
  - — At the @ prompt, type in the number 773000G (the memory location of the BOOT ROM) and press RETURN.
- If you place the SET CLI NODCL command in your STRTxx.COM file, place this SET command near the end of the file after any other DCL commands. This will:
  - — Keep the other DCL commands functioning during your startup procedure.
  - — Allow you to use a double CTRL/C to exit out of the STRTxx.COM file while it is running but before it turns off DCL, thus further allowing you to change a feature with DCL commands.

Interpreter

[NO]DCL

DCL enables the DCL interpreter; if it is already on, it is left on. No confirmational or informational message is displayed. SET CLI DCL is the default.

NODCL disables the DCL interpreter; if the interpreter is already disabled, it is left off. No informational message is displayed.

[NO]CCL

CCL enables the CCL interpreter; if it is already on, it is left on. No confirmational or informational message is displayed. SET CLI CCL is the default.

NOCCL disables the CCL interpreter; if the interpreter is already disabled, it is left off. No informational message is displayed.

[NO]UCL

UCL enables the UCL interpreter; if it is already on, it is left on. No confirmational or informational message is displayed. SET CLI UCL is the default.

NOUCL disables the UCL interpreter; if the interpreter is already disabled, it is left off. No informational message is displayed.

[NO]UCF

UCF enables the UCF interpreter; if it is already on, it is left on. No confirmational or informational message is displayed.

NOUCF disables the UCF interpreter; if the interpreter is already disabled, it is left off. No informational message is displayed. SET CLI NOUCF is the default.

SET CLOCK

(KMON)

Sets the frequency of the system clock.

Format

SET CLOCK *frequency*

Frequency

50

Causes the system to use a 50-Hz line frequency clock rate.

60

Causes the system to use a 60-Hz line frequency clock rate (the default).

SET DL

(DL)

Sets the DL disk handler (RL01/RL02 disks).

Format

SET DL *condition*

Condition

CSR=n

Uses *n* as the CSR address for the DL handler. The valid range for *n* is 160000 to 177570_8.

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

[NO]SUCCES

(Valid only with error-logging support) SUCCES, the default, logs successful I/O transfers as well as errors when running the Error Logger. NOSUCCES logs only I/O errors.

VECTOR=n

Uses *n* as the vector address for the DL handler. The valid range for *n* is 100_8 to 474_8. This option enables you to set a special vector value in the device handler without having to modify and reassemble the handler source code.

SET DM

(DM)

Sets the DM disk handler (RK06/RK07 disks).

Format

SET DM *condition*

Condition

CSR=n

Uses *n* as the CSR address for the DM handler. The valid range for *n* is 160000 to 177570_8.

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

[NO]SUCCES

(Valid only with error-logging support) SUCCES, the default, logs successful I/O transfers as well as errors when running the Error Logger. NOSUCCES logs only I/O errors.

VECTOR=n

Uses *n* as the vector address for the DM handler. The valid range for *n* is 100_8 to 474_8.

SET DU

(DU)

Sets the DU device handler (MSCP disks and diskettes).

Format

SET DU *condition*

Condition

CSR=n

Modifies the DU device handler to use *n* as the CSR address for the first controller. The valid range for *n* is 160000 to 177570_8.

This option enables you to set a special CSR value in the device handler itself without having to modify and reassemble the handler source code.

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

[NO]SUCCES

(Valid only with error-logging support) SUCCES, the default, logs successful I/O transfers as well as errors when running the Error Logger. NOSUCCES logs only I/O errors.

VECTOR=n

Uses *n* as the vector address for the first port of the DU MSCP controller. The valid range for *n* is 100_8 to 474_8.

SET DUx

(DU)

Defines how an MSCP disk is partitioned.

Format

SET DUx *condition*

Condition

CSRy=n

(Valid only if the handler is assembled for multiple ports) Modifies the DU handler to use *n* as the CSR address for DU controller *y*. The variable *y* is an integer in the range 0–3, with 0 being the default value. *SET CSR0=n* is equivalent to *SET CSR=n*.

PART=n

Defines the partition of a disk on which device unit *x* resides. The variable *n* is an integer in the range 0–255, depending on the size of the disk device (each partition is 64K blocks). The default for *n* is 0.

SET DUx PART=n assigns DUx to disk partition *n*. (See the *RT–11 Volume and File Formats Manual* for more information on disk partitioning.)

PORT=n

Assigns DUx to MSCP port *n*. The variable *x* has a default range of 0–7, with 0 being the default value. If you have assembled the DU handler for extended units, the range is from 0 to 64_{10}. You specify an extended DU device unit in the format *Dxx*, where *xx* is a number from 0 through 77_8.

The variable *n* (specifying the port) is an integer in the range 0–3, with 0 being the default value. (See the (dev_book) for more information on using multiple ports with MSCP devices.)

UNIT=n

Defines which unit plug number to access when device unit *x* is specified. The variable *n* is an integer in the range 0–251. The default setting defines *n* equal to *x*.

VECy=n

(Valid only if the handler is assembled for multiple ports) Modifies the DU handler to use *n* as the vector address for controller *y*. The variable *y* is an integer in the range 0–3, with 0 being the default value. *SET DU VEC0=n* is equivalent to *SET DU VECTOR=n*.

SET DW

(DW)

Sets the DW disk handler (RDxx disks).

Format

SET DW *condition*

Condition

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

SLOT=n

(Valid only with the PRO Expander) Assigns a copy of the DW handler to the controller for the PRO Expander RD5x card slot *n*. See the following description for an explanation of how to do this.

[NO]SUCCES

(Valid only with error-logging support) SUCCES, the default, logs successful I/O transfers as well as errors when running the Error Logger. NOSUCCES logs only I/O errors.

[NO]WCHECK

WCHECK verifies output to DW disks by reading data after writing it to the disk. NOWCHECK does not verify output to DW disks.

[NO]WRITE

WRITE write enables DW disk drive unit 0. NOWRITE write locks DW disk drive unit 0. NOWRITE affects both the memory copy and the disk file of the handler.

Description

Support for the PRO Expander

The PRO Expander lets you add one additional RD5x controller and hard disk to a CTI Bus-based processor configuration. RT–11 supports the PRO Expander in the following manner.

The DW handler supports only a single device connected to a single controller. Therefore, to support the second controller and hard disk, you must COPY the DW handler to another handler name (for example, DV) and use a SET command to assign the created handler (DV) to the second device.

Copy the DW handler to DV and use the SET DV SLOT=n command to support the PRO Expander. (Using device name DV is optional; you can use any device name not recognized by or assigned to RT–11.) Use the SLOT=n option only to support a PRO Expander and only with the created handler.

Use the SLOT=n option to identify the PRO Expander RD5x controller card slot in the Professional's backplane. The six card slots are numbered 0 through 5, from front to back. The first RD5x controller is typically in slot 0. The second RD5x controller typically goes in slot 5. Assuming that arrangement, issue the following commands to support the PRO Expander:

```
.COPY SY:DW[X].SYS SY:DV[X].SYS
.SET DV SLOT=5
.INSTALL DV
```

SET DX

(DX)

Sets the DX diskette handler (RX01 diskettes).

Format

SET DX *condition*

Condition

CSR=n

Uses *n* as the CSR address for the DX handler. The valid range for *n* is 160000 to 177570_8. This option enables you to set a special CSR value in the device handler without having to modify and reassemble the handler source code.

CSR2=n

(Valid only if you have generated support for a second controller) Uses *n* as the CSR address for the DX handler. The valid range for *n* is 160000 to 177570_8.

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

[NO]SUCCES

(Valid only with error-logging support) SUCCES, the default, logs successful I/O transfers as well as errors when running the Error Logger. NOSUCCES logs only I/O errors.

VECTOR=n

Uses *n* as the vector address for the DX handler. The valid range for *n* is 100_8 to 474_8. This option enables you to set a special vector value in the device handler without having to modify and reassemble the handler source code.

VEC2=n

(Valid only if you have generated support for a second controller) Uses *n* as the vector address for the DX handler. The valid range for *n* is 100_8 to 474_8.

SET DXx

(DX)

Sets the specified unit of the DX diskette handler (RX01 diskettes).

Format

SET DXx *condition*

Condition

[NO]WRITE

WRITE, the default, write enables DX unit *x*. NOWRITE write locks DX unit *x*. NOWRITE affects both the memory copy and the disk file of the handler.

SET DY

(DY)

Sets the DY diskette handler (RX02 diskettes).

Format

SET DY *condition*

Condition

CSR=n

Uses *n* as the CSR address for the DY handler. The valid range for *n* is 160000 to 177570_8. This option enables you to set a special CSR value in the device handler itself without having to modify and reassemble the handler source code.

CSR2=n

(Valid only if you have generated support for a second controller) Uses *n* as the CSR address for the DY handler. The valid range for *n* is 160000 to 177570_8.

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 2 through 127. The default value for *n* is 8.

[NO]SUCCES

(Valid only with error-logging support) SUCCES, the default, logs successful I/O transfers as well as errors when running the Error Logger. NOSUCCES logs only I/O errors.

VECTOR=n

Uses *n* as the vector address for the DY handler. The valid range for *n* is 100_8 to 474_8. This option enables you to set a special vector value in the device handler without having to modify and reassemble the handler source code.

VEC2=n

(Valid only if you have generated support for a second controller) Uses *n* as the vector address for the DY handler. The valid range for *n* is 100_8 to 474_8.

SET DYx

(DY)

Sets the specified unit of the DY diskette handler (RX02 diskettes).

Format

SET DYx *condition*

Condition

[NO]WRITE

WRITE, the default, write enables DY unit *x*. NOWRITE write locks DY unit *x*. NOWRITE affects both the memory copy and the disk file of the handler.

SET DZ

(DZ)

Sets the DZ diskette handler (RX50 diskettes for Professional 300 series).

Format

SET DZ *condition*

Condition

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

[NO]SUCCES

(Valid only with error-logging support) SUCCES, the default, logs successful I/O transfers as well as errors when running the Error Logger. NOSUCCES logs only I/O errors.

SET EDIT

(KMON)

Defines an editor as the default editor.

Format

SET EDIT *default-editor*

Default-Editor

EDIT

Defines EDIT.SAV to be the default text editor for the monitor EDIT command.

KED

Defines KED.SAV to be the default text editor; requires a VT100-compatible terminal (default with distributed unmapped monitors). For information on the keypad editor, see the *PDP–11 Keypad Editor User's Guide*.

KEX

Defines KEX.SAV to be the default text editor; requires a VT100-compatible (default with distributed mapped monitors). Use KEX as you would KED. For information on the keypad editor, see the *PDP–11 Keypad Editor User's Guide*.

TECO

Defines TECO.SAV to be the default text editor. (TECO is not distributed or supported by Digital.)

SET EL

(KMON)

Sets the Error Logger for single-job monitors.

Format

SET EL *condition*

Condition

[NO]LOG

LOG turns on the Error Logger if the EL handler is loaded and begins logging errors in an EL handler internal buffer. The Error Logger can be turned off by issuing SET EL NOLOG or by unloading the EL handler. RT–11 returns to SET EL NOLOG after a reboot.

NOLOG, the default, turns off the Error Logger.

PURGE

Clears the internal error-log buffer in a single-job monitor. This command is valid only if the Error Logger has been enabled with the SET EL LOG command.

SET ERROR

(KMON)

Controls the severity level of failure that will abort a command file. See the Using Indirect Command Files chapter in the *Introduction to RT–11* for an explanation of using the SET ERROR command with command files.

Format

SET ERROR *condition*

Condition

ERROR
Aborts command files if error, fatal, or unconditional errors occur. This is the default.

FATAL
Aborts command files if fatal or unconditional errors occur.

NONE
Continues processing.

UNCONDITIONAL
Aborts command files if an unconditional error occurs.

WARNING
Aborts command files if warning, error, fatal, or unconditional errors occur.

Description

The following table shows the errors for each setting that abort a command file. The *X* means the command file will abort if it has that error.

	Errors			
Setting	**W (Warning)**	**E (Error)**	**F (Fatal)**	**U (Unconditional)**
Warning	X	X	X	X
Error		X	X	X
Fatal			X	X
Unconditional				X
None				X

SET EXIT

(KMON)

Selects whether or not SWAP.SYS is used on job exit.

Format

SET EXIT *condition*

Condition

[NO]SWAP

SWAP, the default, enables the writing of a program to the swap blocks when the program terminates. This allows you to restart the program with the REENTER command.

NOSWAP disables the writing of a program to the swap blocks when the program terminates. This may prevent you from being able to reenter a program; however, it allows considerably better performance when using slower media (such as RX01 and RX02).

SET FORTRA

(KMON)

Sets the default FORTRAN compiler.

Format

SET FORTRA *compiler*

Compiler

F4

Causes the FORTRAN IV compiler to be called by the COMPILE/FORTRAN, EXECUTE/FORTRAN, and FORTRAN commands. This is the default.

F77

Causes the FORTRAN–77 compiler to be called by the COMPILE/FORTRAN, EXECUTE/FORTRAN, and FORTRAN commands. Which FORTRAN–77 compiler is called is determined by the monitor: F77XM.SAV for mapped monitors, F77.SAV for unmapped monitors.

SET KMON

(KMON)

Sets KMON to recognize IND (INDirect control-file processor) control files.

Format

SET KMON *condition*

Condition

[NO]IND

IND causes IND.SAV to interpret the expression @*filespec* as a control file to be executed.

To avoid confusion, you can use the *IND filespec* syntax to execute control files and leave the @ command set to run command files (see the *RT–11 IND Control Files Manual*.

NOIND, the default, causes KMON to interpret the expression @*filespec* as a command file to be executed.

The syntax *$@filespec* forces NOIND interpretation even when SET IND is in effect.

SET LD

(LD)

Sets the logical-disk subsetting handler.

Format

SET LD *condition*

Condition

CLEAN

Verifies and corrects, if necessary, all current logical-disk assignments by checking them against the files on volumes that are accessible. This command is most useful after you have moved or removed files on a volume, or after you have removed a volume from a device. If a logical-disk file has moved, the new location is noted so that you can continue to use that logical disk. If you have deleted a logical-disk file or the volume containing a logical-disk file is no longer mounted, the logical-disk assignment is disconnected. In the case of a volume that you have removed, the disconnect is only temporary. You can reestablish the assignment when you remount the volume by using the SET LD CLEAN command again. The commands SQUEEZE and BOOT automatically perform the SET LD CLEAN operation.

EMPTY

Dismounts all logical-disk assignments.

Format

SET LDx *condition*

Condition

[NO]WRITE

(Used during disk subsetting) WRITE, the default, defines logical-disk unit x as being write enabled (read/write access allowed). The value x must be an integer in the range 0 through 7 unless you have SYSGENed your system for extended-unit support. WRITE affects both the memory copy and the file of the handler.

If you have extended-unit support, the range for x is 0 through 32_8. The syntax for specifying an extended device unit is as follows:

```
Lxx:
```

where L is the LD handler and xx is the unit number from 0 to 32_8.

NOWRITE, used during disk subsetting, defines logical-disk unit x as being write locked (read-only access allowed). The value x must be an integer in the range 0 through 7, unless you have SYSGENed your system for extended-unit support. With extended-unit support the range of *x* is 0 through 32_8. NOWRITE affects both the memory copy and the file of the handler.

See the description of the logical disk utility in the *RT–11 System Utilities Manual* for more information.

SET LP

(LP)

Sets the line-printer handler.

Format

SET LP *condition*

Condition

[NO]BIT8

BIT8 sets the LP handler to pass the eighth bit unaltered, but does not guarantee the eighth bit will arrive at the output device unaltered. The following can alter or strip the eighth bit:

- Some RT–11 utilities in ASCII mode
- Some RT–11 KMON commands, such as PRINT
- Hardware device initialization program options
- Hardware device straps

NOBIT8, the default, sets the LP handler to force the eighth bit to zero, but does not guarantee the eighth bit will arrive at the output device as zero.

[NO]CR

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 default mode.

NOCR, the default, 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 RETURN.

CSR=n

Modifies the line printer handler to use *n* as the Control and Status Register (CSR) address for the line printer controller. The valid range for *n* is 160000 to 177570_8. 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.

[NO]CTRL

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.)

NOCTRL, the default, ignores nonprinting control characters.

ENDPAG=n

Sets the LP handler to append *n* form feed characters to the end of each file sent to the printer. If ENDPAG=0, the LP handler does not append a form feed character to the end of each file sent to the printer. The default value for *n* is 0.

[NO]FORM

FORM, the default, 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.

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.

[NO]FORM0

FORM0, the default, issues a form feed before printing block 0.

NOFORM0 does not issue a form feed before printing block 0.

[NO]HANG

HANG, the default, 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.

NOHANG generates an immediate error if the line printer is not ready. Valid only when you are running under a monitor with device timeout support (included during system generation). The default is HANG.

[NO]LC

LC allows RT–11 to send lowercase characters to the printer. Use this condition if your printer has a lowercase character set.

NOLC, the default, translates characters in lowercase to uppercase before printing. This is the normal mode.

LENGTH=n

Defines the length of page; causes the line printer to use *n* as the number of lines for each page. The default number of lines for each page is 66. Use this option with the NOFORM and SKIP=n settings.

SKIP=n

Defines the number of lines to skip at page break; 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.

[NO]TAB

TAB sends TAB characters to the line printer.

NOTAB, the default, expands TAB characters by sending multiple spaces to the line printer.

VECTOR=n

Modifies the line printer handler to use *n* as the vector of the line printer controller. The valid range for *n* is 100_8 to 474_8. 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.

WIDTH=n

Sets the line printer width to *n*, where *n* is a decimal integer between 30 and 255 inclusive. RT–11 ignores any characters that print past column *n*. The default is 132.

SET LS

(LS)

Sets the serial-printer handler.

Format

SET LS *condition*

Condition

[NO]BIT8

BIT8 sets the LS handler to pass the eighth bit unaltered, but does not guarantee the eighth bit will arrive at the output device unaltered. The following can alter or strip the eighth bit:

- Some RT–11 utilities in ASCII mode
- Some RT–11 KMON commands, such as PRINT
- Hardware device initialization program options
- Hardware device straps

NOBIT8, the default, sets the LP handler to force the eighth bit to zero, but does not guarantee the eighth bit will arrive at the output device as zero.

[NO]CR

CR, the default, sends carriage return characters to the line 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.)

NOCR prevents RT–11 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 RETURN.

CSR=n

Modifies the line printer handler to use *n* as the Control and Status Register (CSR) address for the printer controller. The valid range for *n* is 160000 to 177570_8. 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.

[NO]CTRL

CTRL, the default, passes all characters, including nonprinting control characters, to the printer. Use this condition to pass the bell character to the LA180 printing terminal.

NOCTRL ignores nonprinting control characters.

ENDPAG=n

Sets the LS handler to append *n* number of form feed FF characters to the end of each file sent to the printer. If ENDPAG=0, the LS handler does not append a form feed FF character to the end of each file sent to the printer. The default value for *n* is 0.

[NO]FORM

FORM, the default, 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.

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.

[NO]FORM0

FORM0, the default, issues a form feed before printing block 0.

NOFORM0 does not issue a form feed before printing block 0.

[NO]GRAPH

GRAPH sets the LS handler to output all characters regardless of the width of the line. Use this command for plotters, such as the LVP–16 or any other graphics devices that use the LS handler. When you SET LS GRAPH, any width limit you specified using the SET LS WIDTH command is ignored.

NOGRAPH, the default, sets the LS handler to ignore all characters that do not fit on a line. Excess characters are not printed. The length of the line is determined by SET LS WIDTH=nnnnn.

[NO]HANG

HANG, the default, 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.

NOHANG generates an immediate error if the printer is not ready. Valid only when you are running under a monitor with device timeout support (included during system generation).

[NO]LC

LC, the default, allows RT–11 to send lowercase characters to the printer. Use this condition if your printer has a lowercase character set.

NOLC translates lowercase characters to uppercase before printing.

LENGTH=n

Defines length of page, causing the printer to use *n* as the number of lines for each page. The default number of lines for each page is 66. Use this option with the NOFORM and SKIP=n settings.

LINE=n

(Valid with multiterminal support only) Modifies the serial-printer handler to use line *n* of a multiterminal monitor as the serial port.

The SET command takes no effect until the handler is LOADed or .FETCHed. (See the LOAD command description in this manual and the .FETCH programmed request description in the *RT–11 System Macro Library Manual*.)

[NO]MTTY

(Valid with multiterminal support only) MTTY selects use of multiterminal as the interface to the serial printer. NOMTTY selects use of a DL interface for which it was built as the interface to the serial printer.

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.

SPEED=n

Sets the printer to run at baud *n*, where *n* can be any of the following baud rates:

```
 50  1200
 75  1800
110  2000
134  2400
150  3600
200  4800
300  9600
600  19200
```

This command is valid only when running on a Professional 300 series system.

You must specify a value for *n* in this command. If you do not use this command, the printer runs at 4800 baud.

[NO]TAB

TAB sends TAB characters to the printer.

NOTAB, the default, expands TABs by sending multiple spaces to the printer.

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.

WIDTH=n

Sets the printer to width *n*, where *n* is a decimal integer between 30 and 255 inclusive. RT–11 ignores any characters that print past column *n*. The default is 132.

SET MODE

(KMON)

Sets or clears the obsolete FBMON$ bit in the configuration offset of the monitor fixed area to indicate the obsolete SJ monitor characteristics (Only for compatibility with RT–11 pre-version 5.6, user-written handlers that require the FBMON$ bit set to OFF).

Format

SET MODE *[NO]SJ*

Condition

[NO]SJ

SET MODE SJ turns the FBMON$ bit off.

SET MODE NOSJ, the default, turns the FBMON$ bit on.

Description

RT–11, before Version 5.6, used the FBMON$ bit to distinguish between the removed SJ monitor and the FB and XM monitors. Since RT–11 no longer supports the SJ monitor, RT–11 no longer references that bit. By default, therefore, FBMON$ is set to ON, and setting the bit to OFF is no longer a valid representation of the operating system.

If you have a handler that checks the FBMON$ bit to make sure there is only one user or to check for a condition other than where monitor data is located, then you can issue the SET MODE SJ command before you install your handler. However, after you install the handler, you should also issue the SET MODE NOSJ command to prevent any program from accidentally thinking the SJ monitor is running the operating system.

If you have a handler that checks the FBMON$ bit of the SJ monitor to determine where monitor data is located, then your handler will not work on the FB, SB, XB, ZB, XM, and ZM monitors.

For more information on the FBMON$ bit, (Bit 0) in the CONFIG Word, see the *RT–11 System Internals Manual*.

SET MM

(MM)

Sets the MM magtape handler (TJU16/TJE16/TU45/TJU77 magtapes).

Format

SET MM *condition*

Condition

CSR=n

Uses *n* as the CSR address for the MM handler. The valid range for *n* is 160000 to 177570_8. This option enables you to set a special CSR value in the device handler without having to modify and reassemble the handler source code.

DEFALT=9

Sets the default settings for 9-track magtape. The 9-track defaults are:

DENSE=809
ODDPAR

DENSE=density

Sets density for the 9-track tape handler. Do not alter the density setting within a volume. A density setting of 1600 bpi (bits per inch) automatically sets parity to odd. The valid density settings for 9-track tape are:

800 (or 809) bpi
1600 bpi

The default is 800.

[NO]ODDPAR

ODDPAR, the default, sets parity to odd for 9-track tape. Digital recommends this setting.

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.

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

VECTOR=n

Modifies the MM handler to use *n* as the vector address. The valid range for *n* is 100_8 to 474_8. This option enables you to set a special vector value in the handler without having to modify and reassemble the handler source code.

SET MS

(MS)

Sets the MS magtape handler (TK25/TS11/TSV05/TSU05/TU80 magtapes).

Format

SET MS *condition*

Condition

CSR=n

Modifies the MS handler to use *n* as the CSR address for the first controller. The valid range for *n* is 160000 to 177570_8. This option enables you to set a special CSR value in the device handler itself without having to modify and reassemble the handler source code. *SET CSR=n* is equivalent to *SET CSR1=n*.

CSRx=n

(Valid only if the handler is assembled for multiple ports) Modifies the MS handler to use *n* as the CSR address for MS controller *x*; *x* can be from 2 to 8.

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

VECTOR=n

Modifies the MS handler to use *n* as the vector address for the first controller. *SET MS VECTOR=n* is equivalent to *SET MS VEC1=n*. This option enables you to set a special vector value in the handler without having to modify and reassemble the handler source code.

VECx=n

(Valid only if the handler is assembled for multiple ports) Modifies the MS handler to use *n* as the vector address for controller *x*; *x* can be from 2 to 8.

SET MT

(MT)

Sets the MT magtape handler (TM11/TMA11/TS03/TE16 magtapes).

Format

SET MT *condition*

Condition

CSR=n

Uses *n* as the CSR address for the MT handler. The valid range for *n* is 160000 to 177570_8. This option enables you to set a special CSR value in the device handler without having to modify and reassemble the handler source code.

DEFALT=n

Sets defaults for 7- or 9-track magtape; *n* can be 7 or 9.

DENSE=n

Sets density for 7- or 9-track magtape; *n* can be 200, 556, 800, 807, or 809. The default is 800 bpi (bits per inch).

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 Device Handlers Manual*.

DUMP

Writes bytes to 7-track magtape at 800 bpi (bits per inch).

[NO]ODDPAR

ODDPAR, the default, sets parity to odd for 7- or 9-track magtape. Digital recommends this setting.

NOODDPAR sets parity to even for 7- or 9-track magtape. Digital does not recommend this setting for normal operation and provides it only for compatibility with other systems.

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

VECTOR=n

Modifies the MT handler to use *n* as the vector address. The valid range for *n* is 100_8 to 474_8. This option enables you to set a special vector value in the handler without having to modify and reassemble the handler source code.

SET MU

(MU)

Sets the MU magtape handler (TMSCP magtape devices).

Format

SET MU *condition*

Condition

CSR=n

Modifies the MU device handler to use *n* as the CSR address for the first controller. The valid range for *n* is 160000 to 177570_8. This option enables you to set a special CSR value in the device handler itself without having to modify and reassemble the handler source code.

RETRY=n

Specifies *n* as the number of times MU attempts to recover from an error. The value *n* must be an integer in the range 1–127. The default value for *n* is 8.

[NO]SUCCES

(Valid only with error-logging support) SUCCES, the default, modifies the MU handler to log successful I/O transfers as well as errors when the Error Logger is running.

NOSUCCES, the default, modifies the MU handler to not log successful I/O transfers as well as errors when the Error Logger is running.

VECTOR=n

Modifies the MU device handler to use *n* as the vector for the first controller. The valid range for *n* is 100 to 474_8. This option enables you to set a special vector value in the device handler without having to modify and reassemble the handler source code.

SET MUx

(MU)

Sets the specified unit of the MU magtape handler (TMSCP magtape devices).

Format

SET MUx *condition*

Condition

CSRy=n

(Valid only if you create the specified MU controller handler during system generation) Modifies the MU device unit *y* to use *n* as the CSR address for a controller. *y* can be 0, 1, 2, or 3. *SET MU CSR0=n* is the same as *SET MU CSR=n*.

DEFALT

Returns magtape unit *n* to the default density of 6250 bits for each inch. Valid only with 9-track TMSCP magtapes and sets only the specified magtape unit.

DENSE=n

Sets density for only 9-track TMSCP magtapes and for only the specified magtape unit. *n* can be 1600 or 6250. Setting DENSE=1600 specifies 1600 bpi (phase encoded). Setting DENSE=6250 specifies 6250 bpi (group code recording).

PORT=n

(Valid only if the handler is assembled for multiple ports) Defines which port to access when magtape unit *x* is specified. The variable *n* is an integer in the range 0–3 and corresponds to the variable *y* used with the SET MU CSRy=n and SET MU VECy=n commands. By default, the port variable *n* is the same as the magtape unit *x*.

UNIT=n

Defines which TMSCP unit number to access when magtape unit *x* is specified. The variable *n* is an integer in the range 0–255. The default value for *n* is 0. Since each TMSCP unit requires a controller, this modification is optional and is used only if you have multiple TMSCP devices and you have modified the unit number on the device front panel.

VECy=n

(Valid only if you create the specified MU controller handler during system generation) Modifies the MU device unit *y* to use *n* as the vector for a controller. *y* can be 0, 1, 2, or 3. *SET MU VECTOR=n* is the same as *SET MU VEC0=n*.

SET NC

(NC)

Sets the Ethernet handler for the Professional 300 series processor.

Format

SET NC *condition*

Condition

SHOW

Displays the station physical address for the DECNA controller.

SET NL

(NL)

Sets the NULL handler.

Format

SET NL *condition*

Condition

[NO]SYSGEN

SYSGEN causes NL to match the current monitor's SYSGEN parameters. Issue SET NL SYSGEN if you receive an error message indicating conflicting SYSGEN options when you attempt to install NL.

NOSYSGEN inhibits NL from installing under the current monitor. SET NL NOSYSGEN is useful when ST has been assigned the logical-device name NL and you want to inhibit installation of the unnecessary NL.

[NO]TOY

SET NL TOY, the default, runs the NL handler installation code, if the RT–11 system is booted with the NLx.SYS handler present on the system disk or if the NL handler is removed and installed with DCL commands.

The NL installation code transfers date and time information from the KDJ11–E Time-Of-Year (TOY) clock to the resident monitor. You might want to include this command in your STRTxx.COM file.

SET NL NOTOY prevents the NL handler from transferring the current date and time to the resident monitor.

Description

The KDJ11–E clock keeps the current date and time; both the PDP–11/93 and 11/94 processors contain this clock.

While the SET NL TOY transfers date and time information from the KDJ11–E clock to the resident monitor, the SETUP command sets the date and time.

Issue the SETUP [CLOCK] DATE and SETUP [CLOCK] TIME commands to set the specified date and time for both the monitor time and the time-of-year clock in a PDP–11/93 and 11/94 processor.

SET NQ

(NQ)

Sets the Ethernet handler for Q-bus processors.

Format

SET NQ *condition*

Condition

CSR=n

Uses *n* as a CSR address for the DEQNA or DELQA controller. The valid range for *n* is 160000 to 177570_8. This option enables you to set a special CSR value in the device handler itself without having to modify and reassemble the handler source code.

SHOW

Displays the station physical address for the DEQNA or DELQA controller.

VECTOR=n

Uses *n* as a vector address for the DEQNA or DELQA controller.

SET NU

(NU)

Sets the Ethernet handler for UNIBUS processors.

Format

SET NU *condition*

Condition

CSR=n

Modifies the NU handler to use *n* as the CSR address. The valid range for *n* is 160000 to 177570_8.

SHOW

Displays the station physical address for the DEUNA or DELQA controller and also identifies the controller type.

VECTOR=n

Modifies the NU handler to use *n* as the vector address.

SET RK

(RK)

Sets the RK device handler (RK05 devices).

Format

SET RK *condition*

Condition

CSR=n

Uses *n* as the CSR address for the RK handler. The valid range for *n* is 160000 to 177570_8. This option enables you to set a special CSR value in the device handler itself without having to modify and reassemble the handler source code.

RETRY=n

Specifies the number of times the device handler attempts to recover from an I/O error. The value *n* must be an integer in the range 1 through 127. The default value for *n* is 8.

[NO]SUCCES

(Valid only with error-logging support) SUCCES, the default, logs successful I/O transfers as well as errors when running the Error Logger. NOSUCCES logs only I/O errors.

VECTOR=n

Uses *n* as the vector address for the RK handler. The valid range for *n* is 100_8 to 474_8. This option enables you to set a special vector value in the device handler without having to modify and reassemble the handler source code.

SET RUN

(VBGEXE)

Enables or disables VBGEXE (the virtual run utility) to automatically load and execute programs in a completely virtual environment. See also V/VRUN.

Format

SET RUN *condition*

Condition

[NO]VBGEXE

VBGEXE enables VBGEXE to automatically load and execute programs under extended-memory monitors if:

- VBGEXE.SAV is on the system disk.
- The program's VBGEX$ (200) bit is set in the job's $JSX.

NOVBGEXE disables VBGEXE to automatically run programs.

SET SD

(SD)

Configures the DBG–11 SD handler.

Format

SET SD *condition*

Description

See the *DBG–11 Symbolic Debugger User's Guide* for the listing and description of the SET SD commands.

SET SL

(SL)

Sets the single-line (SL) command editor.

Format

SET SL *condition*

Description

The command-line editor enables you to edit, store, recall, and manipulate command lines. With this editor, you can:

- Edit a command line with keypad functionality.
- Use the RECALL command and arrow keys to recall previous commands.
- Save and restore commands in the sequence you want.

See the *Introduction to RT–11* for a tutorial explanation of how to use the single-line command editor and for a description of its major functions. Type HELP SL or HELP RECALL for on-line help descriptions.

Note

Changes you make with the SET SL command are permanent. This means, for example, if you SET SL LET and then reboot your computer, the LET utility is still enabled. See the description of the SET command for further information about permanent and temporary SET commands.

Condition

ASK

Allows the single-line editor to determine the type of terminal you are using, so SL can use the proper escape sequences. SL displays on the terminal the type of console you are using and the type of support SL will provide for that terminal. If SL does not support the terminal you are using, SL displays an error message.

[NO]KED

KED turns on keypad editing functionality. In keypad mode, you use the keypad to edit the command line as you would use the keypad with the KED editor. Keypad editing also allows you to recall a cycle of commands in the sequence you specify. Keypad editing is available under all RT–11 monitors.

NOKED, the default, turns off keypad editing.

KMON

Enables use of SL by KMON but not by user programs. SET SL KMON sets SL to edit only a monitor command line; that is, input following the monitor prompt

(.) and monitor input requests. Use SET SL KMON as an alternative to SET SL ON. Digital recommends that you use SET SL KMON.

[NO]LEARN

LEARN leaves help text on the screen and scrolls input below it. This setting helps you learn to use the single-line editor by allowing you to lock the help display on the top half of your screen. You can use the bottom of your screen to type command lines and display terminal output. After you issue the commands SET SL ON and SET SL LEARN, type the PF2 key to display the help frame and lock it on the screen.

NOLEARN, the default, unlocks the help display and allows it to scroll off the screen, so you can use the entire screen to display terminal input and output. The SET SL OFF command performs an automatic SET SL NOLEARN command.

OFF

Disables the single-line editor. This is the default.

ON

Enables the single-line editor for use both by KMON and by user programs and loads the SL handler. This means SL edits all lines typed at the terminal. This must be the last option in a multiple option SET command; for example, SET SL VT100, LEARN, ON. See SET SL KMON.

[NO]LET

LET lets you define your own commands; that is, you can define a character or a program function key (the F keys on the top of your keyboard) to substitute for a command string. See the LET utility description in the *RT–11 System Utilities Manual* for more information. Note that the LET utility is unsupported.

NOLET, the default, disables the LET utility.

[NO]RECALL

RECALL lets you use the RECALL command to recall previously entered commands. However, you can use RECALL only with a mapped monitor.

SET NORECALL, the default, disables the use of the RECALL command.

[NO]SYSGEN

SYSGEN configures the SL handler to match the current monitor's SYSGEN parameters.

NOSYSGEN inhibits SL from installing under the current monitor. This is useful if you have SL on the system disk but do not intend to use it. By not using SL you can save space in extended memory.

[NO]TTYIN

TTYIN enables single-line editing of .TTYIN input; that is, TTYIN lets you edit responses to prompts displayed by the system utilities. However, if special mode or edit mode is set (in bit 12 of the JSW), this command has no effect.

When SET SL TTYIN is enabled, the prompt displays on one line, and your response appears on the following line. (This command allows SL to intercept and edit input requests from .TTYIN. SL always intercepts and edits input requests from .CSIGEN, .CSISPC, and .GTLIN.)

NOTTYIN disables single-line editing of .TTYIN input and causes SL to edit only lines that are requested with .CSIGEN, .CSISPC, and .GTLIN programmed requests.

VTxxx

Tells the single-line editor which type of terminal you are using, so SL can send the proper escape sequences. It is recommended that you use SET SL ASK instead of this command.

This command supports the following terminals:

VT100 (SET SL VT100)
VT102 (SET SL VT102)
VT102 settings can be used for VT200 and above series terminals.

The default setting is SET SL VT100.

WIDTH=n

Allows you to set the line width of the terminal display to n number of characters. n can be from 30 to 132_{10}; the initial value of n is 79.

The maximum allowable width of a line you can input at the terminal is:

```
n - (width of prompt string including monitor prompt) - 1
```

For example, if you issue the command SET SL WIDTH=50, and the prompt consists of only the monitor prompt (.), then the maximum number of characters you can type as input on one line is:

```
50 - 1 - 1 = 48 characters
```

SET SP

(SP)

Sets the spool handler.

Format

SET SP *condition*

Condition

EXIT

EXIT stops SPOOL in a synchronous manner. Use SET SP EXIT to stop SPOOL from within a command file so that the monitor prompt is not returned until all SPOOL activity is terminated.

SET SPx

(SP)

Sets the spool handler for the specified device or for all spooled devices if no device is specified.

Format

SET SPx *condition*

Condition

ENDPAG=n

ENDPAG=n appends *n* terminating form feeds to each file sent to spooled device *x*. If *x* is not specified, ENDPAG=n appends *n* terminating form feeds to files sent to all spooled devices.

SPx ENDPAG=0 suppresses appending terminating form feeds to each file sent to spooled device *x*. If *x* is not specified, ENDPAG=0 suppresses appending terminating form feeds to files sent to all spooled devices.

You can generate end-page support without flag-page support. To do so, include the new SYSGEN conditional SP$EPS=1 to support end pages without necessarily supporting flag pages, when you generate your version of the operating system.

FLAG=n

Specifies the number of flag pages to be prefixed to each file sent to spooled device *x*. If *x* is not specified, SET FLAG causes *n* flag pages to be prefixed to all spooled devices. For RT–11 V5.6, up to six flag pages can be specified for *n*.

PRINT/FLAGPAGE:n, when a value is specified for *n*, overrides this setting. The default for *n* is 2.

[NO]FORM0

FORM0 issues a form feed on spooled device *x* each time SPOOL encounters block 0 of a file to be printed. If *x* is not specified, SET FORM0 issues a form feed on all spooled devices each time SPOOL encounters block 0 of a file to be printed.

NOFORM0, the default, turns off FORM0 mode for spooled device *x*. If *x* is not specified, SET NOFORM0 turns off FORM0 for all spooled devices.

KILL

Aborts output to the *x* spooled device. If *x* is not specified, SET KILL aborts output to all spooled devices.

NEXT

Stops printing the current file on spooled device *x* and proceeds to the next file queued to that device. If *x* is not specified, defaults to device SP0. If no file is queued, the command is ignored.

[NO]WAIT

WAIT interrupts output to spooled device *x*, while output to any other spooled device continues. If *x* is not specified, WAIT interrupts output to all spooled devices.

NOWAIT resumes output to spooled device *x* after output to that device has been interrupted. If *x* is not specified, NOWAIT resumes output to all spooled devices.

[NO]WIDE

WIDE causes SPOOL to generate 132-column flag pages for specified device *x*. If *x* is not specified, SET WIDE generates 132-column flag pages for all spooled devices.

NOWIDE causes SPOOL to generate 80-column flag pages for specified device *x*. If *x* is not specified, SET NOWIDE generates 80-column flag pages for all spooled devices.

SET ST

(ST)

Makes the SYSGEN configuration bits of the ST handler compatible with the currently running monitor.

Format

SET ST *condition*

Condition

SYSGEN

SYSGEN makes the configuration bits of the ST handler compatible with the currently running monitor. Use this command if you copy the ST handler to a different system from the one on which it was created or if you boot a different monitor.

Description

See Chapter 4 of the *DBG–11 Symbolic Debugger User's Guide* for a description of how to create and use the DBG–11 pseudodevice ST (Symbol Table) handler.

SET TT (TERMINAL)

(TT)

Sets the terminal handler. SET TERMINAL can be substituted for SET TT.

Format

SET TT *condition*

Condition

CONSOL=n

(Requires multiterminal support) Changes the background console terminal to the terminal defined as logical terminal *n*. The terminal whose logical-unit number you specify must not be currently attached by the foreground or any system job. You cannot use this setting for a remote line.

[NO]CRLF

(Valid for multi-job monitors only) CRLF, the default, inserts a RETURN/LINE FEED when you attempt to type past the right margin. You can change the margin with the WIDTH command.

When using the single-line (SL) editor, if you SET TT CRLF, you must also set your terminal to wrap. Set your terminal to wrap using the SETUP TERM WRAP command or use the terminal hardware setup feature.

NOCRLF takes no special action at the right margin. RT–11 returns to this condition after a reboot.

[NO]FB

(Valid for multi-job monitors only) FB, the default, treats CTRL/B and CTRL/F (and CTRL/X in monitors that include system job support) as background and foreground program control characters and does not transmit them to your program. The system returns to this condition after a reboot.

NOFB causes CTRL/B and CTRL/F (and CTRL/X in monitors that include system job support) to have no effect. 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.

[NO]FORM

(Valid for multi-job monitors only) FORM indicates that the console terminal is capable of executing hardware form feeds.

NOFORM, the default, simulates form feeds by generating eight line feeds. RT–11 returns to this condition after a reboot.

[NO]PAGE

(Valid for multi-job monitors only) PAGE, the default, interprets CTRL/S as XOFF and CTRL/Q as XON to stop and start terminal output, respectively.

NOPAGE causes CTRL/S and CTRL/Q to have no special meaning.

[NO]QUIET

QUIET disables the echoing of lines from command files.

NOQUIET, the default, enables echoing of command files as they are processed.

[NO]SCOPE

SCOPE, the default for all monitors, echoes DELETE or RUBOUT characters as backspace-space-backspace. Use this mode if your console terminal is a video terminal. 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.

NOSCOPE echoes DELETE or RUBOUT characters by enclosing the deleted characters in backslashes.

[NO]TAB

(Valid for multi-job monitors only) TAB indicates that the console terminal has hardware tab stops.

NOTAB, the default, simulates tab stops every eight positions. Many terminals supplied by Digital have hardware tabs.

WIDTH=n

(Valid for multi-job monitors only) Sets the terminal width to *n*, where *n* is an integer between 30 and 255_{10}; the initial value of *n* is 80. (See SET TT CRLF.)

SET UB

(UB)

Sets the UB handler that supports UNIBUS mapping registers (UMR) hardware in UNIBUS processors.

Format

SET UB *condition*

Condition

[NO]INSTAL

INSTAL, the default, allows installation of the UB handler with valid UNIBUS configurations.

NOINSTAL prevents installation of the UB handler with valid UNIBUS configurations.

PERM=decimal-value

Sets the number of permanent UMRs to reserve for handlers that are not installed during the boot process.

[NO]SERIAL=octal-value

SERIAL=octal-value forces serial satisfaction of outstanding I/O requests for the specified job, where *octal-value* is the job number. Determine the job number by issuing the SHOW JOBS command. Serial satisfaction goes into effect at the next installation of UB (at system boot).

Serialized I/O request satisfaction is often slower than nonserialized satisfaction for the following reasons:

- UB satisfies I/O requests for jobs set to serialized satisfaction at a lower priority than I/O requests for jobs set to nonserialized satisfaction.
- A request that is queued because of insufficient free UMRs is satisfied before any subsequent request is satisfied; I/O requests are always satisfied in the order that they are received. For example, assume UB receives a request requiring three UMRs and only two UMRs are available. That request is queued. Assume the next request received requires only two UMRs. The second request is queued until enough UMRs (three) are free to satisfy the previous request. With nonserialized satisfaction, the second request would have been satisfied when received.

NOSERIAL=octal-value, the default, satisfies outstanding I/O requests for the specified job, where *octal-value* is the job number, as required UMRs are free. I/O requests are satisfied as quickly as possible and not necessarily in serial order.

SET USR

(KMON)

Sets the state of USR (User Service Routine) swapping.

Format

SET USR *condition*

Condition

[NO]SWAP

SWAP, the default, allows the background job to place the USR (User Service Routine) in a swapping state. This setting is not valid for mapped monitors. This is the default mode for unmapped monitors.

NOSWAP locks the USR in memory so that it cannot be swapped.

SET VM

(VM)

Sets the VM (Virtual Memory) handler.

Format

SET VM *condition*

Condition

BASE=nnnnnn

Lets you select the location in memory where block 0 of a virtual disk will begin (the base address). The VM handler uses memory at and above *nnnnnn*, a physical address multiplied by 100_8.

Since the base address is a 22-bit address that must be represented in 16 bits, the bottom six bits (bits 0–5) are always 0. Therefore, when specifying the value *nnnnnn*, use only the top 16 bits of the base address you want. For example, if you want the base address to be 10025600_8, specify 100256 for *nnnnnn*. The default value for *nnnnnn* is 1600_8 under unmapped monitors and 10000_8 under mapped monitors. (The address 10000 is the division between 18- and 22-bit addresses.)

[NO]INSTAL

INSTAL, the default, causes the VM handler to install at the system boot.

NOINSTAL causes an otherwise valid VM handler to not install at the system boot.

SIZE=nnnnnn

Lets you select the memory size in blocks (512 bytes for each block) of the virtual device. If you SET VM SIZE=0, VM allocates all available memory from the SET VM BASE value to the top of physical memory. VM SIZE=0 is the default.

SET WILD

(KMON)

Sets the monitor to recognize wildcards in file specifications.

Format

SET WILD *condition*

Condition

EXPLICIT

Causes RT–11 to recognize file specifications exactly as you type them. If you omit a file name or a file type in a file specification, RT–11 does not automatically replace the missing item with an asterisk (*).

IMPLICIT

Causes RT–11 to interpret missing fields in file specifications as wildcards (asterisks). This is the default.

SET XC

(XC)

Sets the baud rate of the communication-port handler for Professional 300 series processor.

Format

SET XC *condition*

Condition

SPEED=n

Sets the communication port to run at baud *n*, where *n* can be any of the following baud rates:

```
50    1200
75    1800
110   2000
134   2400
150   3600
200   4800
300   9600
600   19200
```

The default baud is 1200.

SET XL

(XL)

Sets the communication-port handler for PDP–11 processors.

Format

SET XL *condition*

Condition

LINE=n

(Valid with multiterminal support only) Modifies the XL handler to use *n* as the multiterminal serial port. The value of *n* can be between 0 and 16; the default value is line 1.

The SET command takes no effect until the handler is LOADed or .FETCHed (See the LOAD command description in this manual and the .FETCH programmed request description in the *RT–11 System Macro Library Manual*).

CSR=n

Uses *n* as the CSR address for the XL handler. The valid range for *n* is 160000 to 177570_8. This option enables you to set a special CSR value in the device handler itself without having to modify and reassemble the handler source code.

[NO]MTTY

(Valid with multiterminal support only) MTTY selects use of multiterminal as the interface to the XL handler. NOMTTY selects use of a DL interface for which it was built as the interface to the XL handler.

VECTOR=n

Uses *n* as the vector address for the XL handler.

SETUP

(SETUP)

Sets hardware characteristics for the terminal, printer, and system clock.

Format

SETUP *[hardware]* *mode[,mode...]*

Parameters

hardware

Specifies one of the following:

- **TERMINAL** or **TT** (the default for printer and terminal modes)

  If you do not specify a hardware parameter, SETUP assumes TERMINAL for all modes except CLOCK modes. For CLOCK modes, SETUP assumes CLOCK.

- **PRINTER** or **LP** or **LS**

  The PRINTER mode is the only mode in which you must specify the *hardware* parameter, which in this case is *printer*.

  If you specify PRINTER, SETUP looks first for device LP and, if not found, then looks for device LS to send the mode to your printer. The Professional 300 series processors support PRINTER modes for only the LA50, LA100, and LQP02 printers.

  Some modes or printer features are incompatible with some arguments. Check your hard-copy terminal or printer user guide. See also Appendix B for a summary listing of the modes that you can use with different printers.

- **CLOCK** (the default for clock modes)

  CLOCK sets modes in your system clock. You need not specify CLOCK to set clock modes in your system clock.

mode

Specifies a hardware feature. Table 3 lists all the modes you can specify with SETUP.

The modes are of five types, depending on the hardware you want to modify and what you want done:

Mode Type	Description
Clock	Sets system clock.
Control	Displays a pointer to help descriptions and modifies some SETUP modes.
National Replacement	Sets national replacement characters for Professional terminals only (see Appendix C for how to use this feature and for descriptions of the national replacement character sets).
Printer	Sets your printer.
Terminal	Sets your terminal.

Most of the control modes apply to Professional 300 series hardware only. Control modes include such features as *help*, *reset*, *save*, and *show*.

You can set several modes in one SETUP command by separating modes with a comma (,). For example, the following command sets three modes for the console terminal:

```
.SETUP SMOOTH,NOCLICK,WRAP
```

You cannot set a mode for a device that does not support that mode; the device will ignore the mode. Refer to your device's user's guide for guidance in selecting proper SETUP files.

Setting Terminal Modes for Different Types of Terminals

- You can set VT200 and VT300 series terminals by using their local menu or by using the SETUP commands.
- You can set VT100 series terminals by using the local set-up mode as described in the appropriate VT100 series user's guide or by using the SETUP commands.
- You can set a Professional 300 series terminal only by using the SETUP commands. Some SETUP modes are valid for only Professional integrated terminals, as indicated.
- You can set a hard-copy terminal by using the particular terminal's set-up procedure or by using the SETUP commands.

Terminal Defaults

The defaults for the VT100, VT200, and VT300 series terminals are listed in the respective terminal user's guide. The defaults for the Professional integrated terminal are indicated in the following mode descriptions.

Description

The SETUP program sets features (modes) of your terminal (video and hard-copy), printer, and system clock. These commands are especially useful in your system start-up command file or an IND control file.

Table 3: Summary of SETUP Modes

Mode	Type	Description
132COLUMNS	Terminal	Displays 132 characters across screen.
80COLUMNS	Terminal	Displays 80 characters across screen.
12HOUR	Clock	Puts the system clock into 12-hour mode (Professional only).
24HOUR	Clock	Puts the system clock into 24-hour mode (Professional only).
480INTERLACE	Terminal	Sets the PRO380 to 480 INTERLACE mode (Professional 380 only).
ANSWERBACK:/text/	Terminal	Places text into the ANSWERBACK buffer (Professional only).
[NO]BCD	Clock	Converts the system clock to BCD [binary] mode (Professional only).
BLOCK	Terminal	Displays the cursor as a block, the default. (Professional only).
[**NO**]**BOLD**	Terminal/Printer	Enables [disables] bold printing. NOBOLD is the default.
CAPS	Terminal	Causes LOCK to function as a VT100 key (Professional only). This is the default.
CLEAR	Terminal/Printer	Clears the screen and performs a soft reset.
[NO]**CLICK**	Terminal	Causes keys to [not] click when pressed (Professional only). CLICK is the default.
CLOCK SHOW	Clock	Displays the current clock settings (Professional only).
COLOR	Terminal	Informs the software that terminal is a color terminal (Professional color terminals only).
COLOR1 ON COLOR2	Terminal	Sets the text to color1 and background to color2; colors are black, blue, cyan, green, magenta, red, white, and yellow (Professional color terminals only).
[**NO**]**COMPOSE**	National Replacement	Enables [disables] compose character key (Professional only).
[NO]**CURSOR**	Terminal	Causes the cursor to be [in]visible (Professional only).

Table 3 (Cont.): Summary of SETUP Modes

Mode	Type	Description
DARK	Terminal	Displays light characters on a dark background.
DATA	National Replacement	Sets NRC language for data processing characters (Professional only).
DATE:dd:mmm:yy	Clock	Sets the system date as specified.
DEFAULTS	Control	Returns the terminal to its factory settings (Professional terminal only).
DENSE	Terminal/Printer	Overlaps two sets of dots for a close approximation of letter-quality printing.
DOWN	Terminal/Printer	Moves the print down 1/2 inch for alignment.
DRAFT	Printer	Sets print to an 8-x-9 dot matrix.
[NO]DST	Clock	Puts the system clock into [non]daylight savings time mode (Professional only).
GENERIC100	Terminal	Sets the reply to a device ID query as a generic VT100.
GRAPHIC	Terminal/Printer	Enters GRAPHIC mode.
HELP	Control	Lists brief descriptions of the SETUP commands.
HORIZONTAL:n	Terminal/Printer	Prints *n* characters for each inch.
[**NO**]**INTERLACE**	Terminal	Enters [exits] INTERLACE mode.
JUMP	Terminal	Scrolls lines across screen as fast as received.
LANDSCAPE	Printer	Sets printer for wider page dimension (LN03 laser printers only).
LANG:xxx	Terminal/Printer	Enables the font and for the specified country code (Professional only).
LETTER	Terminal/Printer	Enables enhanced density and 10 characters per inch.
LIGHT	Terminal	Enables the display of dark characters on a light background.
LISTING	Terminal/Printer	Enables normal density and 16.5 characters per inch.
[**NO**]**MARGINBELL**	Terminal	Enables [disables] bell 8 columns from right margin (Professional only).
MEMO	Terminal/Printer	Enables memo quality printing.

Table 3 (Cont.): Summary of SETUP Modes

Mode	Type	Description
MONO	Terminal	Informs software that the terminal is a black and white terminal (Professional only).
[**NO**]**NEWLINE**	Terminal	Enables [disables] generation of LF when RET is typed.
NORMAL	Terminal/Printer	Turns off terminal and printer character attributes, such as REVERSE, UNDERLINE, and BOLD.
PAGELENGTH:n	Terminal/Printer	Prints *n* lines per page.
PORTRAIT	Printer	Sets the printer for narrow page dimension (LN03 laser printers only).
[NO]**REPEAT**	Terminal	Enables [non]repetition of characters when the key is held down (Professional terminal only).
RESET	Control	Resets the terminal to previously saved characteristics.
[NON]REVERSE	Terminal	Places terminal characters in reverse [normal] video mode.
RETAIN	Control	Splits off unused NRC languages and retains selected language (Professional terminal only).
SAVE	Control	Saves terminal characteristics.
SETCOLOR color [red:value,blue:value, green:value]	Terminal	Customizes text and background colors (Professional 350 and 380 only).
SETCOLOR color FACTORY	Terminal	Sets the specified color to the distributed setting (Professional only).
SHIFT	Terminal	Causes LOCK to function as a typewriter key (Professional only).
SHOW	Control	Shows current settings (Professional only).
SMOOTH	Terminal	Scrolls smoothly and slowly up the screen.
[NO]TABS[:n:n:n]	Terminal/Printer	Sets [clears] tabs at specified columns.
TEXT	Terminal/Printer	Exits GRAPHIC mode.
TIME:hh:mm:ss	Clock	Sets the system time as specified.
TYPE	National Replacement	Sets NRC language for typewriter characters (Professional only).
[**NO**]**UNDERLINE**	Terminal/Printer	Enables [disables] use of underlined printing.

Table 3 (Cont.): Summary of SETUP Modes

Mode	Type	Description
UNDERSCORE	Terminal	Displays the cursor as an underscore (Professional only).
UP	Terminal/Printer	Moves the print up 1/2 inch for alignment.
VERTICAL:n	Terminal/Printer	Prints *n* lines per inch.
VT100	Terminal	Sets terminal into VT100 specific mode.
[**NO**]**WRAP**	Terminal/Printer	Enables [disables] wrapping around to the following line.

Modes

12HOUR

(Valid for Professional 300 series processors only) Converts the contents of the time-of-year clock to 12-hour mode and puts the clock in 12-hour mode.

24HOUR

(Valid for Professional 300 series processors only) Converts the contents of the time-of-year clock to 24-hour mode and puts the clock in 24-hour mode. For example, 1:00 PM is represented as 13:00.

132COLUMNS

Sets your terminal to display 132 characters on each line. 132COLUMNS cannot be used with the Professional series processors when running under an unmapped monitor.

80COLUMNS

Sets your terminal to display 80 characters on a line.

[NO][480]INTERLACE

(Valid for Professional 380 series processors only) 480INTERLACE enables the screen to display graphics at a higher resolution than INTERLACE mode.

NOINTERLACE disables 480INTERLACE (high-resolution graphic) mode. This is the default.

ANSWERBACK:/message/

(Valid for Professional terminals only) Lets you store up to 20 characters in a message that is automatically transmitted from the terminal to the computer at power up. The message must be enclosed by delimiters such as slashes (/) that are not part of the message.

A different form of the SETUP ANSWERBACK command must be used if you want the message in lowercase letters, or if you want to embed returns in the message. Let SETUP prompt for input by typing:

```
.SETUP  RET
Characteristics?: ANSWERBACK:/message/
```

You can use lowercase letters and embed returns in your message. The message is read until the final delimiter (/) is reached, regardless of returns.

[NO]BCD

(Valid for Professional 300 series processors only) BCD (binary coded decimal) converts the contents of the time-of-year clock to BCD and puts the clock in BCD mode.

NOBCD converts the contents of the time-of-year clock to binary, and puts the clock in binary mode.

BLOCK

(Valid for Professional 300 series terminals only) Sets the cursor as a blinking block ([ ]) on your screen. This is the default.

[NO]BOLD

BOLD bolds the characters displayed on your terminal screen or printer. BOLD cannot be coupled with SETUP MEMO.

When you set NOBOLD and the destination is TERMINAL, bolding and underlining (if turned on) are turned off. If you want to keep underlining, you must reissue the SETUP UNDERLINE command.

When you set NOBOLD, and the destination is PRINTER, only bolding is turned off. NOBOLD is the default.

CAPS

(Valid for Professional terminals only) Enables the LOCK key to enter all letters as uppercase. This is the default. In this mode, the LOCK key serves the same function as the CAPS LOCK key on the VT100 series.

CLEAR

Clears the screen, executes a Shift In (SI) function, and performs the Digital standard SOFT RESET.

The SI function moves the G0 (7-bit ASCII) graphic character set into the ASCII code table. That procedure makes sure the video or hard-copy terminal or printer is placed in ASCII standard text mode after receiving a SETUP CLEAR command.

The Digital standard SOFT RESET:

- Sets the current rendition to NORMAL.
- Resets scrolling margins to 1 and 24.

- Sets the ORIGIN mode to ABSOLUTE.
- Sets wrap to OFF.
- Sets CURSOR KEY mode to CURSOR ON.
- Sets KEYPAD mode to NUMERIC.
- Sets INSERT/REPLACE mode to REPLACE.

[NO]CLICK

(Valid for Professional terminals only) CLICK, the default, causes the keys to click when you press them.

NOCLICK causes the keys to make no sound when you press them.

CLOCK SHOW

(Valid for Professional 300 series time-of-year clocks only) Displays the current settings of:

DATE mode (BCD or binary)
12/24 HOUR mode (12 or 24 hour)
Daylight Saving Time (DST) enable (on or off)

COLOR

(Valid for Professional 300 series processors only) Set COLOR only if you have a color terminal. COLOR enables your color terminal to support color functionality with utilities such as GIDIS.

MONO

(Valid for Professional 300 series processors only) Tells your processor that your terminal is not a color terminal. This is the default. Set MONO mode when your processor does not have a color terminal.

[COLOR1] [ON COLOR2]

(Valid for Professional color terminals only) COLOR1 ON COLOR2 specifies the color (COLOR1) of the text and the color (COLOR2) of the screen background. You can specify any of the following colors for COLOR1 and COLOR2: red, yellow, blue, green, black, white, magenta, and cyan.

The following command displays green text on a blue background:

```
.SETUP GREEN ON BLUE
```

You can specify colors for only the text or only the screen background. The following command assigns blue for the text but does not change the background color:

```
.SETUP BLUE
```

The next command assigns blue for the screen background but does not change the text color:

```
.SETUP ON BLUE
```

Color assignment for text and background can be reversed by issuing SETUP LIGHT or SETUP DARK.

[NO]COMPOSE

(Valid for Professional 300 series processors only) Used with the SETUP LANG:code command, COMPOSE lets you combine keys to create composed characters from an NRC set. See Appendix C for further information on the NRC character set.

NOCOMPOSE disables the COMPOSE CHARACTER key.

[NO]CURSOR

(Valid for Professional 300 series processors only) CURSOR, the default, displays a *blinking* cursor on the terminal screen. The cursor's form is set using SETUP BLOCK or SETUP UNDERSCORE.

NOCURSOR removes the display of the cursor from the terminal screen.

DARK

Displays light characters on a dark screen background, or reverses text and background colors on a Professional color video monitor. This is the default.

DATA

(Valid for Professional 300 series processors only) Used with SETUP LANG:code, DATA enables the NRC data-processing character set of the currently defined NRC. See Appendix C for further information on the NRC character set.

DATE[:dd:mm:yy]

When you set DATE with a Professional 300 series processor or with a PDP–11 processor having a KDJ11–E clock, you set the specified date into both the monitor date and the time-of-year clock. If any or all fields of the date are missing, the date is taken from the current monitor date. The date is stored in the time-of-year clock in whatever format (BCD or binary) is currently in the clock. A battery in the processor maintains the clock, so that you need set it only once.

When you set the DATE with a PDP–11 processor not having a KDJ11–E clock, only the monitor date is set.

Century Date Change

The syntax supported by the SETUP [CLOCK] DATE command allows you to enter dates for the years following 1999 if your processor has a KDJ11–E clock. You enter the years as 2000, 2001, and so on.

DEFAULT

(Valid for Professional terminals only) Sets the terminal modes to the factory settings.

DENSE

Sets your dot-matrix printer or hard-copy terminal to operate at 30 characters a second. It overlaps two sets of dots for a close approximation of letter-quality printing. DENSE prints a more legible character than MEMO.

[NO]DST

(Valid for Professional 300 series processors only) DST (Daylight Saving Time) puts the time-of-year clock in daylight saving time mode.

NODST takes the time-of-year clock out of daylight saving time mode.

DOWN

Lowers the print line on your printer or hard-copy terminal 1/12 inch relative to the top-of-form. The paper is moved forward (upward) 1/12 inch. You can use DOWN for remote adjustment of the top-of-form position. DOWN requires the partial line adjustment feature; consult your user guide.

DRAFT

Sets the printer or hard-copy terminal to an 8-x-9 dot matrix. DRAFT mode is the same as the DATA mode used with some printers.

GENERIC100

(Does not apply to VT100 series terminals) Sets a Professional or VT200/300 series terminal to reply to a device attributes request with a code sequence identifying the terminal as a nonspecific VT100 series terminal. Digital recommends that all future programs be written to recognize the GENERIC100 device attributes reply.

The device attributes exchange is the same for the Professional and the VT200/300 series terminals when SETUP GENERIC100 is set. For a complete explanation of the device attributes exchange for these terminals, see the *RT–11 Device Handlers Manual*.

GRAPHIC

(Does not apply to VT100 series terminals) Places your terminal or printer in GRAPHIC mode. GRAPHIC mode selectively displays individual dot patterns using bit-mapping, rather than printing predefined characters. Check your terminal or printer user guide for support of this type of graphics (bit-mapping) capability.

HELP

Gives a pointer to the help messages for major types of SETUP mode.

HORIZONTAL:n

Sets *n* number of characters to be printed in a horizontal inch. Valid values for *n* are 5, 6, 8.25, 10, 12, or 16.5. HORIZONTAL can be used with some hard-copy terminals.

[NO]INTERLACE

INTERLACE enables the screen to display high-resolution graphics.

To use INTERLACE mode with the graphics utility (GIDIS), issue the SETUP INTERLACE command before running GIDIS. You must exit GIDIS before changing the INTERLACE mode.

NOINTERLACE disables INTERLACE (high-resolution graphic) mode. This is the default.

JUMP

Scrolls new lines on the screen as fast as the processor sends them to the terminal. Reading lines in JUMP mode is difficult at higher baud rates.

LANDSCAPE

(Valid for LN03 laser printers only) Sets an LN03 printer to print across the wider dimension of the page. If the paper dimensions are 8 1/2 x 11 inches, LANDSCAPE sets the LN03 to print across the 11-inch dimension.

LANG:code

(Valid for Professional 300 series processors only) Defines the character set of your console terminal and line printer. The command sets the keyboard's character set to the specified code.

You can specify one language code for the terminal and another language code for the terminal screen. When you set KB LANG:code, only the keyboard is set to the specified language code. The terminal screen font remains unaffected. When you set VIDEO LANG:code, only the terminal screen font is set to the specified language code. The keyboard remains unaffected.

The Professional 300 series terminal display can be disrupted when changing from one video font to another. The valid codes with the countries they specify are as follows. *USA* is the default code:

Country	**Code**
Australia	AUS
Austria	A
Belgium	B
Canada (English)	CDNE
Canada (French)	CDNF
Denmark	DK
Finland	SF
France	F
Germany	D
Ireland	IRL
Italy	I
Netherlands	NL
New Zealand	NZ
Norway	N
Spain	SP
Sweden	S
Switzerland (French)	CHF
Switzerland (German)	CHD

United Kingdom	GB
United States	USA

See Appendix C for further information on the NRC character set.

LETTER

Sets the dot-matrix serial-line printer or hard-copy terminal to DENSE and HORIZONTAL:10.

LIGHT

Displays dark characters on a light screen background or reverses text and background colors on a Professional color video monitor.

LISTING

Sets the dot-matrix printer or hard-copy terminal to DRAFT and HORIZONTAL:16.5.

[NO]MARGINBELL

(Valid for Professional 300 series terminals only) MARGINBELL causes the terminal to beep when the cursor is eight characters from the right margin.

NOMARGINBELL, the default, disables the terminal's beep when the cursor is eight characters from the right margin.

MEMO

Changes the printer or hard-copy terminal dot matrix from 8-x-9 to 16-x-9 dots. MEMO causes more fully formed characters to be printed; they are more legible than DRAFT. MEMO can be used with some hard-copy terminals.

[NO]NEWLINE

NEWLINE causes RETURN to generate a <CR><LF> (return, line feed) combination. RT–11 automatically generates a line feed, so setting NEWLINE sends two line feeds.

NONEWLINE causes RETURN to generate only a return. However, RT–11 generates a line feed; so, setting NONEWLINE generates one <CR><LF> (return, line feed) combination.

PAGELENGTH:n

Sets the *n* number of lines printed on a page. Valid values for *n* are 0 to 252. If the argument is 0, the printer or hard-copy terminal ignores form feeds.

PORTRAIT

(Valid for LN03 laser printers only) Sets the LN03 printer to print across the narrower dimension of the page. This is the default. If the paper dimensions are 8 1/2 x 11 inches, PORTRAIT sets the LN03 to print across the 8 1/2-inch dimension.

[NO]REPEAT

REPEAT, the default, sets a key to repeat at the rate of about 30 characters per second when you press the key for more than 1/2 second.

NOREPEAT sets a character to be displayed only once no matter how long you press the key.

RESET

Sets modes, text, and background color to the settings that were set using the most recent SETUP SAVE command, or by a hardware SAVE (see your device's user guide).

RETAIN

(Valid for Professional terminals only) Saves the language code previously specified, using the SETUP LANG:code command. To use the SETUP RETAIN command, you must have the unsupported utility SPLIT.SAV on your system device. Make sure SPLIT.SAV is on your system device before you issue the SETUP RETAIN command. Use the SETUP RETAIN command on your working system only; do not use that command on your distribution or backup media.

You must remove the protection from the PI handler (PI[X].SYS) before you issue the SETUP RETAIN command. To remove the protection from the PI handler, use the UNPROTECT PI[X].SYS command.

SAVE

(Valid for Professional terminals only) Preserves the current SETUP modes, text, and background color and makes those modes the default modes. If the SETUP modes are changed, you can restore them by issuing a SETUP RESET.

SETCOLOR color [red:value,blue:value,green:value]

(Valid for Professional 300 series color terminals only) Lets you mix any or all three primary colors (red, blue, green) to create and store the exact shade you want for any of the eight colors specified in the SETUP [color1] ON [color2] command. These eight colors are BLACK, BLUE, CYAN, GREEN, MAGENTA, RED, WHITE, and YELLOW.

Value is a decimal number that determines the amount of primary color in the created color. A lower number specifies a smaller amount and a higher number a larger amount of that primary color in the mix.

The range of values for primary colors is 0–15 for both the Professional 350 and 380 processors. However, the actual incremental change in intensity between one primary color and the next differs between processors. However, either processor will produce approximately the same color from the primary colors.

For either processor, any primary color not specified is assigned the value zero. If all primary colors are assigned zero value, the color is BLACK.

The Professional 350 has eight primary color values for GREEN and RED and four values for BLUE. Primary color intensity changes with values 0, 2, 4, 6, 8, 10, 12, and 14 for GREEN and RED, and changes with values 0, 4, 8, and 12 for BLUE. See the following table:

Value	Blue	Green	Red
0	change	change	change
1			
2		change	change
3			
4	change	change	change
5			
6		change	change
7			
8	change	change	change
9			
10		change	change
11			
12	change	change	change
13			
14		change	change
15			

The Professional 380 implements the full range of values (0–15) in primary color amounts. For example, the following command first sets BLUE to medium, with a little GREEN and no RED. The command then generates a medium-blue text that contains a little green:

```
SETUP SETCOLOR BLUE GREEN:1,BLUE:7
```

SETCOLOR color FACTORY

(Valid for Professional color terminals only) Sets the specified color to the distributed setting.

SHIFT

Sets the LOCK key so that it can enable all keys to enter their alternate (shifted) function. In this mode, the LOCK key serves the same function as the shift lock key on a typewriter.

SHOW

(Valid for Professional terminals only) Displays current SETUP mode settings on the screen. For example, if the terminal is set for the original (factory-set) default modes, issuing the SETUP SHOW command displays the following report:

```
CAPS          is ENABLED
CLICK         is ENABLED
COLOR         is DISABLED
COMPOSE       is DISABLED
CURSOR        is ENABLED
DARK          is ENABLED
DATA          is ENABLED
JUMP          is ENABLED
LIGHT         is DISABLED
MARGINBELL    is DISABLED
MONOCHROME    is ENABLED
NEWLINE       is DISABLED
REPEAT        is ENABLED
SHIFT         is DISABLED
SMOOTH        is DISABLED
TYPE          is DISABLED
WRAP          is DISABLED
132COLUMNS    is DISABLED
80COLUMNS     is ENABLED
No Answerback Message
Device attributes reply string: VT100 with AVO and processor

Tabstop settings:
        T       T       T       T       T       T       T       T      T
12345678901234567890123456789012345678901234567890123456789012345678901234567890

 NRC is set to USA
Video FONT is set to USA
```

SMOOTH

Scrolls new lines on the screen at a maximum of six lines a second. SMOOTH lets you read lines on the screen more easily. For processors other than the Professional 300 series, AUTOX must be set using the local set-up mode or the local set-up menu when you use SMOOTH.

[NO]TABS[:n:n:n...]

TABS:n specifies the column number *n* for each successive tab stop. You must include the SET TT TAB command in your start-up command file, or issue the command each time you boot your system.

By default, tab stops are set every eight columns. To restore the default tab settings, issue SETUP TABS with no argument.

Set NOTABS to clear all tab stops. To implement only the new tab stops specified by SETUP TABS:*n*, issue NOTABS before TABS:*n*. Otherwise, the previous tab stops remain in effect along with the new tab stops.

TEXT

Exits your terminal or printer from graphic mode. Text mode displays predefined characters from a selected character set. This is the default.

TIME[:hh:mm:ss]

TIME, on a Professional 300 series processor or on a PDP–11 processor with a KDJ11–E clock, sets the specified time into the monitor time and time-of-year clock. If any or all fields of TIME are missing, they are taken from the current monitor time. The time is stored in the time-of-year clock in whatever format (BCD or binary) is currently in the clock. A battery in the processor maintains TIME, so that you need set it only once.

TIME, on a PDP–11 processor not having the KDJ11–E clock, sets only the monitor time.

TYPE

(Valid for Professional 300 series processors only) Used with SETUP LANG:code, TYPE enables the NRC typewriter character set of the currently defined NRC. See Appendix C for further information on the NRC character set.

[NO]UNDERLINE

UNDERLINE causes displayed or printed text to be underlined. If the *hardware* parameter is TERMINAL and SETUP BOLD is in effect, issuing SETUP NOBOLD automatically issues SETUP NOUNDERLINE. See SETUP BOLD.

When you set NOUNDERLINE, the default, text displayed or printed is not underlined.

UNDERSCORE

(Valid for Professional terminals only) Makes the cursor appear as a blinking underline (_) on your screen.

UP

Raises the print line in your printer or hard-copy terminal 1/12 inch relative to the top-of-form. The paper is moved backward (downward) 1/12 inch. You can use UP for remote adjustment of the top-of-form position. UP requires support of the partial line adjustment feature; check your user guide.

VERTICAL:n

Sets the number (*n*) of lines printed for each vertical inch. Valid values for *n* are 2, 3, 4, 6, 8, or 12. VERTICAL can be used with some hard-copy terminals.

VT100

Sets the terminal to transmit and decode ANSI codes. SETUP VT100 is also used by the Professional terminal to reply to a device attributes request as though it were a VT100 (with advanced video option). See the *RT–11 Device Handlers Manual* for a complete explanation of the device attributes exchange.

VT52

Sets the terminal to transmit and decode according to the requirements of the VT52 terminal. Digital does not support VT52 mode.

[NO]WRAP

WRAP moves characters you type that exceed the right margin to the beginning of the following screen line.

NOWRAP, the default, causes characters you type that exceed the right margin to overwrite the last character of the current screen line and be lost.

SHOW

(RESORC)

Lists on the console terminal information about the monitor configuration and the available hardware.

Format

SHOW *option*

Command Options

```
ALL
COMMANDS
  [ /OUTPUT:filespec ]
  [ /PRINTER         ]
  [ /TERMINAL        ]

CONFIGURATION
DEVICES[:dd]
ERRORS
  [ /ALL             ]
  [ /FILE:filespec   ]
  [ /FROM[:date]     ]
  [ /OUTPUT:filespec ]
  [ /PRINTER         ]
  [ /SUMMARY         ]
  [ /TERMINAL        ]
  [ /TO[:date]       ]

JOBS
MEMORY
QUEUE
SUBSET
TERMINALS
UMR
```

Description

Unlike other DCL commands, the SHOW command parameters are called options of the command. The SHOW command displays the following information:

- Hardware configuration
- Monitor version
- Total amount of memory on the system and organization of physical memory
- Special features in effect
- Device names and logical device-name assignments
- Whether a device is assigned as a default device

- Terminal characteristics for terminals currently active on a multiterminal system
- Logical-disk subsetting
- Device handler status
- If you are running the Error Logger, QUEUE, or SPOOL, the SHOW command can provide information on:
  - — Errors
  - — The update status of files waiting to be sent to an output device
  - — User-defined commands

Specifying SHOW Without an Option

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. This list reflects any additions or deletions you have made with the INSTALL and REMOVE commands. The listing also includes additional information about devices. For example, an asterisk next to a device indicates you have assigned that device to be a default device. A default device is created by using the *ASSIGN dev ** .

The SHOW Command Informational Messages

The informational messages displayed by the SHOW command are as follows.

Message	**Meaning**
(RESORC) *or* =RESORC	The device or unit is assigned to the background job RESORC (for multi-job unmapped and mapped monitors only).
(F) *or* =F	The device or unit is assigned to the foreground job (for multi-job unmapped and mapped monitors only and monitors without system-job support).
(jobname) *or* =jobname	The device or unit is assigned to the system or foreground job (for multi-job unmapped and mapped 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.

An Example Illustrating the Preceding Messages

The following example was created under a multi-job unmapped monitor that has system-job support. It shows the status of all devices known to the system:

```
.SHOW
TT  (Resident)
DU  (Resident)
  DU0   = SY, DK, OBJ, SRC, BIN
  DU1   = LST, MAP
MQ  (Resident)
DL  (Loaded)
DM
DX  (Loaded)
  DX0:  (MYPROG)
LP: (Loaded=QUEUE)
MT
5 free slots
```

The listing shows first that TT, MQ, and DU are resident in memory. The other device handlers known to the system are DL, DM, DX, LP, and MT. There are five free slots in the table. DU0 has the logical names SY, DK, OBJ, SRC, and BIN. DU1 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.

Options

ALL

Acts as a combination of CONFIGURATION, DEVICES, device assignments (SHOW command with no option), JOBS, TERMINALS, MEMORY, and SUBSET in that order:

```
.SHOW ALL

RT-11XM  V05.6
Booted from DU0:RT11XM
22 bit addressing is on

USR     is set NOSWAP
EXIT    is set SWAP
KMON    is set NOIND
RUN     is set NOVBGEXE
MODE    is set NOSJ
TT      is set NOQUIET
ERROR   is set ERROR
SL      is set OFF
EDIT    is set KEX
FORTRAN is set FORTRA
KMON nesting depth is 3

CLI is set DCL, CCL, UCL, NO UCF
```

```
PDP 11/23 PLUS Processor
512KB of memory
Extended Instruction Set (EIS)
Memory Management Unit
Parity Memory
60 Hertz System Clock

Device I/O time-out support
System job support
FPU support

Device    Status                   CSR     Vector(s)
------    ------                   ---     ---------
  DL      Installed               174400   160
  DM      Not installed           177440   210
  DU      Resident                172150   154
  DX      Installed               177170   264
  DY      Not installed           177170   264
  LD      Installed               000000   000
  LP      Not installed           177514   200
  LS      Installed               176500   470 474 300 304
  MM      Not installed           172440   224
  MS      Installed               172522   224 300
  MT      Not installed           172520   224
  MU      Installed               174500   260
  NL      Installed               000000   000
  NQ      Not installed           174440   120
  NU      Not installed           174510   120
  RK      Not installed           177400   220
  SL      Installed               000000   000
  SP      Installed               000000   110
  UB     -Not installed           170200   000
  VM      Installed               177572   000
  XL      Installed               176500   300 304
  DW      Not installed           000000
  DZ      Not installed           000000
  NC      Not installed           000000
  PI     -Not installed           000000   000
  XC      Not installed           173300   210 214

TT  (Resident)
DU  (Resident)
    DU0 = DK , SY
MQ  (Resident)
LD
SL
DL
DX
VM
SP
XL
MS
MU
LS
NL
17 free slots
```

```
Job  Name   Console Level State    Low    High   Impure
---  ----   ------- ----- -----    ---    ----   ------
 0   RESORC   0       0   Run     000000 105142 121630

No multi-terminal support

------- Extended Memory --------
Address   Module    Words   Type
-------   ------    -----   ----
17760000  IOPAGE     4096.   PRM    HDW    BYP
02000000  MEMTOP
01000000  VM       131072.   SHR
00220700  ......    93984.
00206300  SL         2688.   PVT
00200200  MU         1568.   PVT
00171700  SP         1632.   SHR
00165000  DU         1248.   PVT
00160000  MCA        1280.   PRM
00000000  KERNEL    28672.   PRM    HDW

------ Low Memory -------
Address   Module    Words
-------   ------    -----
157340    DU          144.
117206    RMON       8237.
105144    USR        2577.
001000    ..BG..    17458.

No LD units mounted
```

COMMANDS

SHOW COMMANDS sends a list of created UCL commands to the terminal. The COMMANDS parameter can be qualified by the /OUTPUT:filespec, /PRINTER, or /TERMINAL (the default) options. These options are position dependent; that is, they must follow the COMMANDS parameter. The following example illustrates the SHOW COMMANDS display:

```
.SHOW COMMANDS
!                          User Command Linkage (UCL)

      ALL              :== SHOW ALL\TIME\DATE\\
      FILES            :== DIRECTORY/VOLUME/COLUMNS:1 *.TXT
      V                :== VBGEXE
      VM               :== $@VM\BOOT VM:\\
      VMPRG            :== $@VMPRG\BOOT VM:\\
      VMGEN            :== $@VMGEN\BOOT VM:\\
      VMMIN            :== $@VMMIN\BOOT VM:\\
      STAT*us          :== SHOW ALL\SHOW QUEUE\\
      KILL             :== UNPRO ^\DEL/NOQ ^\\
```

COMMANDS/OUTPUT:filespec

Sends a list of created UCL commands to the specified file (SHOW COMMANDS /OUTPUT:filespec).

COMMANDS/PRINTER

Sends a list of created UCL commands to the printer (SHOW COMMANDS /PRINTER).

CONFIGURATION

Displays the monitor version number and update level, the monitor SET options in effect, the hardware configuration, the total amount of memory on the system, and the special features in effect (if any). The listing varies, of course, depending on which monitor and which hardware system you are using.

Description of a CONFIGURATION Listing

- **Version Number and Update Level**

  First, the listing always shows the version number and update level of the currently running monitor.

- **Monitor Information**

  Next, information about the monitor is displayed.

  — The first line indicates the device from which the system was bootstrapped.

  — The next line indicates whether or not 22-bit addressing is on if you are running a mapped monitor.

  — Then the listing shows:

  Extended device-unit support
  SET CLOCK and SET FORTRA conditionals
  User service routine (USR) status: SWAP or NOSWAP
  EXIT status: SWAP or NOSWAP
  Active command file processor: KMON or IND
  SET RUN status
  SET MODE status
  Terminal status: QUIET or NOQUIET
  ERROR severity level
  SL status: ON, OFF, KMON
  Default editor for the EDIT command
  File nesting level (a decimal number)
  Status of .SCCA support and the .SCCA flag when enabled
  CLI (Command Line Interpreter) setting

- **System Hardware Configuration**

  Next, the listing displays the system hardware configuration. This includes:

  — The processor type, such as a MicroPDP–11 or a PDP 11/84

  — The total amount of memory your system contains; for example:

  ```
  1022KB of memory
  ```

- A separate line for each of the following items that is present on your system:

  ```
  Extended arithmetic element (EAE)
  FP11 Hardware Floating Point Unit
  Commercial Instruction Set (CIS)
  Extended Instruction Set (EIS)
  Floating Instruction Set (FIS)
  KT11 Memory Management Unit
  Parity Memory
  ECC Memory
  Cache Memory
  60 Hertz System Clock
  ```

- Another line displays any graphics hardware (VT11 or VS60) you might have
- The clock frequency (50 or 60 Hertz) displays next
- The KW11–P programmable clock, if one is on your system
- Finally, the listing either shows that no special features are in effect, or it lists the appropriate features from the following list:

  ```
  Device I/O time-out support
  Error logging support
  Floating Point Microcode
  FPU support
  Multi-terminal support
  Memory parity support
  SB timer support
  System job support
  ```

An Example of the SHOW CONFIGURATION Display

The following example was created on a PDP 11/23 PLUS processor:

```
.SHOW CONFIGURATION

RT-11XM  V05.6
Booted from DU0:RT11XM
22 bit addressing is on

USR     is set NOSWAP
EXIT    is set SWAP
KMON    is set NOIND
RUN     is set VBGEXE
MODE    is set NOSJ
TT      is set NOQUIET
ERROR   is set ERROR
SL      is set OFF
EDIT    is set KEX
FORTRAN is set FORTRA
KMON nesting depth is 3

CLI is set DCL, CCL, UCL, NO UCF
```

```
PDP 11/23 PLUS Processor
512KB of memory
Extended Instruction Set (EIS)
Memory Management Unit
Parity Memory
60 Hertz System Clock

Device I/O time-out support
System job support
FPU support
```

DEVICES[:dd]

Displays:

- RT–11 device handlers and their status.
- CSR addresses and vectors.
- Installability information for any device handler that, because of the handler characteristics or your system configuration, cannot be installed in some manner on your system. You can obtain this information for a specific device by including the optional argument *dd*. The *dd* variable specifies the two-letter permanent device mnemonic.

SHOW DEVICES:DU displays additional DU status information. The port display indicates an invalid port by displaying an asterisk (*) before the port number. For example, port = *n, where n = 0,1,2,3, indicates that port n was not installed. If SHOW DEVICES:DU displays port = n, where n = 0,1,2,3, then port n is installed.

SHOW DEVICES:MU displays MU status information for the PORT and UNIT of each installed TMSCP controller in the same manner that the DU argument displays MSCP status information. (Partitioning is invalid with magtapes.)

The SHOW DEVICES command displays the following messages about a device's installation status.

Message	**Meaning**
Installed	Known to the monitor
nnnnnn	Load address of handler
Resident	Permanently located in memory
Not installed	Not known to the monitor
-Not installed	Not known to the monitor and not installed because the handler special features do not match those of the monitor
Not BSTRAP installable	Handler must be installed by the INSTALL
Not KMON installable	Handler must be installed at the system boot
Not installable	Handler cannot be installed on this system

The following example illustrates the SHOW DEVICES command, first with a specified device and then without a specification:

```
.SHOW DEVICES:DL

Device    Status                   CSR     Vector(s)
------    ------                   ---     ---------
  DL      Installed               174400   160

.SHOW DEVICES

Device    Status                   CSR     Vector(s)
------    ------                   ---     ---------
  DL      Installed               174400   160
  DM      Not installed           177440   210
  DU      Resident                172150   154
  DX      Installed               177170   264
  DY      Not installed           177170   264
  LD      Installed               000000   000
  LP      Not installed           177514   200
  LS      Installed               176500   470 474 300 304
  MM      Not installed           172440   224
  MS      Installed               172522   224 300
  MT      Not installed           172520   224
  MU      Installed               174500   260
  NL      Installed               000000   000
  NQ      Not installed           174440   120
  NU      Not installed           174510   120
  RK      Not installed           177400   220
  SL      Installed               000000   000
  SP      Installed               000000   110
  UB     -Not installed           170200   000
  VM      Installed               177572   000
  XL      Installed               176500   300 304
  DW      Not installed           000000
  DZ      Not installed           000000
  NC      Not installed           000000
  PI     -Not installed           000000   000
  XC      Not installed           173300   210 214
```

Because of its special format, the TT handler is never listed.

ERRORS

SHOW ERRORS 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 the chapter on that utility in the *RT–11 System Utilities Manual*. Note that error logging is a special feature, available only through the system-generation process. Because the Error Logger can compile 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.

SHOW ERROR invokes ERROUT, one of the programs in the error logging package. ERROUT runs as a background job under a multi-job unmapped monitor and mapped monitors, and as the only job under the SB monitor. 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. If you type the SET dd: NOSUCCESS command before you use the Error Logger, the Error Logger compiles statistics on only the errors that occurred, not the successful I/O transfers. Therefore, the reports generated when you type SHOW ERRORS will list only the errors that occurred. For complete descriptions of the reports ERROUT creates, see the Error Logging chapter in the *RT–11 System Utilities Manual*.

ERRORS
Displays 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.

ERRORS/FILE:filespec
Displays 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 statistical compilations.

ERRORS/FROM:date
Displays 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 specifies a two-digit day.

mmm specifies the first three characters of a month's name.

yy specifies the last two digits of a year.

ERRORS/TO:date
Displays 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
Displays the I/O transfer and error report at the line printer.

ERRORS/SUMMARY
Displays 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
Displays the I/O transfer and error report at the terminal. /TERMINAL is the default setting.

JOBS

Displays data about the jobs that are currently loaded. This option also tells the following:

- 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)
- Console the job owns (with a nonmultiterminal monitor, this space is blank)
- Priority level of the job
- Job's running state (running, suspended, or done but not unloaded)
- Low and high memory limits of the job

- Start address of the job’s impure area

The following example 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  126110  132344
```

MEMORY

Lists the organization of physical memory. MEMORY shows the location of each low memory component and, under a mapped monitor, each extended memory region as well.

The MEMORY listing displays such information as where jobs are loaded, where device handlers are loaded, where in memory KMON and the USR will reside, the number of words of memory each occupies, and the type of region in the extended memory map. Memory addresses are displayed in octal.

The SHOW MEMORY command displays cache-bypass status for a global region, using the symbol BYP rather than the symbol GBL.

If you are running under a mapped monitor, the SHOW MEMORY listing is divided into two sections, the first for extended memory and the second for kernel memory.

The following example displays the organization of physical memory when running under the SB monitor:

```
.SHOW MEMORY

Address   Module    Words
-------   ------    -----
160000    IOPAGE     4096.
154332    DU          915.
135166    RMON       3890.
001000    ..BG..    23611.
```

The next example shows the organization of physical memory when running under the XM monitor:

```
.SHOW MEMORY

------- Extended Memory --------
Address   Module    Words   Type
-------   ------    -----   ----
17760000  IOPAGE     4096.   PRM    HDW    BYP
02000000  MEMTOP
01000000  VM       131072.   SHR
00220700  ......    93984.
00206300  SL         2688.   PVT
00200200  MU         1568.   PVT
00171700  SP         1632.   SHR
00165000  DU         1248.   PVT
00160000  MCA        1280.   PRM
00000000  KERNEL    28672.   PRM    HDW

------ Low Memory -------
Address   Module    Words
-------   ------    -----
157340    DU          144.
117206    RMON       8237.
105144    USR        2577.
001000    ..BG..    17458.
```

QUEUE

Displays a listing of the contents of the queue for QUEUE or SPOOL, or both if both are running.

This option is invalid if you are not running QUEUE or SPOOL (see the *RT–11 System Utilities Manual*). If the SPOOL utility is running, SHOW QUEUE requires RESORC.SAV be on device SY. If the QUEUE utility is running, SHOW QUEUE requires RESORC.SAV and QUEMAN.SAV be on device SY.

The SPOOL status report shows whether each SPOOL device unit and associated output device is active or inactive, the number of blocks spooled for output, and the number of free blocks in SPOOL's work file.

The following example shows two output devices, LS0 and LP0, attached to SPOOL, with LS0 printing and LP0 idle:

```
 Unit  Device  Status
 SP0:  LS0:    ACTIVE,      56 blocks spooled
 SP1:  LP0:    IDLE

   944 Free blocks in workfile
```

The QUEUE status report shows the output device, jobname, input files, job status, and number of copies for each job that is queued. The next example command lists the current contents of the queue for 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
MT1:    SZYM   Q       1      REFMAN.TXT
LP0:    JOYCE  Q       1      SSM   .DOC
                       1      DOCPLN.DOC
```

The job status column contains 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.

SUBSET

Displays the subsetting of physical disks into logical disks. This command displays:

- Assigned logical-disk units
- An asterisk next to any logical-disk unit assigned as a default device. A default device is created by using *ASSIGN dev ** .
- The file name to which each logical-disk unit is assigned
- The size of each logical disk in decimal blocks
- Any logical name assigned to a logical disk

For example, if you mount a 1000-block file DU4:WRK.DSK on logical device unit LD0 and assign it the name WRK, SHOW SUBSET displays:

```
LD0 is DU4:WRK.DSK[1000.] = WRK
```

The following sample command line displays the logical disks into which the physical disks DU and DL1 are divided:

```
.SHOW SUBSET
LD0 is DU:DISK.LST[4000.]
LD2 is DL1:DISK.SRC[1200.]
LD1 is DL1:WORK.DSK[600.]
```

An asterisk (*) following the file information indicates that although the logical-disk assignment exists, the file does not exist on the volume that is currently mounted in the drive unit. A number sign (#) indicates that the device handler is not loaded. These symbols are especially useful in determining the status of logical-disk assignments after you use the SET LD CLEAN command.

If LD.SYS is not installed, RT–11 displays the message *LD handler unavailable*. If no logical-disk units have been mounted (by using the MOUNT command), RT–11 displays *No LD units mounted*.

TERMINALS

Indicates the status of and special features in effect for currently active terminals on multiterminal systems. If the monitor does not include multiterminal support, the following message displays:

```
No multi-terminal support
```

Multiterminal support is a special feature; it is not part of the distributed RT–11 monitors.

If the monitor includes multiterminal support, SHOW TERMINALS displays a table of the existing terminals and lists the following information:

Unit number:	0-16
Owner:	Background, foreground, system job, or none
Type:	Local Remote (dial-up) Console S-Console (shared by background and foreground or system job) Is attached to another job (the foreground)
Interface type:	DL, DZ, or DH
Width:	Width in characters, up to 255
SET options in effect:	TAB CRLF FORM SCOPE
Line speed:	Baud rate if DZ or DH; not applicable (N/A) 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         Console   DL   80  No  No   No    Yes   N/A
  1         Local     DL   80  No  No   No    Yes   N/A
 10         Local     DH   80  No  No   No    Yes  9600
 11         Local     DH   80  No  No   No    Yes  9600
 12         Remote    DH   80  No  No   No    Yes  9600
```

UMR

Displays information about the UMRs (UNIBUS mapping registers) if the UB pseudohandler is loaded, or information about why UB is not loaded. UNIBUS mapping register support is described in the *RT–11 System Internals Manual*.

The following is an example display.

```
UMR allocation
--------------------------------------------------
00 ...... 1  10 ...... 2  20 ...... 9  30 ..MS.. P
01 ...... 1  11 ...... 2  21 ...... 9  31 ...... P
02 ...... 1  12 ...... 2  22 ...... 9  32 ...... P
03 ...... 1  13 ...... 4  23 ...... 9  33 ...... P
04 ...... 1  14 ...... 4  24 ...... 9  34 ...... P
05 ...... 2  15 ...... 4  25 ...... 9  35 ..DU.. P
06 ...... 2  16 ...... 4  26 ...... 9  36 ..DU.. P
07 ...... 2  17 ...... 9  27 ...... 9  37 IOPAGE P

2. UMRs in use
2. UMRs permanently assigned
0. UMRs dynamically assigned

0. requests waiting for UMR allocation

RESORC = NOSERIAL
```

SQUEEZE

(DUP)

- Moves all permanent files into contiguous blocks toward the beginning of the specified disk.
- Consolidates directory entries.
- Consolidates unused blocks into one or more empty areas at the end of the disk, depending on the existence and location of file BADs.

Format

SQUEEZE *device*

Command Options
/OUTPUT:device /[NO]**QUERY** /WAIT

Parameter

device

Specifies the random-access volume to be compressed.

Description

To perform a SQUEEZE operation, RT–11 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.

RT–11 requests confirmation before it performs the SQUEEZE operation. You must type Y RETURN to execute the command.

You should not attempt a SQUEEZE operation on a volume that a running foreground job is using. Data may be written over a file that the foreground job has open, thereby corrupting the file and possibly causing a system crash.

Use the SQUEEZE command if you get a directory full error and there is still space remaining on the volume; the SQUEEZE command consolidates both unused blocks on a volume and directory entries, making more usable space for both storage and directory entries.

Squeezing a Volume with BAD File Types

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. RT–11 inserts files before and after BAD files until the space between the last file is moved and the BAD file is smaller than the next file to be moved.

Squeezing the System Volume

If you perform a SQUEEZE operation on the system volume, RT–11 automatically reboots the running monitor when the compressing operation completes. This reboot takes place 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.

Options

/OUTPUT:device

Transfers 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 and empty random-access volume. (Use the INITIALIZE command to do this.) Note that RT–11 does not request confirmation before this operation proceeds. If the output device is not initialized, RT–11 displays an error message and does not execute the command.

Note: /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 DU0: to DU1: in compressed format, leaving DU0 unchanged.

```
.SQUEEZE/OUTPUT:DU1: DU0:
```

/[NO]QUERY

/QUERY causes RT–11 to request confirmation before it executes a SQUEEZE operation. You must respond by typing Y (or any string beginning with Y) RETURN to initiate execution of an operation. RT–11 interprets any other response to mean NO; it does not perform the specified operation. This is the default operation. /QUERY is invalid with the /OUTPUT option.

/NOQUERY suppresses the confirmation message that displays before a SQUEEZE operation executes. The following command compresses all the files on device DU1: and does not query:

```
.SQUEEZE/NOQUERY DU1:
```

/WAIT

Initiates the SQUEEZE operation, but then pauses and waits for you to mount the volume you want to SQUEEZE. This option is useful if you have a single-disk system.

When RT–11 pauses, it displays *Mount input volume in <device>; Continue?*. Mount the volume and type Y RETURN to continue the SQUEEZE operation.

Type N RETURN or a CTRL/C to abort the SQUEEZE operation and return control to the monitor. Any other response causes the message to repeat.

When the SQUEEZE operation is complete, RT–11 prompts you to remount the system volume. Mount the system volume and type Y RETURN. If you type any other response, RT–11 prompts you to mount the system volume until you type Y. RT–11 then displays the keyboard monitor prompt.

The following sample command line SQUEEZEs an RL02 disk:

```
.SQUEEZE/WAIT DL0:
DL0:/Squeeze; Are you sure? Y
Mount input volume in DL0:; Continue? Y
Mount system volume in DL0:; Continue? Y
```

RT–11 may repeat the mount input volume, mount output volume cycle several times to complete the SQUEEZE operation.

SRUN

(KMON)

Initiates system jobs. See also FRUN, R, RUN, and V/VRUN.

Format

SRUN *filespec*

Filespec Options
/BUFFER:value /LEVEL:value /NAME:[job-number/job-name] /PAUSE /TERMINAL:value

Parameter

filespec

Specifies the program to be executed.

The default file type is REL and the default device is SY.

You can use the SRUN command to run a virtual SAV image program on a mapped monitor, but you must type the file type (SAV) explicitly.

Description

Because this command runs a system job, it is valid only for monitors that have system-job support, a special feature enabled through the system generation process. Before you run a system job, make sure you load into memory the device handlers your system job needs.

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 displays on the terminal.

Running a System Job in a Completely Virtual Environment

To run a system job in the completely virtual environment, use the SRUN command in the following format:

```
SRUN  SY:VBGEXE.SAV[/NAME:prgnam][/options]
```

If you do not specify /NAME:prgnam, VBGEXE prompts with "Program?". Enter a file specification at the prompt to indicate the file to be run as a completely virtual system job. The default file extension is SAV. If you omit the device specification, VBGEXE tries to find the save-image file on the system device. If it cannot find the file there, it tries to find the save-image file on the default data device.

Options

/BUFFER:value

Reserves more space in memory than the actual program size. The *value* argument specifies the octal number of words in memory to allocate.

Virtual Jobs

If you use /BUFFER for a virtual job linked with the /V option (or /XM), RT–11 ignores /BUFFER because it has already provided a buffer in extended memory.

FORTRAN IV System Jobs

You must use this option to execute a FORTRAN IV system job.

The following formula determines the space needed to run a FORTRAN program as a system job:

```
n = [[504+(35*N)+(R-136)+A*512]/2]+[10*qcount]+[6*num]+[25*INTSET]+[64+R/2]
```

where:

`A` Specifies the maximum number of files open at one time. Each file opened as double buffered should be counted as two files.

`N` Specifies the maximum number_{10} 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` Specifies 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.

`qcount` Specifies queue elements.

`num` Specifies the number of channels.

`INTSET` Specifies the SYSLIB INTSET function.

Include the following optional elements in the formula if you want to use the indicated system subroutine library (SYSLIB) functions:

`[10*qcount]` Specifies space for queue elements, which the IQSET function requires.

`[6*num]` Specifies space for the number of channels, which the ICDFN function requires.

`[25*INTSET]` Specifies space for the number of INTSET calls issued in the program, which the INTSET function requires.

`[64+R/2]` Specifies space for completion routines and a second record buffer. Any functions, including INTSET, that invoke completion routines must include 64_{10} 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_{10} bytes, or the length that was set at FORTRAN installation time.

To specify a decimal number for *n*, include a decimal point (.) after the number you specify. The formula for determining the *n* calculates *n* as a decimal number, so include the decimal point when you specify the number.

Two Reasons for an Error Message

- 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
  ```

- The previous message also appears if there is not enough free memory for the background job or if a completion routine in an unmapped monitor is activated during another completion routine. In the latter case, the job aborts; you should use a multi-job unmapped monitor to run multiple active completion routines.

/LEVEL:value

Assigns an execution priority level to the job, where *value* 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 displays at the terminal. If omitted, the priority level defaults to the highest level that is unassigned.

/NAME:job-number/job-name

Assigns a *job-number* or a logical *job-name* to a program.

Programmed requests and SYSLIB calls use this number or name to reference a system job. If you do not specify a number or name, RT–11 assumes the file name of the program.

The following commands run VTCOM as a system job and assign it the number 6. All further references to VTCOM must use 6, and that number/name assignment remains in effect until you reboot your system.

```
SRUN SY:VTCOM.SAV/NAME:6/PAUSE
LOAD XC=6
RESUME 6
```

You would then specify 6 in response to the system job prompt (Job?) to call VTCOM. The SHOW JOBS command would display the job 6.

If you attempt to assign the same number or logical job-name to two system jobs, an error message displays at the terminal.

/PAUSE

(Used for debugging a program) When you press RETURN at the end of the command string, RT–11 displays the load address of your program and waits. You can then use ODT to examine or modify the program before starting execution (see the chapter on ODT in the *RT–11 System Utilities Manual*). You must use the RESUME command to restart the system job.

The following command loads the program MFUNCT.REL, displays the load address, and waits for a RESUME command to begin execution:

```
.SRUN MFUNCT/PAUSE
Loaded at 126556
.RESUME MFUNCT
```

/TERMINAL:value

(Requires multiterminal support, a feature available only through system generation) Changes the console of the system job. The *value* argument specifies 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 monitor, unless you use the SET TT: CONSOL command to change this.

START

(KMON)

Initiates execution of the program currently in memory (loaded with the GET command) at the address you specify.

Note that the START command is less versatile than the RUN command and may not initialize all data structures needed to run a program.

Format

START *[address]*

Parameter

address

Specifies an optional even octal number representing any 16-bit address. If you omit the address or if you specify 0, RT–11 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 may occur and the monitor displays an error message. Note that this command is valid for background jobs only, and not for extended-memory virtual jobs.

Examples

1. This command loads MYPROG.SAV into memory and begins execution:

```
.GET MYPROG
.START
```

2. This command loads MYPROG.SAV, which has previously been linked with ODT, into memory and begins execution at ODT's starting address (obtained from the link map):

```
.GET MYPROG
.GET ODT
.START 1222
.ODT V05.00
*
```

SUSPEND

(KMON)

Temporarily stops execution of the foreground or system job.

Format

SUSPEND *[jobname]*

Parameter

jobname

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.

Description

RT–11 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 System Subroutine Library Manual* for an explanation of completion routines). You can continue execution of the job by typing the RESUME command.

The SUSPEND command is not valid for single-job monitors.

Examples

1. This command suspends execution of the foreground job that is currently running on a system that does not have system job support:

   ```
   .SUSPEND
   ```

2. This command suspends execution of the system job, QUEUE, that is currently running on a system that does have system job support:

   ```
   .SUSPEND QUEUE
   ```

TIME

(KMON)

Sets or displays the time of day.

Format

TIME *[hh:mm:ss]*

Parameter

hh:mm:ss

Specifies the hours, minutes, and sections, where:

hh specifies the hour (from 0 to 23).

mm specifies the minutes (from 0 to 59).

ss specifies the seconds (from 0 to 59).

RT–11 keeps time on a 24-hour clock.

Description

To enter the time of day, specify the time in the format described above. You should do this as soon as you bootstrap RT–11.

To display the current time of day, type the TIME command without an argument.

When you install the standard RT–11 monitors, the clock rate is preset to 60 Hertz. Consult the *RT–11 System Generation Guide* for information on setting the clock to a 50-Hertz rate.

All monitors except the SB monitor 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 SB monitor resets the time under these conditions only if you select timer support during the system generation process.

Examples

1. This command enters the time, eleven fifteen in the morning:

   ```
   .TIME 11:15
   ```

   As this example shows, if you omit one of the arguments, RT–11 assumes 0.

2. This command displays the current time:

   ```
   .TIME
   11:15:01
   ```

TYPE

(PIP)

Displays the contents of one or more files on the terminal.

Format

TYPE *filespec[s]*

Command Options

```
⎧ ⎡ /BEFORE[:date] ⎤ ⎫
⎨ ⎣ /SINCE[:date]  ⎦ ⎬
⎪  /DATE[:date]      ⎪
⎩  /NEWFILES         ⎭
  /COPIES:value
  /DELETE
  /INFORMATION
  /[NO]LOG
  /QUERY
  /WAIT
```

Parameter

filespec[s]

Specifies the file or files to be typed. If you omit the file type for a file specification, RT–11 assumes LST.

If you specify more than one file, separate the files by commas. You can explicitly specify up to six file specifications. RT–11 displays the files in the order you specify in the command line.

You can also use wildcards in the file specification. In this case, RT–11 displays the files in the order in which they occur in the directory of the device you specify.

Description

To stop the display of multiple files, type CTRL/O. Though the display stops, PIP continues to execute. Output to the terminal resumes when you type CTRL/O a second time or when PIP reaches the beginning of the next file.

Specifying a Date Argument to an Option

Some of the options accept a date as an argument. The format for specifying the date is:

[dd][:mmm][:yy]

where:

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

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), RT–11 uses the values from the current system date. For example, if you specify only the year 90 and the current system date is May 4, 1991, RT–11 uses the date 4:MAY:90. If the current 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 at 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.)

Options

/BEFORE[:date]

Displays on the terminal all files on a specified volume created before a specified date. The following command displays only those MAC files on DK created before March 24, 1991:

```
.TYPE/BEFORE:24:MAR:91 *.MAC
```

/COPIES:value

Lists more than one copy of the file. The meaningful range of values for the *value* argument is from 2 to 32_{10} (1 is the default). The following command, for example, displays three copies of the file REPORT.LST on the terminal:

```
.TYPE/COPIES:3 REPORT
```

/DATE[:date]

Displays on the terminal only those files with a certain creation date. If no date is specified, the current system date is used. The following command prints on the console all MAC files on DK created on March 20, 1991:

```
.TYPE/DATE:20:MAR:91 DK:*.MAC
```

/DELETE

Deletes 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 if you want to be asked about each file to be displayed and deleted.

The following example displays a BASIC program on the terminal, then deletes it from DU1:

```
.TYPE/DELETE DU1:PROG1.BAS
```

/INFORMATION

Changes the severity level of the error message that displays when not all of the input files you specified are found. If you do not use /INFORMATION, RT–11 displays an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, RT–11 displays an informational message to tell you which files it cannot find, but execution continues.

In the following example, the input files FILE1.TXT and FILE3.TXT are displayed. However, since RT–11 is unable to find DL0:FILE2.TXT, RT–11 displays a message to inform you:

```
.TYPE/INFORMATION DL0:(FILE1,FILE2,FILE3).TXT
?PIP-I-File not found DL0:FILE2.TXT
```

/[NO]LOG

/LOG displays on the terminal the names of the files that were displayed by the current command. Normally, RT–11 displays 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 prevents a list of the displayed files from displaying on the terminal. You can use this option to suppress the log if you use a wildcard in the file specification.

/NEWFILES

Displays only those files that have the current date. The following example shows a convenient way to display all new files after a session at the computer:

```
.TYPE/NEWFILES *.LST
Files copied:
DK:REPORT.LST   to TT:
```

/QUERY

Causes RT–11 to request confirmation before it performs the operation. /QUERY is 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 displayed on the terminal replace the log messages that would normally appear.

You must respond to a query message by typing Y (or any string beginning with Y) RETURN to initiate execution of an operation. RT–11 interprets any other response to mean NO and does not perform the specific operation:

```
.TYPE/QUERY/DELETE *.LST
Files copied/deleted:
DK:OUTFIL.LST   to TT:? N
DK:REPORT.LST   to TT:? Y
```

/SINCE[:date]

Displays on the terminal all files on a specified volume created on or after a specified date. The following command displays only those MAC files on DK created on or after April 21, 1991:

```
.TYPE/SINCE:21:APR:91 *.MAC
```

/WAIT

Initiates the TYPE operation, but then pauses and waits for you to mount the volume that contains the volumes you want to display. This option is useful if you have a single-disk system.

When RT–11 pauses, it displays *Mount input volume in <device>; Continue?*. Mount the input volume and type Y RETURN to continue the operation. Type N RETURN or press CTRL/C to abort the operation and return control to the monitor.

After RT–11 completes the operation, RT–11 displays the following message prompting you to mount the system volume:

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

Mount the system volume and type Y RETURN. If you type any other response, RT–11 prompts you to mount the system volume until you type Y.

The following sample displays AJAX.DOC from an RL02 disk:

```
.TYPE/WAIT DL0:AJAX.DOC
Mount input volume in DL0:; Continue? Y
```

After RT–11 has displayed AJAX.DOC at the terminal, it issues the following prompt:

```
Mount system volume in DL0:; Continue? Y
```

When you mount the system volume and type Y RETURN, you terminate the TYPE operation.

UNLOAD

(KMON)

- Removes previously loaded handlers from memory, thus freeing the memory space they occupied.
- Removes terminated foreground or system jobs.

Format

UNLOAD $\left\{\begin{array}{l}\textit{device[,device,...]}\\ \textit{jobname[,jobname,...]}\end{array}\right\}$

Parameters

device

Specifies the device handler to be unloaded.

The colon that follows the device handler is optional. However, if you have a system job with the same name as a device handler, you must include the colon to remove that handler, or RT–11 will remove the job rather than the handler. If you do not type a colon, RT–11 checks the table of system jobs for a job with that name before it checks the device table.

jobname

Specifies the job to be unloaded.

Description

You can specify both device handlers and job names on the same command line.

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, RT–11 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 magtape, you must unload and then load (with the LOAD command) the appropriate device handler (MM, MT, or MS).

RT–11 does not accept an UNLOAD command while a foreground job is running if the foreground job owns any units of that device, because a handler the foreground job needs might become nonresident. You can unload a device while a foreground job is running if none of its units belongs to the foreground job.

A special function of this command is to remove a terminated foreground or system job and reclaim memory, since RT–11 does not automatically return the space occupied by the foreground or system job to the free memory list.

Examples

1. This command unloads the foreground job and frees the memory it occupied. The command is valid only if the foreground job is not running:

   ```
   .UNLOAD F
   ```

2. This command unloads the system job QUEUE:

   ```
   .UNLOAD QUEUE
   ```

3. This command clears ownership of all units of DU. If DU is the system device, the DU handler itself remains resident:

   ```
   .UNLOAD DU:
   ```

4. This command releases the line printer and RL02 handlers and frees the area they previously held:

   ```
   .UNLOAD LP:,DL:
   ```

UNPROTECT

(PIP)

Removes a file’s protected status so that you can delete the file.

Format

UNPROTECT *filespec[s]*

Command Options

```
⎧ ⎡ /BEFORE[:date] ⎤ ⎫
⎨ ⎣ /SINCE[:date]  ⎦ ⎬
⎪  /DATE[:date]      ⎪
⎩  /NEWFILES         ⎭
  /EXCLUDE
  /INFORMATION
  /[NO]LOG
  /QUERY
  /SETDATE[:date]
  /SYSTEM
  /WAIT
```

Parameter

filespec[s]

Specifies the file or files whose protected status you want to remove.

You can explicitly specify up to six file specifications. If you specify more than one file, separate the files with commas. You can also use wildcards in the file specifications.

Description

Use the DIRECTORY /PROTECTION and /NOPROTECTION options to determine the protection status of files on a volume. A P next to the block-size number of a file’s directory entry indicates that the file is protected from deletion.

Specifying a Date Argument to an Option

Some of the options accept a date as an argument. The format for specifying the date is:

[dd][:mmm][:yy]

where:

dd specifies the day (a decimal integer in the range 1–31).

mmm specifies the first three characters of the name of the month.

yy specifies the year (a decimal integer in the range 73–99).

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), RT–11 uses the values from the current system date. For example, if you specify only the year 90 and the current system date is May 4, 1991, RT–11 uses the date 4:MAY:90. If the current 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 at 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.)

Options

/BEFORE[:date]

Removes protection from only those files created before the specified date. If no date is specified, the current system date is used. The following command removes the protected status of all MAC files on DK created before March 20, 1991:

```
.UNPROTECT/BEFORE:20:MAR:91 *.MAC
Files unprotected:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

/DATE[:date]

Removes protection from only those files with a certain creation date. If no date is specified, the current system date is used. The following command removes the protected status of all MAC files on DK created on March 20, 1991:

```
.UNPROTECT/DATE:20:MAR:91 *.MAC
Files unprotected:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

/EXCLUDE

Removes protection from all the files on a device except the ones you specify. The following command, for example, removes protection from all files on DU0 except SAV files:

```
.UNPROTECT/EXCLUDE DU0:*.SAV
?PIP-W-No .SYS action
Files unprotected:
DX0:ABC.OLD
DX0:AAF.OLD
DX0:COMB.
DX0:MERGE.OLD
```

/INFORMATION

Changes the severity level of the error message that displays when not all of the input files you specified are found. If you do not use /INFORMATION, RT–11 displays an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, RT–11 displays an informational message to tell you which files it cannot find, but execution continues.

In the following example, RT–11 removes protection from input files FILE1.TXT and FILE3.TXT. However, since RT–11 is unable to find DL0:FILE2.TXT, RT–11 displays a message to inform you:

```
.UNPROTECT/INFORMATION DL0:(FILE1,FILE2,FILE3).TXT
?PIP-I-File not found DL0:FILE2.TXT
```

/[NO]LOG

/LOG lists on the terminal a log of the files from which protection is removed by the current command. This is the default mode of operation when you use wildcards in the file specification. Note that if you specify /LOG, RT–11 does not ask you for confirmation before execution proceeds. Use both /LOG and /QUERY to invoke logging and querying.

/NOLOG suppresses a listing of the files from which protection is being removed from appearing on the terminal.

/NEWFILES

Removes protection from only the files that have the current system date. The following example removes protection from the files created today:

```
.UNPROTECT/NEWFILES DU1:*.BAK
Files unprotected:
DU1:MERGE.BAK ? Y
```

/QUERY

Requests a confirmation message from RT–11 before it removes protection from each file. This option is useful on operations that involve wildcards, when you may not be sure which files RT–11 will select for the operation.

Note: 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 any string beginning with Y) RETURN to initiate execution of a particular operation. RT–11 interprets any other response as NO and does not perform the operation.

The following example shows querying. Protection is removed only from the file DU1:AAF.MAC:

```
.UNPROTECT/QUERY DU1:*.*
Files unprotected:
DU1:ABC.MAC   ? N
DU1:AAF.MAC   ? Y
DU1:MERGE.FOR ? N
```

/SETDATE[:date]

Causes RT–11 to put the date you specify on all files from which it removes protection. If you specify no date, the current system date is used. If the current system date is not set, RT–11 places zeros in the directory entry date position. Normally, RT–11 preserves the existing file creation date when it removes protection from a file.

The following example removes protection from files and changes their dates to the current system date:

```
.UNPROTECT/SETDATE DU0:*.FOR
Files unprotected:
DU0:ABC.FOR
DU0:AAF.FOR
DU0:MERGE.FOR
```

/SINCE[:date]

Removes protection from only those files created on or after the specified date. If no date is specified, the current system date is used. The following command removes protection from all MAC files on DU0 created on or after April 21, 1991:

```
.UNPROTECT/SINCE:21:APR:91 DU0:*.MAC
 Files unprotected:
DU0:A.MAC
DU0:B.MAC
DU0:C.MAC
```

/SYSTEM

Removes protection from system (SYS) files when you use wildcards in the file type. If you omit this option in a wildcard operation, the system files are excluded from the unprotect operation and a message is displayed on the terminal to remind you of this. This example removes protection from all files on DU0 with the file name MM, including SYS files:

```
.UNPROTECT/SYSTEM DU0:MM.*
 Files unprotected:
DU0:MM.MAC
DU0:MM.OBJ
DU0:MM.SAV
DU0:MM.SYS
```

If you specify a SYS file type, you do not need to specify the /SYSTEM option to UNPROTECT system files.

/WAIT

Initiates the UNPROTECT operation but then pauses for you to mount the volume that contains the files whose protection status you want to change. This option is useful if you have a single-disk system.

When RT–11 pauses, it displays *Mount input volume in <device>; Continue?*, where <device> represents the device into which you mount the volume. Mount the volume and type Y RETURN. Type N RETURN or press CTRL/C to abort the operation and return control to the monitor.

When the operation completes, RT–11 displays the *Continue?* message again. Mount the system volume and type Y RETURN. If you type any other response, RT–11 prompts you to mount the system volume until you type Y. RT–11 then displays the monitor prompt.

The following example removes protection from the file FILE.MAC on an RL02 disk:

```
.UNPROTECT/WAIT DL0:FILE.MAC
Mount input volume in DL0:; Continue? Y
DL0:FILE.MAC? Y
Mount system volume in DL0:; Continue? Y
```

V/VRUN

Runs a program as a completely virtual job. This means the job runs as a background job in a pseudo unmapped-monitor environment, often enabling the program to run faster and to use less low-memory space than is otherwise required. See also FRUN, R, RUN, and SRUN.

Format

V *program*

VRUN *program*

Parameter

Program

Specifies the program you want to run. The default device for V is SY, while the default device for VRUN is DK.

Description

The V and VRUN commands both run VBGEXE (the Virtual Background Execution Utility). VBGEXE is called the *virtual* run utility since it appears to extend the amount of low memory available under a mapped monitor. See the *RT–11 System Utilities Manual* for a description of this utility. A program run using VBGEXE is referred to as a completely virtual job or as running in a completely virtual environment.

The V and VRUN commands parallel the R and RUN commands. This means the V command is identical to the VRUN command with one exception:

- The default location for the program run by the V command is the system device.
- The default location for the program run by the VRUN command is the default data device.

With V or VRUN, you can specify any random-access device as the location of the program to be run.

Programs You Can Run with V or VRUN

You can run most utility programs with V or VRUN. If a job is not a valid one for the V or VRUN command, that job does not run and VBGEXE issues the error message:

```
?VBGEXE-F-Cannot run in completely virtual environment <dev:filnam.typ>
```

Making V/VRUN Automatic

You can cause a job to be automatically run as a completely virtual job without using V or VRUN. To do so, include the command SET RUN VBGEXE in your start-up command file (or just issue it), if you are running under a mapped monitor.

If you issue the SET RUN VBGEXE command, then:

- If you use R or RUN (or V or VRUN) and the job or environment is valid for running under VBGEXE, the job runs as a completely virtual job.
- If you use R or RUN and the job or environment is not valid for running under VBGEXE, the monitor attempts to run the job as a straight background job.
- If you use V or VRUN and the job or environment is not valid for running under VBGEXE, VBGEXE issues an error message and the job does not run.
- If you issue an FRUN or SRUN command, the specified job is run as a foreground or a system job. The SET RUN VBGEXE command does not apply to foreground or system jobs.

See the description of the SET RUN [NO]VBGEXE command for further information.

Displaying the RUN State (VBGEXE or NOVBGEXE)

Issue the SHOW CONFIGURATION command to get a system display showing whether RUN is set to VBGEXE or NOVBGEXE.

Examples

The following three examples show three different ways of running the MACRO–11 assembler as a completely virtual job. The progam to be compiled is MYPROG.MAC. The object file is to be named MYPROG.OBJ and the listing file, MYPROG.LST.

1. In the first example, V invokes the assembler which then prompts for CSI command-line input:

   ```
   .V MACRO  RET
   *MYPROG,MYPROG=MYPROG  RET
   *
   ```

2. In the second example, V uses DCL command-line syntax to issue the same command as the preceding one. In this case, since V is a DCL command, RT–11 returns you to the keyboard monitor dot prompt when it finishes execution:

   ```
   .V MACRO MYPROG MYPROG  RET
   .
   ```

3. In the third example, the SET RUN command enables VBGEXE to automatically load and execute programs in a completely vitual environment. Then the MACRO command automatically compiles MYPROG in a virtual environment without it being necessary to use the V command:

   ```
   .SET RUN VBGEXE  RET
   .MACRO MYPROG/LIST:MYPROG  RET
   .
   ```

Appendix A

SL Command-Line Editing Functions

A.1 Basic SL Functions

Table A–1 lists the keypad and keyboard keys you can use with basic command-line editing and their functions.

Table A–1: Basic SL Command-Line Editing Functions

Key	Function
PF1	Changes the function of other keys when used in combination with them. Also referred to as the GOLD key.
PF1 S	Saves the displayed command. The S key is located on your main keyboard and is not case sensitive.
PF1 X	Displays the saved command. The X key is located on your main keyboard and is not case sensitive.
PF2	Displays a screen of helpful SL information on the top half of your screen (above the command line). If you first issue the command SET SL LEARN, subsequently pressing PF2 causes the SL help screen to be locked onto the display. Issue the command SET SL NOLEARN to remove the help screen.
PF4	Delete line function key. Deletes command line from cursor position to end of line. The PF1 PF4 combination restores the deleted line.
↑ or CTRL/E	Moves the cursor backward through the command stack, displaying previous commands. The PF1 ↑ combination displays the next command in the current cycle of commands.
↓ or CTRL/V	Moves the cursor forward through the command stack, displaying previous commands. The PF1 ↓ combination establishes an offset into the stack of previous commands.
← or CTRL/D	Moves the cursor toward the beginning of the command line. The PF1 ← combination moves the cursor to the beginning of the command line.

Table A–1 (Cont.): Basic SL Command-Line Editing Functions

Key	Function
→	Moves the cursor toward the end of a command line. The PF1 → combination moves the cursor to the end of the command line.
LINE FEED or CTRL/J	Deletes the element of command syntax to the left of the cursor. An element of syntax can be the device handler name and unit number, file name, file type or extension, command, any options, and any arguments to those options.
DELETE or ⌫	Deletes the character to the left of the cursor. The PF1 DELETE (or ⌫) combination restores that deleted character or character position.
BACKSPACE or CTRL/H	Switches the character at the current cursor position with the character to the right of the cursor and moves the cursor to the right. The PF1 BACKSPACE combination switches the character at the current cursor position with the character to the left of the cursor and moves the cursor to the left.
CTRL/B	Requires that TT is set to condition NOFB. Same as ↑.
CTRL/F	Requires that TT is set to condition NOFB. Same as →.
CTRL/R or CTRL/W	Redisplays the current command line. Use CTRL/R or CTRL/W to refresh the command line on your terminal screen if the command line becomes garbled or corrupted.
CTRL/U	Deletes all characters from the current cursor position to the beginning of the command line. The PF1 CTRL/U combination restores those deleted characters.
RETURN	Issues the currently displayed command. You can press RETURN and issue a command no matter where the current cursor position is in that command. The PF1 RETURN combination truncates the command (deletes all characters from the cursor position to the end of the command line) and issues the truncated command.

A.2 Keypad SL Functions

Table A–2 summarizes the Keypad SL functionality that you enable with SET SL KED, ON command.

Table A–2: KED Keypad SL Command-Line Editing Functions

Key	Function
PF1	Changes the function of other keys when used in combination with them. Also referred to as the GOLD key.
PF2	Displays a HELP frame on your terminal screen.
PF4	Deletes the command line from the cursor position to the end of the command line. The PF1 PF4 combination restores that deleted command line.
0	Moves the cursor to the beginning of the command line. If the cursor is at the beginning of the command line, 0 moves the cursor to the beginning of next previous or next most recent command line. The PF1 0 combination is the open-line function. Produces a blank line for entering a command.
1	Causes the cursor to move across one element of syntax. An element of syntax can be the device handler name and unit number, file name, file type or extension, command, any options, and any arguments to those options.
2	Moves the cursor to the end of the command line. If the cursor is at the end of the command line, 2 moves the cursor to the end of the next previous or next most recent command line. The PF1 2 combination deletes all characters from the cursor position to the end of the line.
3	Moves the cursor one character position.
4	Causes subsequent cursor movement to be forward.
5	Causes subsequent cursor movement to be in reverse.
, (comma key)	Deletes one character at the cursor position. The PF1 , combination restores that deleted character.
- (hyphen key)	Deletes one element of command syntax at the cursor position. An element of syntax can be the device handler name and unit number, file name, file type or extension, command, any options, and any arguments to those options. The PF1 - combination restores that element of command syntax.
ENTER	Issues the displayed command. Equivalent to the RETURN key.

A.3 Summary of SL Functions

The following tables summarize SL's full editing capabilities with a mapped monitor.

General Editing Functions

Command or Key	Function
CTRL/A	Switches between INSERT and REPLACE modes.
RECALL/ALL	Recalls all previously issued commands.
RECALL *command* or RECALL *number-in-command-stack*	Recalls a previous command.
Arrow Keys	Move the cursor through the command stack or across the command line in the direction of the arrows.

Delete Functions

Key	Function
PF4	Deletes the command line from the cursor position to the end of the command line.
CTRL/U	Deletes the command line from the cursor position to the beginning of the command line.
Keypad Comma Key ,	Deletes one character at the cursor position.
Keypad Hyphen Key -	Deletes one element of command syntax at the cursor position.
LINE FEED or CTRL/J	Deletes the element of command syntax to the left of the cursor.
DELETE Key ⌫	Deletes the character to the left of the cursor.

Restore Functions

Key	Function
PF1 together with the keypad key you pressed to delete information	Restores a command or command element that you deleted with a keypad function.

Movement Functions

Key	Function
CTRL/H	Switches the character at the cursor position with the character to the right of the cursor, also moving the cursor to the right.

Key	Function
PF1 together with ←	Moves the cursor to the beginning of the command line.
PF1 together with → or Keypad 2	Moves the cursor to the end of the preceding command line.
Keypad 1	Moves the cursor across one element of syntax.
Keypad 3	Moves the cursor one character position.
Keypad 4	Causes subsequent cursor movement to be forward.
Keypad 5	Causes subsequent cursor movement to be backward.

Display Functions

Key	Function
CTRL/E or ↑	Displays the previous command.
CTRL/R or CTRL/W	Redisplays the current command line.
PF2	Displays a help listing of SL commands.

Appendix B

Printer Support for the SETUP Command

The printers in the following tables are organized in alphabetical order under the section headers.

B.1 Uniform SETUP Mode and Control Support for Printers

The following SETUP modes and controls act in a uniform manner. That is, the action of the mode or control is consistent for all indicated printers. A *Yes* in the table indicates support for the indicated mode or control. Dashes (—) indicate no support.

Table B–1: Uniform SETUP Modes and Controls for LA50, LA75, LA100, and LA210 Printers

Modes and Controls	LA50	Turbo LA75 and LA75	LA100†	LA210†
[NO]BOLD	Yes	Yes	—	—
CLEAR	Yes	Yes	Yes	Yes
DOWN	Yes	Yes	Yes	Yes
GRAPHIC	Yes	Yes	Yes	—
HELP	Yes	Yes	Yes	Yes
HORIZONTAL	Yes	Yes	—	Yes
LANDSCAPE	—	—	—	—
LANG	Yes	Yes	Yes	—
PAGELENGTH	Yes	Yes	Yes	Yes
PORTRAIT	—	—	—	—
RESET	Yes	Yes	Yes	Yes
[NO]TABS:n	Yes	Yes	Yes	—
TEXT	Yes	Yes	Yes	—
[NO]UNDERLINE	Yes	Yes	Yes	Yes
UP	Yes	Yes	Yes	—
VERTICAL	Yes	Yes	Yes	Yes
[NO]WRAP	Yes	Yes	Yes	—

†Set the LA100 and LA210 printers to AUTO mode before issuing SETUP commands.

Table B–2: Uniform SETUP Modes and Controls for LA324, LA1100, and LA2100 Printers

Modes and Controls	LA324	LA1100	LA2100
[NO]BOLD	Yes	Yes	Yes
CLEAR	Yes	Yes	Yes
DOWN	Yes	Yes	Yes
GRAPHIC	Yes	—	—
HELP	Yes	Yes	Yes
HORIZONTAL	Yes	Yes	—
LANDSCAPE	—	Yes	—
LANG	Yes	Yes	—
PAGELENGTH	Yes	Yes	Yes
PORTRAIT	—	Yes	—
RESET	Yes	Yes	Yes
[NO]TABS:n	Yes	Yes	Yes
TEXT	Yes	Yes	—
[NO]UNDERLINE	Yes	Yes	Yes
UP	Yes	—	—
VERTICAL	Yes	—	—
[NO]WRAP	Yes	Yes	Yes

Table B–3: Uniform SETUP Modes for LG01, LG02, LJ250, LN03, and LQP02 Printers

Modes and Controls	LG01	LG02	LJ250	LN03	LQP02
[NO]BOLD	—	—	Yes	Yes	Yes
CLEAR	—	—	Yes	Yes	Yes
DOWN	—	Yes	Yes	Yes	Yes
GRAPHIC	—	—	—	—	—
HELP	Yes	Yes	Yes	Yes	Yes
HORIZONTAL	Yes	Yes	Yes	Yes	—
LANDSCAPE	—	Yes	—	Yes	—
LANG	—	—	—	Yes	—

Table B–3 (Cont.): Uniform SETUP Modes for LG01, LG02, LJ250, LN03, and LQP02 Printers

Modes and Controls	LG01	LG02	LJ250	LN03	LQP02
PAGELENGTH	Yes	Yes	Yes	Yes	Yes
PORTRAIT	—	Yes	—	Yes	—
RESET	Yes	Yes	Yes	Yes	Yes
[NO]TABS:n	—	—	—	Yes	Yes
TEXT	—	Yes	Yes	Yes	—
[NO]UNDERLINE	Yes	Yes	Yes	Yes	Yes
UP	—	Yes	Yes	—	—
VERTICAL	Yes	Yes	Yes	—	—
[NO]WRAP	—	Yes	Yes	Yes	Yes

B.2 Nonuniform SETUP Mode and Control Support for Printers

The following SETUP modes and controls act in a manner specific to particular printers. That is, the action of the mode or control depends on the printer you are using. Dashes (—) indicate no support.

Table B–4: Nonuniform SETUP Modes and Controls for LA50, LA75, LA100, and LA210 Printers

Modes and Controls	LA50	Turbo LA75 and LA75†	LA100‡	LA210‡
DENSE	—	Medium Density	Letter Mode, High Density	—
DRAFT	Draft	Draft	Draft	Draft
LETTER	Hor:10	Medium Density Hor:10	Letter Mode Medium Density Hor:10	Hor:10
LISTING	Draft Hor:16.5	Draft Hor:16.5	Draft Hor:16.5	Draft Hor:16.5
MEMO	Enhanced Density	High Density Hor:10	Letter Mode Medium Density	Letter Mode Medium Density

†At default setting (NLQ and LQ panel lights off). In NLQ or LQ mode, the set (panel) mode overrides SETUP character density modes such as DENSE, DRAFT, and so on.

‡Set the LA100 and LA210 printers to AUTO mode before issuing SETUP commands.

Table B–5: Nonuniform SETUP Modes and Controls for LA324, LA1100, and LA2100 Printers

Modes and Controls	LA324	LA1100	LA2100
DENSE	Letter Mode High Density	Letter Mode High Density	—
DRAFT	Draft	Draft	—
LETTER	Letter Mode Medium Density Hor:10	Letter Mode Medium Density Hor:12	Hor:10
LISTING	Draft Hor:18	Draft Hor:18	Hor:16.5
MEMO	Letter Mode Medium Density	Letter Mode Medium Density	—

Table B–6: Nonuniform SETUP Modes for LG01, LG02, LJ250, LN03, and LQP02 Printers

Modes and Controls	LG01	LG02	LJ250	LN03	LQP02
DENSE	—	—	—	—	—
DRAFT	—	—	—	—	—
LETTER	Hor:10	Hor:10	Hor:10	Hor:10	Hor:10
LISTING	Hor:16.5	Hor:16.5	Hor:16.5	Hor:16.5	Hor:12
MEMO	—	—	—	—	—

Appendix C

National Replacement Character Sets

C.1 National Replacement Character Set Modes

The SETUP program lets you set modes for national replacement character (NRC) sets for Professional 300 series processors. The following SETUP modes can be set using the SETUP LANG:code command:

COMPOSE
NOCOMPOSE — When you set COMPOSE, you can combine keyboard keys to create composed characters from an NRC set.

When you set NOCOMPOSE, the Compose key does not function.

DATA
TYPE — When you set DATA, the data processing version of the current keyboard is used. When you set TYPE, the typewriter version of the current NRC set is used.

C.2 Compose Sequences and Dead Diacritical Keys for NRC Language Keyboards

This section describes two methods for composing characters on Professional 300 series processors. The first method uses the Compose key and compose sequences. The other method described uses dead diacritical keys to compose characters. Compose sequences and dead diacritical keys vary among NRC set language keyboards. The compose sequences and dead diacritical keys are listed in Section C.3.

C.2.1 Composing Characters Using the Compose Key

To create a composed character, press the Compose key. The red Compose indicator at the top of the keyboard should light. Next, type the two characters that together form the character you want to create. Neither character that you type is displayed on the screen; only the resulting composed character is displayed. The Compose indicator should go out.

If you type the wrong character after pressing the Compose key, press the ⌫ key to cancel the compose sequence. If you type an invalid sequence, the keyboard beeps and the compose sequence is canceled.

Valid compose sequences are determined by the keyboard language selection. Use the SETUP KB LANG:code command to make the keyboard language selection (see the SETUP LANG:code command).

C.2.2 Composing Characters Using Dead Diacritical Keys

Dead diacritical keys are used to compose characters. Dead diacritical keys are used only with other character keys; they do not function by themselves. Dead diacritical keys are listed in Section C.3 for each NRC set.

To create a composed character, first press a dead diacritical key. The red Compose indicator at the top of the keyboard should light. Next, press a character key. Neither character that you type is displayed on the screen; only the resulting composed character is displayed. The Compose indicator should go out when you press a character key.

If you press a dead diacritical key while creating a composed character, using the Compose key, the dead diacritical key functions as a normal character key in the compose sequence.

C.3 National Replacement Character Set Language Keyboards

This section describes the national replacement character (NRC) set language keyboards for Professional 300 series processors.

Keyboards are supported for the following countries:

Australia
Austria
Belgium
Canada (English)
Canada (French)
Denmark
Finland
France
Germany
Ireland
Italy
Netherlands
New Zealand
Norway
Spain
Sweden
Switzerland (French)
Switzerland (German)
United Kingdom
United States

Each keyboard is described in a table. The tables have the following headings:

Character	Lists characters created by compose sequence.

On Keyboard Data Indicates whether or not the character created by the compose sequence is on the keyboard in DATA mode.

Type Inidicates whether or not the character created by the compose sequence is on the keyboard in TYPE mode.

Sequence Lists the compose sequences used to create the composed character.

Dead Diacritical Keys Lists dead diacritical keys supported on the keyboard for that country.

Unimplemented Keys Lists keys that are ignored.

USASCII Indicates character is US character set.

NRC Indicates character is in National Replacement set.

Each language keyboard has two modes for setting characters: DATA mode and TYPE mode. DATA mode sets certain characters that are appropriate for program development. DATA mode is the default mode setting. TYPE mode sets certain characters that are more appropriate for word processing.

The keyboard tables are divided into DATA and TYPE modes. The characters that are set when you specify DATA or TYPE mode are listed in the corresponding sections in each language keyboard table.

C.3.1 North American Keyboard

Language Code: AUS, CDNE, NZ, or USA.

Table C–1: North American Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	Y
’	space ’	Y	Y
@	a a	Y	Y
[	((	Y	Y
\	/ /	Y	Y
]	))	Y	Y
^	space ^	Y	Y
‘	space ‘	Y	Y
{	(-	Y	Y
|	/ ^	Y	Y
}	) -	Y	Y
~	space ~	Y	Y

Dead Diacritical Keys: NONE

Unimplemented Keys: NONE

C.3.2 United Kingdom Keyboard

Table C–2: United Kingdom Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	Y
’	space ’	Y	Y
@	a a	Y	Y
[	((	Y	Y
\	/ /	Y	Y
]	))	Y	Y
^	space ^	Y	Y
‘	space ‘	Y	Y
{	(-	Y	Y
|	/ ^	Y	Y
}	) -	Y	Y
~	space ~	Y	Y
£	- L	N	Y
Dead Diacritical Keys			
		^	^
		‘	‘
		~	~
Unimplemented Keys			
		1/4	1/4
		1/2	1/2
		o	o
			§
		USASCII	NRC
		#	£

C.3.3 Belgian or French Language Keyboard

Language Code: B or F

Table C–3: Belgian or French Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	Y
’	space ’	Y	Y
@	a a	Y	Y
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	N	N
}	) -	N	N
~	space ~	N	N
£	- L	N	N
§	o s	N	Y
o	O ^	Y	Y
à	‘ a	Y	Y
ç	, c	Y	Y
è	‘ e	N	Y
é	’ e	Y	Y
ù	‘ u	N	Y
Dead Diacritical Keys			
		"	"
		‘	‘
		^	^
		~	~
Unimplemented Keys: NONE			
		USASCII	NRC
		#	£
		@	a
		[	°
		\	ç
		]	§
		{	e
		|	ù
		}	è
		~	"

C.3.4 Canadian (French) Language Keyboard

Language Code: CDNF

Table C–4: Canadian (French) Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	Y
’	space ’	Y	Y
@	a a	Y	Y
[	((	Y	N
\	/ /	Y	Y
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	Y	Y
}	) -	N	N
~	space ~	N	N
à	‘ a	N	N
â	^ a	N	N
ç	, c	N	Y
è	e	Y	Y
é	’ e	Y	Y
ê	^ e	N	N
î	^ i	N	N
ô	^ o	N	N
ù	‘ u	N	N
û	^ u	N	N
Dead Diacritical Keys			
		‘	‘
		^	^
		~	~
Unimplemented Keys			
		E	E
		o	o
			ç
		USASCII	NRC
		@	a
		[	
		\	ç
		]	ê
		^	
		{	é
		|	ù
		}	è
		~	û

C.3.5 Danish Language Keyboard

Language Code: DK

Table C–5: Danish Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	N
’	space ’	Y	Y
@	a a	Y	Y
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	((-	N	N
|	/ ^	N	N
}	) -	N	N
~	space ~	N	N
Ä	" A	N	N
Â	* A	N	Y
Æ	A E	Y	Y
Ø	/ O	N	Y
Ü	" U	N	N
ä	" a	N	N
â	* a	N	Y
æ	a e	Y	Y
ø	/ o	N	Y
ü	" u	N	N
Dead Diacritical Keys			
		"	"
		,	,
		^	^
		‘	‘
		~	~
Unimplemented Keys			
		USASCII	NRC
		@	Ä
		[	Æ
		\	Ø
		]	Â
		^	Ü
			ä
		{	æ
		|	Ø
		}	â
		~	û

C.3.6 Finnish Language Keyboard

Language Code: SF

Table C–6: Finnish Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	N
’	space ’	Y	Y
@	a a	Y	Y
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	N	N
}	) -	N	N
~	space ~	N	N
Ä	" A	Y	Y
Â	* A	N	Y
ö	" O	Y	Y
Ø	/ O	N	Y
Ü	" U	N	Y
ä	" a	Y	Y
â	* a	N	Y
é	’ e	N	N
ö	" o	Y	Y
ü	" u	N	Y
Dead Diacritical Keys			
		^	^
		‘	‘
		~	~
Unimplemented Keys			
		o	o
		§	§
		USASCII	NRC
		[	Ä
		\	ö
		]	Â
		^	Ü
			é
		}	â
		~	û

C.3.7 German Language Keyboard

Language Code: A or D

Table C–7: German Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	Y
’	space ’	Y	Y
@	a a	Y	N
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	N	N
}	) -	N	N
~	space ~	N	N
§	o s	Y	Y
Ä	" A	N	Y
ö	" O	Y	Y
ü	" U	N	Y
ä	" a	N	Y
ö	" o	Y	Y
ß	s s	Y	Y
ü	" u	N	Y
Dead Diacritical Keys			
		,	,
		^	^
		‘	‘
		~	~
Unimplemented Keys: NONE			
		USASCII	NRC
		@	§
		[	Ä
		\	ö
		]	ü
		{	ä
		|	ö
		}	ü
		~	ß

C.3.8 Dutch Language Keyboard

Language Code: NL

Table C–8: Dutch Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	Y
’	space ’	Y	Y
@	a a	Y	Y
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	Y	Y
}	) -	N	N
~	space ~	N	N
£	- L	N	Y
	i j	N	N
	- F	N	N
Dead Diacritical Keys			
		"	"
		,	,
		^	^
		‘	‘
		~	~
Unimplemented Keys			
		o	o
			§
		USASCII	NRC
		#	£
		@	3/4
		[	ij
		\	1/2
		]	|
		{	"
		|	f
		}	1/4
		~	’

C.3.9 Italian Language Keyboard

Language Code: I

Table C–9: Italian Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	N
’	space ’	Y	Y
@	a a	Y	N
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	N	N
}	) -	N	N
~	space ~	N	N
£	- L	N	Y
§	o s	N	Y
o	O ^	Y	Y
à	‘ a	N	Y
ç	, c	N	Y
è	‘ e	Y	Y
é	’ e	N	Y
ì	‘ i	Y	Y
ò	‘ o	Y	Y
Dead Diacritical Keys			
		^	^
		‘	‘
		~	~
Unimplemented Keys: NONE			
		USASCII	NRC
		#	£
		@	§
		[	o
		\	§
		]	é
		‘	ù
		{	à
		|	ò
		}	è
		~	ì

C.3.10 Norwegian Language Keyboard

Language Code: N

Table C–10: Norwegian Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	N
’	space ’	Y	Y
@	a a	Y	Y
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	N	N
}	) -	N	N
~	space ~	N	N
Ä	" A	N	N
Â	* A	N	Y
Æ	A E	Y	Y
Ø	/ O	N	Y
ü	" U	N	N
ä	" a	N	N
â	* a	N	Y
æ	a e	Y	Y
ø	/ o	N	Y
ü	" u	N	N
Dead Diacritical Keys			
		"	"
		,	,
		^	^
		‘	‘
		~	~
Unimplemented Keys			
		USASCII	NRC
		@	Ä
		[	Æ
		Ø	O
		]	Â
		^	Ü
			ä
		{	æ
		|	Ø
		}	â
		~	û

C.3.11 Spanish Language Keyboard

Language Code: SP or E

Table C–11: Spanish Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	N
’	space ’	Y	Y
@	a a	Y	N
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	N	N
}	) -	N	N
~	space ~	N	N
¡	!!	N	Y
£	- L	N	Y
§	o s	N	Y
o	O ^	Y	Y
¿	??	N	Y
Ñ	N ~	Y	Y
ç	, c	N	Y
ñ	n ~	Y	Y
Dead Diacritical Keys			
		"	"
		’	’
		^	^
		‘	‘
		~	~
Unimplemented Keys: NONE			
		a_ _	a_ _
		o_ _	o_ _
		USASCII	NRC
		#	£
		@	§
		[	¡
		\	Ñ
		]	¿
		{	o
		|	ñ
		{	ç

C.3.12 Swedish Language Keyboard

Language Code: S

Table C–12: Swedish Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	N
’	space ’	Y	Y
@	a a	Y	N
]	((	Y	N
\	/ /	Y	N
[	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
}	(-	N	N
|	/ ^	N	N
{	) -	N	N
~	space ~	N	N
Ä	" A	Y	Y
Â	* A	N	Y
É	’ E	N	N
ö	" O	Y	Y
ü	" U	N	Y
ä	* a	N	Y
â	" a	Y	Y
é	’ e	N	N
ö	" o	Y	Y
ü	" u	N	Y
Dead Diacritical Keys			
		^	^
		‘	‘
		~	~
Unimplemented Keys			
		o	o
			§
		USASCII	NRC
		@	É
		[	Ä
		\	ö
		]	Â
		^	ü
		‘	é
		{	ä
		|	
		}	
		~	ü

C.3.13 Swiss (French) Language Keyboard

Language Code: CHF

Table C–13: Swiss (French) Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	N
’	space ’	Y	Y
@	a a	Y	N
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	N	N
}	) -	N	N
~	space ~	N	N
á	‘ a	Y	Y
ä	" a	N	Y
ç	, c	N	Y
è	‘ e	Y	Y
é	’ e	Y	Y
ê	^ e	N	N
î	^ i	N	N
ô	^ o	N	N
ö	" o	N	Y
ú	’ u	N	N
û	^ u	N	N
ü	" u	N	Y
Dead Diacritical Keys			
		"	"
		‘	‘
		^	^
		~	~
Unimplemented Keys			
			£
		USASCII	NRC
		#	£
		@	a
		[	o
		\	ç
		]	§
		{	ê
		|	ù
		}	è
		~	"

C.3.14 Swiss (German) Language Keyboard

Language Code: CHD

Table C–14: Swiss (German) Language Keyboard

		On Keyboard	
Character	**Sequence**	**Data**	**Type**
"	space "	Y	Y
#	+ +	Y	N
’	space ’	Y	Y
@	a a	Y	N
[	((	Y	N
\	/ /	Y	N
]	))	Y	N
^	space ^	N	N
‘	space ‘	N	N
{	(-	N	N
|	/ ^	N	N
}	) -	N	N
~	space ~	N	N
á	‘ a	N	Y
ä	" a	N	Y
ç	, c	N	Y
è	‘ e	N	Y
é	’ e	N	Y
ê	^ e	N	N
î	^ i	N	N
ô	^ o	N	N
ö	" o	Y	Y
ú	’ u	N	N
û	^ u	N	N
ü	" u	Y	Y
Dead Diacritical Keys			
		"	"
		‘	‘
		^	^
		~	~
Unimplemented Keys			
		USASCII	NRC
		@	§
		[	Ä
		\	ö
		]	ü
		{	ä
		|	ö
		}	ü
		~	ß

Index