RT–11 System User's Guide

AA–5279C–TC

March 1983

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

This manual supersedes the *RT–11 System User's Guide*, AA–5279B–TC. This manual contains Update Notice 1, AD–5279C–T1.

Operating System: RT–11 Version 5.1

To order additional documents from within DIGITAL, contact the Software Distribution Center, Northboro, Massachusetts 01532.

To order additional documents from outside DIGITAL, refer to the instructions at the back of this document.

digital equipment corporation · maynard, massachusetts

First Printing, March 1980
Updated, March 1981
Revised, March 1983
Updated, July 1984

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

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

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

© Digital Equipment Corporation 1980, 1981, 1983, 1984.
All Rights Reserved.

Printed in U.S.A.

A postage-paid READER'S COMMENTS form is included on the last page of this document. Your comments will assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

	DIGITAL™	
DEC	MASSBUS	RT–11
DECmate	PDP	UNIBUS
DECsystem–10	P/OS	VAX
DECSYSTEM–20	Professional	VMS
DECUS	Rainbow	VT
DECwriter	RSTS	Work Processor
DIBOL	RSX	

Contents

Page

Preface xi

Part I RT–11 Overview

Chapter 1 System Components

1.1 Hardware 1–1
1.2 Software 1–2
1.2.1 Monitors 1–3
1.2.1.1 Single-Job (SJ) Monitor 1–3
1.2.1.2 Foreground/Background (FB) Monitor 1–4
1.2.1.3 Extended Memory (XM) Monitor 1–4
1.2.2 Device Handlers 1–5
1.2.3 System Utility Programs 1–6
1.2.3.1 Editing 1–6
1.2.3.2 General Purpose 1–6
1.2.3.3 System Jobs 1–8
1.2.3.4 Debugging and Patching 1–9
1.2.3.5 BATCH 1–9
1.2.4 Language Processors 1–9
1.3 RT–11 Software Documentation 1–9
1.4 System Services 1–10
1.4.1 Keyboard Monitor Commands 1–10
1.4.2 System Programs 1–11
1.4.3 The Relationship Between Complex Commands and System Programs 1–11
1.4.4 The System Macro Library and Programmed Requests 1–12
1.4.5 SYSLIB FORTRAN-Callable Subprograms 1–12

Chapter 2 Program Development

2.1 Using an Editor (EDIT, KED, KEX, OR K52) 2–1
2.2 Using the Assembler (MACRO) 2–1
2.3 Using the Linker (LINK) 2–2
2.4 Using the Debugger (ODT or VDT) 2–2

2.5 Using the Librarian (LIBR) 2–2
2.6 Using a High-Level Language (FORTRAN, BASIC, or DIBOL) 2–3

Part II System Communication

Chapter 3 System Conventions

3.1 Start-Up Procedure 3–1
3.2 Data Formats 3–2
3.3 Device Names 3–3
3.4 File Names and File Types 3–4
3.5 Device Structures 3–6
3.6 Special Function Keys 3–6
3.7 Foreground/Background Terminal I/O 3–9
3.8 Type-Ahead Feature 3–10

Chapter 4 Keyboard Commands

4.1 Command Syntax 4–1
4.1.1 Factoring File Specifications 4–4
4.1.2 File Type Specification 4–5
4.1.3 Abbreviating Keyboard Commands 4–5
4.1.4 Keyboard Prompts 4–5
4.2 Wildcards 4–6
4.3 Editing Command Lines and Terminal Input 4–9
4.3.1 The GOLD Key (PF1) 4–10
4.3.2 The Help Key (PF2) 4–10
4.3.3 Moving the Cursor 4–11
4.3.4 Reproduce Last Command Executed 4–12
4.3.5 Delete Line from Cursor to End of Line 4–13
4.3.6 Restore Last Line Deleted 4–13
4.3.7 Delete One Character to Left of Cursor 4–13
4.3.8 Switch Positions of Two Characters 4–14
4.3.9 Delete All Characters to Left of Cursor 4–14
4.3.10 Truncate and Execute Command Line 4–14
4.3.11 Execute Entire Command Line 4–15
4.3.12 Redisplay Current Line 4–15
4.3.13 Redisplay Line Before Last Line Executed 4–15
4.4 Defining Commands with Distributed DCLSAV 4–15
4.4.1 Command Definition Syntax 4–15
4.4.2 Defining Command Abbreviations 4–16
4.4.3 Displaying Command Definitions 4–16.1
4.4.4 Adding Information from a Command Line 4–16.1
4.4.5 Redefining Commands 4–16.1
4.4.6 Deleting Commands 4–16.2
4.5 Indirect Files 4–16.2
4.5.1 Creating Indirect Files 4–16.2
4.5.2 Executing Indirect Files 4–19
4.5.3 Start-Up Indirect Files 4–22

4.6 Keyboard Monitor Commands 4–22
ABORT 4–24
ASSIGN 4–25
B 4–27
BACKUP 4–28
BOOT 4–32
CLOSE 4–34
COMPILE 4–35
COPY 4–42
CREATE 4–58
D 4–60
DATE 4–61
DEASSIGN 4–62
DELETE 4–63
DIBOL 4–68
DIFFERENCES 4–72
DIRECTORY 4–80
DISMOUNT 4–92
DUMP 4–93
E 4–98
EDIT 4–99
EXECUTE 4–103
FORMAT 4–113
FORTRAN 4–115
FRUN 4–124
GET 4–127
GT 4–128
HELP 4–130
INITIALIZE 4–132
INSTALL 4–138
LIBRARY 4–139
LINK 4–146
LOAD 4–155
MACRO 4–157
MOUNT 4–163
PRINT 4–165
PROTECT 4–171
R 4–175
REENTER 4–176
REMOVE 4–177
RENAME 4–178
RESET 4–183
RESUME 4–184
RUN 4–186
SAVE 4–187
SET 4–189
SHOW 4–207
SQUEEZE 4–216
SRUN 4–218
START 4–220
SUSPEND 4–221
TIME 4–222
TYPE 4–223

UNLOAD 4–227
UNPROTECT 4–229
4.7 Concise Command Language (CCL) 4–233

Chapter 5 Indirect Control File Processor (IND)

5.1 Creating an Indirect Control File 5–1
5.1.1 Labels 5–2
5.1.2 IND Directives and Keyboard Commands 5–3
5.1.2.1 IND Directives 5–3
5.1.2.2 Keyboard Commands 5–4
5.1.3 Comments 5–5
5.2 Executing Indirect Control Files 5–6
5.2.1 IND Options 5–7
5.2.1.1 Delete Control File Option (/D) 5–7
5.2.1.2 Suppress Keyboard Commands Option (/N) 5–8
5.2.1.3 Suppress Console Display Option (/Q) 5–8
5.2.1.4 Command Tracing Option (/T) 5–8
5.2.2 Passing Parameters 5–8
5.2.3 Nested Indirect Control Files 5–9
5.2.4 Executing Indirect Command Files from Control Files 5–10
5.3 Directive Summary 5–11
5.4 Symbols 5–16
5.4.1 Local and Global Symbols 5–17
5.4.2 Logical Symbols 5–17
5.4.3 Numeric Symbols 5–17
5.4.3.1 Defining the Radix of a Numeric Symbol 5–17
5.4.3.2 Numeric Expressions 5–18
5.4.4 String Symbols 5–19
5.4.5 Special Symbols 5–19
5.4.6 Symbol Value Substitution 5–22
5.5 IND Directives 5–23
5.5.1 Define a Label (.label) 5–23
5.5.1.1 Label Processing 5–24
5.5.1.2 Direct Access Labels 5–24
5.5.2 Define Logical End of File (/) 5–24
5.5.3 .ASK Directive 5–25
5.5.3.1 Syntax 5–25
5.5.3.2 Question Display 5–26
5.5.3.3 Responses 5–26
5.5.4 .ASKN Directive 5–27
5.5.4.1 Syntax 5–27
5.5.4.2 Determining the Radix 5–28
5.5.4.3 Question Display 5–29
5.5.4.4 Responses 5–29

5.5.5 .ASKS Directive 5–30
5.5.5.1 Syntax 5–30
5.5.5.2 Determining the Radix of Range and Timeout Values 5–32
5.5.5.3 Question Display 5–32
5.5.5.4 Responses 5–32
5.5.6 Begin Block (.BEGIN) 5–33
5.5.7 Chain to Another File (.CHAIN) 5–34
5.5.8 Close File (.CLOSE) 5–34

5.5.9 Send Data to File (.DATA) . . . 5-34
5.5.10 Decrement Numeric Symbol (.DEC) . . . 5-35
5.5.11 Delay Execution (.DELAY) . . . 5-36
5.5.12 Disable Option (.DISABLE) . . . 5-36
5.5.13 Display Symbol Table (.DUMP) . . . 5-37
5.5.14 Enable Option (.ENABLE) . . . 5-39
5.5.15 End Block (.END) . . . 5-46
5.5.16 Delete Symbols (.ERASE) . . . 5-46
5.5.17 Exit Current Control File (.EXIT) . . . 5-47
5.5.18 Call a Subroutine (.GOSUB) . . . 5-48
5.5.19 Branch to a Label (.GOTO) . . . 5-48
5.5.20 Logical Tests . . . 5-49
5.5.20.1 Test If Symbol Meets Specified Condition (.IF) . . . 5-49
5.5.20.2 Test If Symbol Is Defined or Not Defined (.IFDF/.IFNDF) . . . 5-50
5.5.20.3 Test If Operating Mode Is Enabled or Disabled (.IFENABLED/.IFDISABLED) . . . 5-50
5.5.20.4 Test If Device Is Loaded (.IFLOA/.IFNLOA) . . . 5-51
5.5.20.5 Test If Symbol Is True or False (.IFT/.IFF) . . . 5-51
5.5.20.6 Compound Tests . . . 5-53
5.5.21 Increment Numeric Symbol (.INC) . . . 5-53
5.5.22 Branch on Error (.ONERR) . . . 5-54
5.5.23 Opening Data Files . . . 5-55
5.5.23.1 Open File (.OPEN) . . . 5-55
5.5.23.2 Open File for Append (.OPENA) . . . 5-56
5.5.23.3 Open File for Read (.OPENR) . . . 5-56
5.5.24 Parse a String (.PARSE) . . . 5-56
5.5.25 Purge File (.PURGE) . . . 5-58
5.5.26 Read a Record (.READ) . . . 5-58
5.5.27 Return from a Subroutine (.RETURN) . . . 5-59
5.5.28 Set Numeric Symbol to Decimal or Octal (.SETD/.SETO) . . . 5-59
5.5.29 Set Symbol to Logical Value (.SETL) . . . 5-59
5.5.30 Set Symbol to Numeric Value (.SETN) . . . 5-60
5.5.31 Set Symbol to String Value (.SETS) . . . 5-61
5.5.32 Set Symbol to True or False (.SETT/.SETF) . . . 5-62
5.5.32.1 .SETT Directive . . . 5-62
5.5.32.2 .SETF Directive . . . 5-63
5.5.33 Terminate Processing (.STOP) . . . 5-63
5.5.34 Test a Symbol (.TEST) . . . 5-64
5.5.35 Test for Installed Device (.TESTDEVICE) . . . 5-65
5.5.36 Test for File (.TESTFILE) . . . 5-67
5.5.37 Obtain Volume Identification (.VOL) . . . 5-67

Part III Text Editing

Chapter 6 Text Editor (EDIT)

6.1 Calling EDIT . . . 6-1
6.2 Modes of Operation . . . 6-1
6.3 Special Key Commands . . . 6-2
6.4 Command Structure . . . 6-4

6.4.1 Arguments . . . 6-5
6.4.2 Command Strings . . . 6-6
6.4.3 Current Location Pointer . . . 6-7
6.4.4 Character- and Line-Oriented Command Properties . . . 6-7
6.4.4.1 Character-Oriented Commands . . . 6-7
6.4.4.2 Line-Oriented Commands . . . 6-8
6.4.5 Command Repetition . . . 6-9
6.5 Memory Usage . . . 6-11
6.6 Editing Commands . . . 6-12
6.6.1 File Open and Close Commands . . . 6-12
6.6.1.1 Edit Read . . . 6-12
6.6.1.2 Edit Write . . . 6-13
6.6.1.3 Edit Backup . . . 6-14
6.6.1.4 End File . . . 6-14
6.6.2 File Input/Output Commands . . . 6-15
6.6.2.1 Read . . . 6-15
6.6.2.2 Write . . . 6-16
6.6.2.3 Next . . . 6-18
6.6.2.4 Exit . . . 6-19
6.6.3 Pointer Relocation Commands . . . 6-19
6.6.3.1 Beginning . . . 6-20
6.6.3.2 Jump . . . 6-20
6.6.3.3 Advance . . . 6-21
6.6.4 Search Commands . . . 6-22
6.6.4.1 Get . . . 6-22
6.6.4.2 Find . . . 6-23
6.6.4.3 Position . . . 6-24
6.6.5 Text Listing Commands . . . 6-24
6.6.5.1 List . . . 6-24
6.6.5.2 Verify . . . 6-26
6.6.6 Text Modification Commands . . . 6-26
6.6.6.1 Insert . . . 6-26
6.6.6.2 Delete . . . 6-27
6.6.6.3 Kill . . . 6-28
6.6.6.4 Change . . . 6-29
6.6.6.5 Exchange . . . 6-30
6.6.7 Utility Commands . . . 6-31
6.6.7.1 Save . . . 6-31
6.6.7.2 Unsave . . . 6-32
6.6.7.3 Macro . . . 6-33
6.6.7.4 Execute Macro . . . 6-34
6.6.7.5 Edit Version . . . 6-34
6.6.7.6 Uppercase and Lowercase Commands . . . 6-35
6.7 Display Editor . . . 6-36
6.7.1 Using the Display Editor . . . 6-37
6.7.2 Immediate Mode . . . 6-38

6.8 EDIT Example 6–39
6.9 EDIT Error Conditions 6–40

Part IV Setting Hardware Characteristics

Chapter 7 Hardware Characteristics Program (SETUP)

7.1 SETUP Commands 7–1
7.1.1 Mode Commands 7–1
7.1.2 Control Commands 7–2
7.2 SETUP Modes 7–2
7.2.1 Terminal Modes 7–2
7.2.2 Terminal and Printer Modes 7–7
7.2.3 Clock Modes 7–11
7.3 SETUP Controls 7–13
7.4 Summary of SETUP Modes and Controls 7–14

Appendix A Monitor Command Abbreviations and System Utility Program Equivalents

Index

Figures

2–1 Program Development Cycle 2–3
4–1 Sample Command Syntax 4–3
4–2 Format of a 12-Bit Binary Number 4–190
5–1 Indirect Control File Line Elements 5–1
6–1 Display Editor Format, 12-Inch Screen 6–36

Tables

1–1 RT–11 Hardware Components 1–2
3–1 Permanent Device Names 3–3
3–2 Standard File Types 3–4
3–3 Device Structures 3–7
3–4 Special Function Keys 3–7
4–1 Commands Supporting Wildcards 4–7
4–2 Wildcard Defaults 4–8
4–3 Single-Line Editor Function Keys 4–11
4–4 DIRECTORY Sort Categories 4–89
4–5 Verification Bit Patterns 4–115
4–6 FORTRAN Listing Codes 4–122
4–7 Display Screen Values 4–129
4–8 Default Directory Sizes 4–136
4–9 Execution and Prompting Sequence of LIBRARY Options 4–144
4–10 Prompting Sequence for LINK Options 4–147

4–11 Cross-Reference Sections 4–158
4–12 .DSABL and .ENABL Directive Summary 4–158
4–13 .LIST and .NLIST Directive Summary 4–161
4–14 SET Device Conditions and Modifications 4–199
5–1 IND Options 5–7
5–2 IND Directive Summary 5–11
5–3 Operating Modes 5–14
5–4 Special IND Characters 5–15
5–5 Arithmetic, Logical and Relational Operators 5–16
5–6 IND Special Symbols 5–20
5–7 IND Operating Modes 5–40
5–8 Errors Intercepted by .ONERR Directive 5–54
6–1 EDIT Key Commands 6–2
6–2 EDIT Command Categories 6–4
6–3 Command Arguments 6–5
6–4 EDIT Commands and File Status 6–15
6–5 Write Command Arguments 6–17
6–6 Jump Command Arguments 6–20
6–7 Advance Command Arguments 6–21
6–8 List Command Arguments 6–25
6–9 Delete Command Arguments 6–27
6–10 Kill Command Arguments 6–28
6–11 Change Command Arguments 6–29
6–12 Exchange Command Arguments 6–31
6–13 Unsave Command Arguments 6–32
6–14 M Command and Arguments 6–33
6–15 Immediate Mode Commands 6–39

Preface

This manual describes how to use the RT–11 operating system; it provides enough information for you to perform ordinary tasks such as program development, program execution, and file maintenance.

The manual is written for you if you are already familiar with computer software fundamentals and have some experience using RT–11. If you have no RT–11 experience, you should first read the *Introduction to RT–11* before consulting this manual. If you have experience with an earlier release of RT–11 (this is Version 5), you should read the *RT–11 System Release Notes* to learn how RT–11 Version 5 differs from earlier versions. You can also read the *RT–11 System Utilities Manual* to learn how to use the RT–11 system utilities that perform the keyboard commands described in this manual. If you are interested in more sophisticated programming techniques or in system programming, you should read this manual first and then proceed to the *RT–11 Programmer's Reference Manual* and the *RT–11 Software Support Manual*.

The next section, Chapter Summary, briefly describes the chapters in this manual and suggests a reading path to help you use the manual efficiently.

Chapter Summary

Part I, RT–11 Overview, Chapter 1 and 2, describes the RT–11 operating system in general. It lists the hardware and software components of the RT–11 system, describes the monitors, and explains the program development process with RT–11.

Part II, System Communication, Chapters 3 through 5, describes system conventions, such as data formats, physical device names, file naming conventions, and special function key commands. Chapter 4 introduces the keyboard monitor commands that you use to communicate with the monitor and to perform system jobs. Chapter 5 describes how to use IND, the indirect control file processor.

Part III, Text Editing, Chapter 6, describes the RT–11 text editor, (EDIT) and shows you how to create and modify files with it.

Part IV, Setting Hardware Characteristics (SETUP), shows you how to set hardware characteristics for your terminal, printer, and system clock.

Appendix A contains a summary of the keyboard monitor commands, their abbreviations, and their system program equivalents.

Documentation Conventions

A description of the symbolic conventions used throughout this manual follows. Familiarize yourself with these conventions before you continue reading.

1. In examples that show user input and computer output, user input is in red.

2. This manual uses the symbol (RET) to represent a carriage return (the **RETURN** key), (LF) to represent a line feed, (SP) for a space, and (TAB) to represent a tab. Unless the manual indicates otherwise, terminate all commands or command strings with a carriage return.

3. Terminal and console terminal are general terms used throughout all RT-11 documentation to represent any terminal device, including DECwriters and video terminals.

4. To produce certain characters in system commands, you must type a letter key while pressing the control (CTRL) key. For example, while holding down the CTRL key, type C to produce the CTRL/C character. Key combinations of this type are documented as (CTRL/C), (CTRL/Z), and so on.

5. In discussions of command syntax, uppercase letters represent the command name, which you must type. Lowercase letters represent a variable, for which you must supply a value.

   Square brackets ([ ]) enclose optional items; you may include the item in brackets or you may omit it, as you choose.

   The ellipsis symbol (...) represents repetition. You can repeat the item that precedes the ellipsis.

   This is a typical illustration of command syntax:

   ```
   .DELETE[/option...] filespec[/option...]
   ```

   This example shows that you must type the word DELETE, as shown, and that you can follow it with one or more options of your choice, but none are required. You must then leave a space, and supply a file specification. The file specification can also be followed by one or more options, but none are required. Here is a typical command string:

   ```
   .DELETE/QUERY/CONFIRM DLO:MYFILE.TMP
   ```

Part I
RT–11 Overview

Part I of this manual provides a description of the hardware and software components that make up the RT–11 operating system, and a summary of the program development cycle.

Chapter 1 lists all the hardware devices, monitors, utility programs, and language processors available in the RT–11 computer system. This chapter also lists the keyboard commands available in RT–11.

Chapter 2 gives a general description of the steps involved in the program development cycle. This chapter also summarizes the use of the RT–11 librarian and high-level languages.

Chapter 1
System Components

RT–11 is DIGITAL's smallest real-time and program development operating system for the PDP–11 family of minicomputers. This single-user operating system runs on hardware configurations ranging from the microprocessor-based PDP-11/03 through the larger PDP–11/44 with cache memory. RT–11 is designed to be small, efficient, reliable, and easy to use.

The RT–11 computer system consists of hardware, software, and documentation. This chapter describes briefly the components available for you to use with RT–11.

1.1 Hardware

The hardware components of an RT–11 system are drawn from the following categories:

- PDP-11, LSI–11, Professional 300 series, and SBC–11 family computers (except the 11/70 or VAX computers)
- Printing and video terminals
- Core and solid-state memory
- Line frequency and programmable clocks
- Random-access mass storage devices
- Other peripheral devices

The smallest possible hardware configuration for an RT–11 system must include a PDP–11, LSI–11, Professional 300 series, or SBC–11 computer, one terminal, 16K words of memory, a random-access mass storage device for the system device, and a system backup device. Larger systems can have a clock, more memory, more terminals, and more peripheral devices.

Table 1–1 lists specific hardware devices that can make up an RT–11 computer system.

Table 1–1: RT–11 Hardware Components

Device Type	Controller	Device Name
Card reader	CR11	CR11
Clock	—	KW11–L, KW11–P
DECtape II data cartridge	DL11, DLV11	TU58
Disk	RK11, RKV11 RK611 RL11, RLV11, RLV12 RC11 RQDX1 UDA–50	RK05, RK05F RK06, RK07 RL01, RL02 RC25 RD50/RD51 RA80
Diskette	RX11, RXV11 RX211, RXV21 RQDX1	RX01 RX02 RX50
Display processor	VT11 VS60 VS11	– – –
Line printer	LS11 LV11 LP11, LPV11	– LV11 All LP11–controlled printers (LP05, LP25, LP26)
Magtape	TM11, TMA11 RH11 TSV	TU10, TE16 TU16, TU45, TV77 TS11, TS05, TS90, TK25
Asynchronous terminal interface	DL11, DLV11 DZ11, DZV11	LA120, LA34, LA12, LA180 LQP02 VT100, VT101, VT102, VT105, VT125

1.2 Software

The software components of the RT–11 computer system can be divided into four general groups:

- Monitors
- Device handlers
- Utility programs
- Language processors

These are described in the following sections.

1.2.1 Monitors

An RT-11 monitor is a collection of routines that control the operation of programs, schedule operations, allocate resources, and perform input and output. A monitor comprises three major components: RMON (resident monitor); USR (User Service Routine); and KMON (keyboard monitor). The resident monitor (RMON) is the part of the monitor that is always present in memory. It is the executive controller for the entire system. The user service routine (USR) performs operations related to input and output, such as opening and closing files. The keyboard monitor (KMON) is the interface between you and the other parts of the system. It contains routines to process the keyboard monitor commands, which are your means of performing common system operations such as loading and running programs, assigning alternate device names, and copying and deleting files.

RT-11 provides three different operating environments that represent compromises among size, speed, and capability. Three types of monitors, all containing the main parts described above, supervise the different environments. These three monitors are the single-job (SJ) monitor, the foreground/background (FB) monitor, and the extended memory (XM) monitor. The three environments are upward compatible:

- The single-job (SJ) monitor supports the basic environment.
- The foreground/background (FB) monitor includes all the support of the single-job monitor and adds the ability to run more than one job, as well as some extra features.
- The extended memory (XM) monitor is an extension of the foreground/background monitor that includes all the foreground/background features, plus extended memory capabilities.

1.2.1.1 Single-Job (SJ) Monitor – The single-job monitor, called the SJ monitor, can run one job at a time. It is the smallest of the three monitors. While the SJ monitor does not offer some of the optional features that the other monitors have, you can use all the system utility programs, most of the keyboard monitor commands, and many of the programmed requests.

Only 16K words of memory are required for a single-job system, though, and since the SJ monitor uses approximately 2K words itself, this leaves approximately 14K words for system utility programs or for your application program. The SJ monitor is ideal for real-time applications that require a high data transfer rate because it services interrupts quickly. In the single-job environment, programs can access up to 28K words of memory (up to 30K words on some LSI–11s).

A version of the SJ monitor, the base-line (BL) monitor, also runs in a minimum configuration of 16K words of memory, but it does not support optional monitor and device functions. The BL monitor is best suited for very small hardware configurations, or for larger configurations where the application requires minimal executive support.

1.2.1.2 Foreground/Background (FB) Monitor — The foreground/background monitor, called the FB monitor, can accommodate two jobs that appear to run concurrently: a foreground job and a background job. All programs that run in the single-job environment, including system utility programs and language processors, can run as background jobs in the foreground/background environment. The foreground job is the time-critical, real-time job, and the FB monitor gives it priority over the background job. The FB monitor can also run system jobs. System job support is a system generation option which allows you to run up to eight jobs, including the foreground and background jobs. A foreground/background system requires 24K words of memory and a system clock.

Quite often, the central processor of a computer system spends much of its time waiting for some external event to occur. Usually, this event is a real-time interrupt or the completion of an I/O transfer. The FB monitor lets you take advantage of the unused processor capacity to accomplish lower priority work in the background.

Whenever the foreground job reaches a state in which no useful processing can be done until some external event occurs, the monitor executes the background job. The background job runs until the foreground job is again ready to execute. The processor then interrupts the background job and resumes the foreground job.

In effect, the FB monitor allows a time-critical job to run in the foreground while less critical work takes place in the background. All the system utility programs and language processors can run as background jobs in a foreground/background system, although more than 16K words of memory may be required. Thus, you can use FORTRAN or KED, for example, in the background, while the foreground job is collecting, storing, and analyzing data.

Compared to the SJ monitor, the FB monitor is somewhat larger and has slightly slower response time. However, it provides support for the foreground/background environment. In this environment, programs can access 28K words of memory (up to 30K words on some LSI–11s). Special keyboard monitor commands link, run, suspend, and resume foreground jobs. In addition, programmed requests permit a foreground job and a background job to transmit data to one another. Special system jobs (described in Part II of the *RT–11 System Utilities Manual*) run in the foreground/background environment.

1.2.1.3 Extended Memory (XM) Monitor — The extended memory monitor, called the XM monitor, includes all the features of the FB monitor. Throughout this manual, references to the foreground/background environment also apply to the extended memory environment, unless otherwise stated.

The XM monitor allows you to use memory configurations larger than the 28K words supported by the SJ and FB monitors. On 18-bit Q-bus and

UNIBUS processors, the XM monitor supports up to 124K words. On 22-bit Q-bus processors, the XM monitor supports up to 2048K words (2 megawords). This permits foreground and background jobs to extend their logical program space beyond the 32K-word limit imposed by the 16-bit PDP–11 address word to a total of 128K words per job. The XM monitor requires a system with the Extended Instruction Set (EIS), a KT11 memory management unit, and more than 32K words of memory.

Extended memory services, or the ability to use memory mapping, are available at a variety of levels. For DIBOL users, for example, the mapping to extended memory is completely transparent. FORTRAN programmers can use virtual arrays to store large amounts of data in extended memory. A LINK option permits RT–11 programmers to store overlays in extended memory instead of on disk, thus increasing an overlaid program's execution speed markedly. A virtual .SETTOP programmed request permits a MACRO–11 program to dynamically allocate buffers in extended memory without concern for memory mapping. Finally, on the most basic level, RT–11 provides other programmed requests that MACRO–11 programs can use to control their own mapping to extended memory. (Keep in mind that designing an application program to use extended memory this way requires considerable thought and careful planning. The *RT–11 Software Support Manual* and Chapter 11 of the *RT–11 System Utilities Manual* provide more detailed information on using extended memory.)

In the extended memory environment, jobs are described as being either privileged or virtual jobs. Foreground or background jobs that execute in the foreground/background or single-job environment can also execute in the extended memory environment as privileged jobs. That is, they use a one-to-one default mapping from logical virtual to physical memory. Except for jobs that include interrupt service routines, these privileged jobs need no major changes to execute properly in the extended memory environment.

1.2.2 Device Handlers

Device handlers are routines that provide the interface to the various hardware devices in the computer system. The handlers drive, or service, peripheral devices and control the physical activities on the devices. In RT–11, the terms device handler and device driver are used interchangeably.

A handler exists for every device the system supports (except for the VT11). When you reference a device by its physical name, such as DL: for the RL02 disk, you are actually referring to the name of the device handler for that peripheral.

Chapter 3 contains a list of all the devices that RT–11 supports, along with their physical names. If you need to use a peripheral device that is not supported by RT–11, you usually must write the handler for it yourself. The procedure for doing this is documented in the *RT–11 Software Support Manual*.

1.2.3 System Utility Programs

RT-11 provides a number of utility programs to help you develop programs and perform system housekeeping. The following sections describe these utilities briefly and refer you to more detailed descriptions in the documentation set.

1.2.3.1 Editing — You use text editors to create and modify source programs and to maintain files of any ASCII data, such as memos or documentation for your own application programs. DIGITAL distributes two text file editors with RT–11, so you can choose the one that best suits your needs and experience: EDIT and KED. DIGITAL also distributes, but does not support the K52 text editor for use with VT52 terminals.

The RT-11 text editor (EDIT, described in Chapter 6 of this manual) is a character-oriented editor suitable for hard copy terminals. Its text manipulation commands permit you to make text insertions or changes quickly and easily. EDIT also has a special mode for VT11 or VS60 graphics display terminals.

The keypad editors (KED and K52 described in the *PDP–11 Keypad Editor User's Guide*) are for the video terminals that have the special function keypad. The keypad keys control the editing functions. They permit you to position a visible cursor anywhere in your text file and make insertions or changes easily. KED runs on the VT100 family of terminals, and K52 runs on VT52 terminals. A virtual keypad editor, KEX, is also available for editing when running under the XM monitor. You use KEX exactly as you would KED; see the *PDP–11 Keypad Editor User's Guide* for information on how to use KED.

A subset of the keypad editor, the single-line editor, allows you to edit command lines and input as you type them by using the PF1-PF4 and cursor control keys. The single-line editor is described in Section 4.3 of this manual.

1.2.3.2 General Purpose — RT–11 provides several utility programs that help you perform maintenance on your system and aid in program development. You can obtain the services these programs provide by using the keyboard commands described in this manual, or you can call these programs directly as described in the *RT–11 System Utilities Manual*. Each of these programs is described in greater detail in separate chapters of the *RT–11 System Utilities Manual*.

The binary file comparison program (BINCOM) compares two binary files and lists the differences between them. It can provide a quick way of telling whether two data files, or output from two versions of a program, are identical. BINCOM can also produce a file that can be run as an indirect command file for the save image patch program (SIPP) to patch one file in the binary comparison so it matches the other.

The backup utility program (BUP) provides an easy way to back up and restore large files and entire volumes onto several smaller volumes.

The directory listing program (DIR) performs a wide range of directory listing operations and can list details about certain files, such as file names, file types, and block sizes.

The dump utility program (DUMP) prints all or any part of a file or volume in octal words, octal bytes, ASCII characters, or Radix-50 characters.

The general device utility program (DUP) performs general device tasks such as initializing devices, scanning for bad blocks, duplicating device contents, and reorganizing files on the device. It operates only on RT–11 file-structured devices.

The file exchange utility program (FILEX) transfers files between RT–11 and the following systems, on DECtape and disks: DECsystem-10, PDP-11 RSTS/E, and DOS BATCH. FILEX also transfers files between RT-11 and other systems on diskettes that use IBM interchange format.

The volume formatting utility program (FORMAT) provides a way to format RK05, RK06, and RK07 disks, and diskettes. It also provides disk verification by writing patterns and reading them on each block of your volume.

The logical disk subsetting handler (LD) allows you to assign files as logical disks. Thereafter you can treat them as if they were separate RT-11 directory-structured volumes.

The librarian utility (LIBR) lets you create and maintain libraries of functions and routines. These routines can be stored on a random-access device in library files where the linker can reference them and add them to a program's memory image file. You can create object libraries and macro libraries. The latter are used by the MACRO assembler.

The linker utility (LINK) converts a collection of object modules from compiled or assembled programs and subroutines into a memory image file that RT–11 can load and execute. The linker also allows you to:

- Search library files for subroutines that you specify
- Produce a load map that lists the assigned absolute addresses
- Set up a disk or memory resident overlay structure for large programs
- Create a symbol table file that lists all the global symbols used in the program
- Produce files suitable for execution as foreground jobs

The peripheral interchange program (PIP) is the RT–11 file maintenance program. It transfers files between devices that are part of the RT–11 system, and it deletes and renames files as well.

The resource program (RESORC) lists information about your system configuration and system generation special features.

The SETUP program lets you set hardware characteristics for your terminal, printer, or system clock. Although developed specifically for the Professional 300 series processors, SETUP can be used to set some hardware characteristics for other devices supported by RT–11. A complete description of SETUP is located in Chapter 7.

The source file comparison program (SRCCOM) performs a character-by-character comparison of two ASCII text files. You can request that the differences be listed in an output file or directly on the line printer or terminal to make sure that edits to a file have been performed correctly. SRCCOM can also produce a file that is suitable as input to SLP, the source file patching utility.

1.2.3.3 System Jobs — RT–11 provides five utilities that you can run as foreground jobs or, if you have enabled system job support through the system generation process, as system jobs: the Error Logger, the Queue Package, the Transparent Spooler Package (SPOOL), the Virtual Terminal Communication Package (VTCOM), and KEX. The Error Logger, the Queue Package, SPOOL, and VTCOM run under both the FB and XM monitors; KEX runs only under the XM monitor. (The Error Logger also runs under the SJ monitor.) System jobs are described in more detail in the *RT–11 System Utilities Manual*.

The Error Logger keeps a statistical record of all I/O transfers for each device it supports. The Error Logger also records memory parity and cache errors as they occur. With the Error Logger enabled on your system volume, you can collect data on each I/O and memory error that occurs. The Error Logger consists of three programs, a data file, and a handler. This utility is a special feature; that is, you must enable it through the system generation process.

The Queue Package transfers files to any valid non-file-structured RT–11 device. The Queue Package is particularly useful for queuing files for subsequent printing, although output is not restricted to the line printer. Unlike the Error Logger, the Queue Package is not a special feature available only through system generation.

The transparent spooler (SPOOL) transfers files to the line printer. Unlike the QUEUE package, SPOOL does not require complete files, but rather accepts output as it becomes available. A software customization can be applied to direct SPOOL's output to any valid non-file-structured RT–11 device. SPOOL is not a special feature and does not require system generation to be available.

The virtual terminal communication package (VTCOM) lets your computer act as a terminal when connected to a host computer. When running VTCOM, you can access facilities on an RT–11, RSX–11, RTEM–11, or VAX/VMS host system and perform ASCII file transfers between the host computer and your stand-alone system. Binary files can be transferred if the host system is running RTEM–11 with the TRANSF program installed. VTCOM is not a special feature and does not require system generation to be available.

1.2.3.4 Debugging and Patching — These utility programs help you to find, diagnose, and correct programming errors. Debugging and patching programs are described in more detail in the *RT–11 System Utilities Manual*.

The on-line debugging technique (ODT) is an object module that you link with your program. It helps you debug assembled and linked programs. ODT can:

- Print and change the contents of specified locations
- Execute all or part of the object program
- Search the object program for specific bit patterns

The object module patch program (PAT) performs minor modifications to files in object format (output files produced by the FORTRAN compiler or the MACRO assembler). It can merge several object files into one.

The save image patch program (SIPP) can update programs that were linked with the RT–11 V4 or V5 linker. It can also update non-overlaid programs from versions V3 and V3B of RT–11.

The source language patch program (SLP) provides an easy way to make changes to source files. SLP can use an indirect command file created by a SRCCOM option to make two source files match.

1.2.3.5 BATCH — The batch program (BATCH, described in Appendix A of the *RT–11 System Utilities Manual*) is a complete job-control language that allows RT–11 to operate unattended.

1.2.4 Language Processors

RT–11 supports a number of language processors to help you develop programs. The *Introduction to RT–11* contains detailed information on the differences between assembly language and high-level languages. It also offers guidelines for choosing a programming language and provides demonstrations of MACRO, BASIC, and FORTRAN programs.

The MACRO–11 assembler (see Chapter 12 of the *RT–11 System Utilities Manual*) is part of the RT–11 system. Because MACRO–11 is an assembly language, it gives you control over the system at the most elementary However, it may be more difficult to learn and use than any of the high-level languages.

The high-level languages RT–11 supports are:

- DIBOL
- BASIC
- FORTRAN IV

1.3 RT–11 Software Documentation

The software documentation for the RT–11 system consists of the manuals that document the RT–11 system itself, plus the documentation for any optional languages or application packages you may have.

The *Guide to RT–11 Documentation* summarizes the manuals in the RT–11 documentation set. Reading this guide gives you a general picture of the topics covered in the manuals.

To find more specific information, refer to the *RT–11 Master Index*. This is a compilation of the indexes of the other RT–11 manuals. It pinpoints references by manual name and page number. It also indicates which reference is the primary source of information on the specific topic.

1.4 System Services

The RT–11 system provides many services that allow you, for example, to copy and delete files, to examine locations in memory, to run programs, and to open and close files. Some of these services are available to you at the console terminal; others are available to application programs.

1.4.1 Keyboard Monitor Commands

The keyboard monitor commands are a set of English–language commands that permit you to perform common system operations. When you type a keyboard monitor command at the console terminal, RT–11 responds by performing the operation you specify. The monitor then prompts you for another command and waits for you to respond. Chapter 4 describes the syntax and function of each of the keyboard monitor commands.

The set of keyboard monitor commands consists of two types of commands: simple and complex. Simple, or direct, commands are executed directly by the keyboard monitor, and no other software components are required. The complete set of simple commands is as follows:

ABORT	DEASSIGN	GT	REMOVE	SET	UNLOAD
ASSIGN	Deposit	INSTALL	RESET	SRUN	
Base	Examine	LOAD	RESUME	START	
CLOSE	FRUN	R	RUN	SUSPEND	
DATE	GET	REENTER	SAVE	TIME	

Complex, or expanded, commands are not executed directly by the keyboard monitor. Instead, a utility program or language processor is called by the keyboard monitor to perform the operation. The keyboard monitor expands the command line piece by piece and translates the command into an R command followed by a program name and one or more lines of file specifications and options for that program. When the operation completes, control returns to the keyboard monitor and it prompts you for another command. The set of complex commands is as follows:

BACKUP	DIBOL	EXECUTE	LINK	SHOW
BOOT	DIFFERENCES	FORMAT	MACRO	SQUEEZE
COMPILE	DIRECTORY	FORTRAN	MOUNT	TYPE
COPY	DISMOUNT	HELP	PRINT	UNPROTECT
CREATE	DUMP	INITIALIZE	PROTECT	
DELETE	EDIT	LIBRARY	RENAME	

1.4.2 System Programs

Another way to obtain services from RT-11 is to invoke system utility programs or language processors yourself, instead of invoking them indirectly through the keyboard monitor commands. By using this method you can obtain all the services provided by the complex keyboard monitor commands. (The only way to obtain the services provided by the simple keyboard monitor commands is to issue those commands.) A limited number of utility program operations are not implemented through the keyboard monitor. In addition, you must run some of the utility programs directly in order to use them at all. Programs in this group include the patching and debugging utilities.

To invoke a system utility program or a language processor, you run the appropriate program and specify a combination of file specifications and single alphabetic character options. The *RT–11 System Utilities Manual* describes how to use the system utility programs and MACRO language processor directly. Chapter 1 describes the syntax you use to interact with the utility programs and language processors. Chapters 2 through 23 contain detailed information on each program.

You can also invoke the system utility programs and language processors by using CCL, the concise command language. Section 4.7 of this manual describes how to use CCL.

1.4.3 The Relationship Between Complex Commands and System Programs

It is possible to obtain the services provided by the complex keyboard monitor commands by directly running the appropriate system programs. Appendix A of this manual provides a complete list of the keyboard monitor commands and the system programs they invoke.

The following examples demonstrate two ways of copying a listing of a program from the default disk, where it is stored as MYFILE.T, to the line printer. The keyboard monitor command to do this is as follows:

```
.PRINT MYFILE.MAC
```

The commands to invoke a utility program, specify the same operation, and return control to the monitor are:

```
.R PIP
*LP:=DK:MYFILE.LST
*^C
```

(CTRL/C echoes on your terminal as ^C.)

So, although there are two ways of obtaining the same services, bear in mind that the syntax for using the utility programs and language processors is quite different from the keyboard monitor command syntax. Since the keyboard commands are easy to remember and easy to use, it makes sense to use them whenever possible.

1.4.4 The System Macro Library and Programmed Requests

The system macro library, called SYSMAC.SML, contains macro definitions that you can use in MACRO assembly language programs and in device handlers. You reference the definitions in your assembly language program, and they expand into lines of source code. These macros can save you considerable programming effort. See the *RT–11 Programmer's Reference Manual*.

1.4.5 SYSLIB FORTRAN-Callable Subprograms

All of the system subroutine library (SYSLIB) routines are written in MACRO. They give the FORTRAN programmer many of the services that the MACRO programmer can obtain from the system macro library (SYSMAC.SML). These subprograms can be called from a program written in any programming language, as long as the program conforms to the FORTRAN calling conventions described in the *RT–11 Programmer's Reference Manual*.

Chapter 2
Program Development

RT-11 provides several program development aids, including editors, an assembler, a linker, a debugger, and a librarian. High-level languages, such as FORTRAN or BASIC, are optionally available.

This chapter describes briefly the program development cycle, which is illustrated in Figure 2–1. The *Introduction to RT–11* contains a much more thorough treatment of program development including demonstrations of MACRO, BASIC, and FORTRAN programs.

2.1 Using an Editor (EDIT, KED, KEX, or K52)

You use an editor to create and modify textual material. Text may be the statements in a source program, or any other ASCII data such as reports or memos. In this respect, using an editor is analogous to using a typewriter; you sit at a keyboard and type text. However, the functions of an editor far exceed those of a typewriter. Once a text file has been created, you can modify, relocate, replace, merge, or delete text, all by means of editing commands. When you are satisfied with your text, you can save it on a storage device where it is available for later reference.

2.2 Using the Assembler (MACRO)

Program development does not stop with the creation of a source program. Since the computer cannot understand any language but machine language, you need an intermediary program to convert source code into instructions the computer can execute. This is the function of an assembler.

The assembler accepts alphanumeric representations of PDP-11 instructions, and produces as output the appropriate machine code, called object code. You can direct the assembler to generate a listing of both the source code and binary output, as well as cross-reference listings that are helpful during the program debugging process. In addition, the assembler is capable of detecting certain common coding errors and issuing appropriate warnings.

The assembler's output is called object output because it is composed of object, or binary, code. On PDP-11 systems, the object output is called a module; it contains your source program in the binary language that, when linked, is executable by a PDP-11 computer.

2.3 Using the Linker (LINK)

Source programs may be complete and functional by themselves; however, some programs are written in such a way that they must be used with other programs or modules to form a complete and logical flow of instructions. For this reason, the object code produced by the assembler must be relocatable. That is, assignment of memory locations must be deferred until the code is combined with all other necessary object modules. The linker performs this function.

The linker combines and relocates separately assembled object programs. The output produced by the linker is a load module, the final linked program that is ready for execution. You can, if you wish, request a load map that displays all addresses assigned by the linker.

2.4 Using the Debugger (ODT or VDT)

You can rarely create a program that does not contain at least one error, either in the logic of the program or in its coding. You may discover errors while you are editing the program, or the assembler may find errors during the assembly process and inform you by means of error codes. The linker may also catch certain errors and issue appropriate messages. Often, however, it is not until execution that you discover your program is not working properly. Programming errors may be extremely difficult to find, and for this reason, a debugging tool, ODT (described in Chapter 20 of the *RT-11 System Utilities Manual*), is available to help you find the cause of errors.

ODT allows you to control the execution of your program interactively. With it, you can examine the contents of individual locations, search for specific bit patterns, set designated stopping points during execution, change the contents of locations, continue execution, and test the results — all without editing and reassembling the program.

Note that it is advisable to test new programs by having them process data for which results are already known. If the results do not match, you know you have errors.

Use VDT, the Virtual Debugging Technique, to debug virtual and privileged jobs in an XM system. You can also use VDT to debug jobs in FB, SJ, and multiterminal systems. See the *RT-11 Software Support Manual* for more information on using VDT.

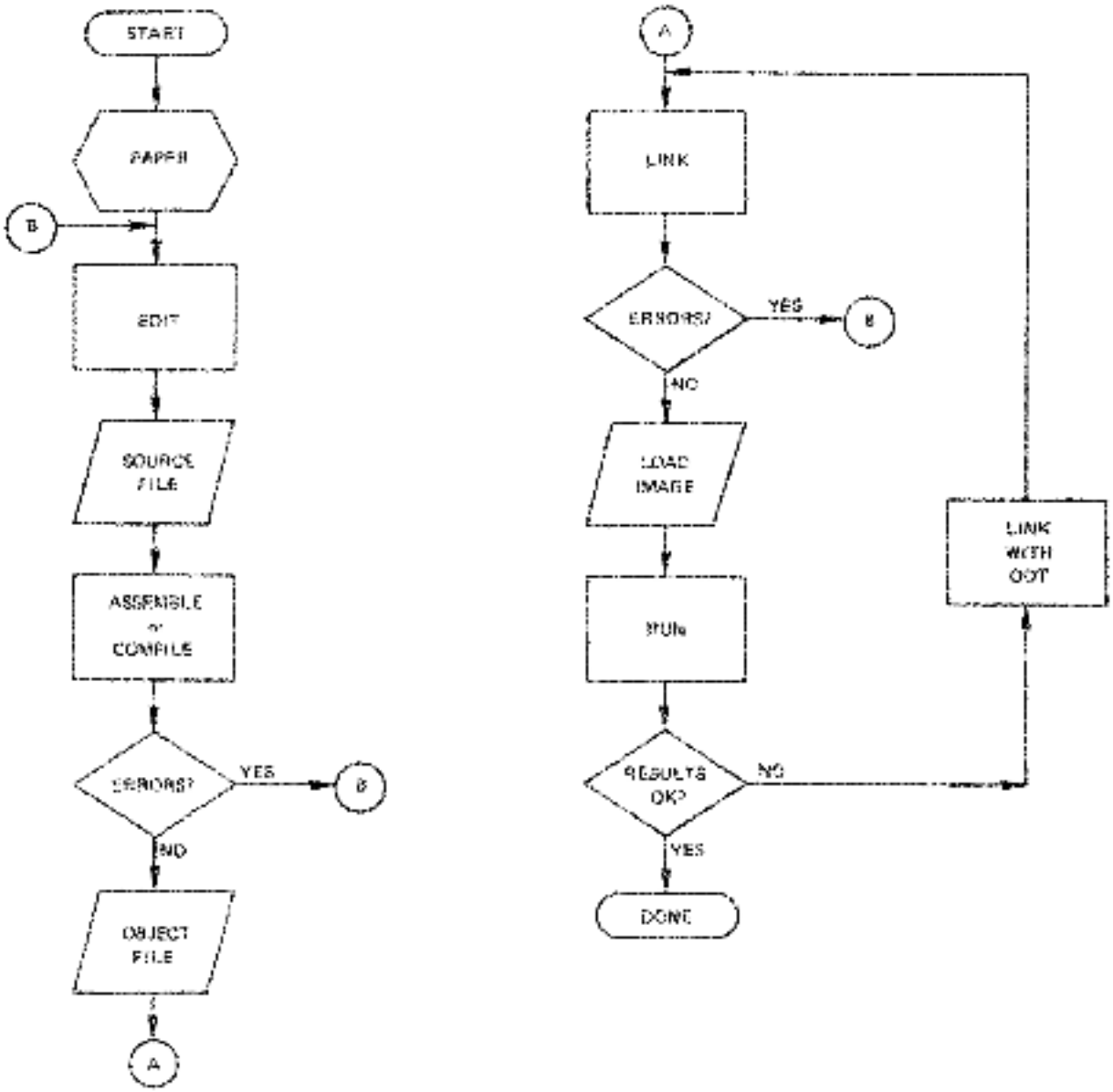
2.5 Using the Librarian (LIBR)

When programs are written and debugged, they are useful to other programmers. Often, routines that are common to many programs, such as input and output routines, or sections of code that are used over and over again, are more useful if they are placed in a library where they can be retrieved by any interested user. A librarian provides such a service by allowing creation of a library file. Once created, the library can be expanded or updated, or a directory of its contents can be listed.

2.6 Using a High-Level Language (FORTRAN, BASIC, or DIBOL)

High-level languages simplify your work by providing an alternative means, other than assembly language, of writing a source program. Generally, high-level languages are easy to learn. A single command causes the computer to perform many machine-language instructions. You do not need to know about the mechanics of the computer to use a high-level language. In addition, some high-level languages, such as BASIC, offer a special immediate mode that allows you to solve equations and formulas as though you were using a calculator. You can concentrate on solving the problem rather than on using the system.

Figure 2–1: Program Development Cycle



Part II
System Communication

The monitor is the center of RT–11 system communications: it provides access to system and user programs, performs input and output functions, and controls foreground and background jobs.

You communicate with the monitor through keyboard commands and programmed requests. You can use the keyboard commands (described in Chapter 4) to load and run programs, start or restart programs at specific addresses, modify the contents of memory, and assign and deassign alternate device names, to name only a few of the functions.

Programmed requests (described in detail in the *RT–11 Programmer's Reference Manual*) are instructions that request the monitor to perform services. These instructions allow assembly language programs to use the monitor features. A running program communicates with the monitor through programmed requests. FORTRAN programs have access to programmed requests through the system subroutine library (SYSLIB). Programmed requests can, for example, manipulate files, perform input and output, and suspend and resume program operations.

Of the three chapters in this part, Chapter 3 describes system conventions and contains information that helps you get started with RT–11. Chapter 4 introduces the keyboard monitor commands, which are your means of controlling the RT–11 system, and Chapter 5 describes how to use IND, the indirect control file processor.

Chapter 3
System Conventions

This chapter contains information that will help you start using the RT-11 system. It describes:

- Start-up procedure
- Data formats
- Physical device names
- File names and file types
- Device structures
- Special function keys
- Foreground/background terminal I/O
- Type-ahead feature

Before you operate the RT-11 system, you should be familiar with the special character commands, file naming procedures, and other conventions that are standard for the system. These conventions are described in this chapter.

3.1 Start-Up Procedure

For information on building the system and loading the monitor, refer to the *Introduction to RT-11*, to your *RT-11 Automatic Installation Booklet*, or to the *RT-11 Installation Guide*.

When the system is built and you load the monitor into memory, the monitor prints one of the following identification messages on the terminal:

```
RT-11SJ (S)  Vxx.nn
RT-11FB (S)  Vxx.nn
RT-11XM (S)  Vxx.nn
```

The message indicates which monitor (SJ, FB, or XM) is loaded; you specify that monitor when you install the system. The (S) indicates that the monitor was created through the system generation process. (The S designation does not appear with distributed monitors.)

Vxx represents the version and release number of the monitor -- for example, V05 for Version 5. nn represents the submission number and the patch level -- for example, 01B for number 1.

As soon as a monitor takes control of the system, it attempts to execute keyboard monitor commands from a start-up indirect command file called STARTS.COM for the SJ monitor, STARTF.COM for the FB monitor, or STARTX.COM for the XM monitor. You can place commands in this start-up file that will perform routine tasks, such as assigning logical device names to physical devices or setting the current date. If the monitor does not find the appropriate file, it issues a warning message. (Note that if you do not want the start-up indirect command file feature, you can disable it during system generation or you can apply a software customization.)

After executing the start-up indirect command file, the system prints its prompt (.) indicating that it is ready to accept commands. Make sure the system device is write-enabled.

3.2 Data Formats

The RT-11 system stores data in two formats: ASCII and binary. The binary data can be organized in many formats, including object, memory image, relocatable image, and load image.

Files in ASCII format conform to the American Standard Code for Information Interchange, in which each character is represented by a 7-bit code. Files in ASCII format include program source files created by the editor and BASIC, listing and map files created by various system programs, and data files consisting of alphanumeric characters.

Files in binary object format consist of data and PDP-11 machine language code. Object files are the files the assembler or language compiler produces; they are used as input to the linker.

The linker can produce runnable files in one of three formats: memory image format (.SAV), relocatable image format (.REL), or load image format (.LDA).

A memory image (.SAV) file is a picture of what memory looks like after you load a program. The file itself requires the same number of disk blocks as the corresponding number of 256-word memory blocks. A memory image file does not require relocation and can run in a single-job environment, as a background program under the FB or XM monitor, or as a foreground virtual job under the XM monitor.

A relocatable image (.REL) file is linked as though its bottom address were 1000, but relocation information is included with its memory image. When you call the program 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.

You can produce a load image (.LDA) file for compatibility with the PDP-11 paper tape system. The absolute binary loader loads this file. You can load and execute load image files in stand-alone environments without relocating them.

3.3 Device Names

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

Table 3–1: Permanent Device Names

Permanent Name	I/O Device
DBn:	TU58 DECtape II (n is an integer in the range 0–3)
DK:	Default logical storage device for all files (DK: is initially the same as SY:)
DKn:	Specified unit of same device type as DK
DLn:	RL01, RL02 disk (n is an integer in the range 0–3)
DMn:	RK05, RK07 disk (n is an integer in the range 0–7)
DUn:	MSCP disk or diskette: RC25 fixed/removable, RD51 and RD52 fixed, and RA80 fixed Winchester disk; RX50 diskette (n is an integer in the range 0–7)
DW:	RD50/RD51 disk (Professional 300 series)
DXn:	RX01 diskette (n is an integer in the range 0–3)
DYn:	RX02 diskette (n is an integer in the range 0–3)
DZ:	RX50 diskette (Professional 300 series)
ER:	SJ monitor Error Logger pseudodevice
LD:	Logical disk subsetting handler pseudodevice
LP:	Line printer
LS:	Serial line printer (hard-copy output device connected to a DL11 interface)
MMn:	TJU16/TU45 (industry-compatible) magtape (n is an integer in the range 0–7)
MQ:	Message queue pseudodevice for interjob communication under FB and XM monitors.
MSn:	TK25/TS11/TS05/TU80 magtape (n is an integer in the range 0–7)
MTn:	TM11/TMA11/TS03/TE16 (industry-compatible) magtape (n is an integer in the range 0–7)
NL:	Null pseudodevice
PI:	Professional 300 series interface handler
RKn:	RK05 disk cartridge drive (n is an integer in the range 0–7)
SP:	Transparent spooler handler
SY:	Default logical system device; device and unit from which system is bootstrapped
SYn:	Specified unit of same device type as SY:
TT:	Console terminal keyboard and display (hard-copy or video screen)
VM:	Extended memory handler
XC:	Communication port handler (Professional 300 series)
XL:	Communication port handler

In addition to using the permanent names shown in Table 3–1, you can assign logical names to devices. A logical name takes precedence over a physical name and thus provides device independence. With this feature, you do not have to rewrite a program that is coded to use a specific device if the device becomes unavailable. You associate logical names with physical devices by using the ASSIGN command. This command is described in Section 4.6.

3.4 File Names and File Types

You can reference files symbolically 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. It is good practice to conform to the standard file types for RT–11. If you do not specify a file type for an input or output file, most system programs use or assign an appropriate default file type. Table 3–2 lists the standard file types used in RT–11.

Table 3–2: Standard File Types

File Type	Meaning
.ANS	SYSGEN answer file
.BAC	Compiled BASIC program
.BAD	Files with bad (unreadable) blocks; you can assign this file type whenever bad areas occur on a device. The .BAD file type makes the file permanent in that area, preventing other files from using it and consequently becoming unreadable
.BAK	Editor backup file
.BAS	BASIC source file (BASIC input)
.BAT	BATCH command file
.BLD	Command file to execute SYSGEN monitor (.MON) and device handler (.DEV) build files
.BUP	Backup utility program output file
.CND	SYSGEN conditional file
.COM	KMON indirect command file, IND indirect control file, or SIPP command file
.CTL	BATCH control file generated by BATCH compiler
.CTT	BATCH internal temporary file
.DAT	BASIC, FORTRAN, or IND data file

(Continued on next page)

Table 3–2: Standard File Types (Cont.)

File Type	Meaning
.DBL	DIBOL source file
.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)
.FOR	FORTRAN IV source file (FORTRAN input)
.LDA	Absolute binary (load image) file (optional linker output)
.LOG	BATCH log file
.LST	Listing file (MACRO, FORTRAN, LIBR, or DIBOL output)
.MAC	MACRO source file (LIBR, MACRO or SRCCOM input)
.MAP	Map file (linker output)
.MLB	MACRO library output file
.MON	SYSGEN monitor build file
.OBJ	Relocatable binary file (MACRO or FORTRAN output, linker input, LIBR input and output)
.REL	Foreground job relocatable image (linker output, default for monitor FRUN and SRUN commands)
.SAV	Memory image; default for R, RUN, SAVE, and GET keyboard monitor commands; default for linker output
.SCP	SCP command file
.SML	System MACRO library
.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	ERROUT temporary file
.TXT	Text file
.WRK	Temporary work file

3.5 Device Structures

RT–11 devices are categorized according to two characteristics: their method of processing information and their physical structure.

All RT–11 devices are either randomly accessed or sequentially accessed. Random-access devices allow the system to process blocks of data in a 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, DECtape, and DECtape II 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, line printer, and terminal.

File-structured devices are those devices that allow the system to store data under assigned file names. RT–11 devices that are file-structured include all disk, diskette, DECtape II, and magtape devices. Non-file-structured devices, however, do not store files; they contain a single logical collection of data. These devices, which include the line printer and terminal, are generally used for reading and listing information.

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 Software Support Manual* contains a more detailed explanation of a device directory.) RT–11 directory-structured devices include all disks and DECtapes. 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 the system must read the device sequentially to obtain all information about all files.

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

3.6 Special Function Keys

Special function keys and keyboard commands let you communicate with the RT–11 monitor to allocate system resources, manipulate memory images, start programs, and use foreground/background services.

The special functions of certain terminal keys you need for communication with the keyboard monitor are explained in Table 3–4. In an FB system, the keyboard monitor runs as a background job when no other background job is running.

Table 3–3: Device Structures

Device	Structure			
	File	Non-file	RT-directory	Non-RT-directory
Random Access				
Disk, diskette	x		x	
DECtape II	x		x	
Sequential Access				
MagTape	x			x
Line printer		x		
Terminal		x		

Enter CTRL commands by holding the CTRL key down while typing the appropriate letter.

Table 3–4: Special Function Keys

Keys	Function
CTRL/A	Valid only after you type the monitor GT ON command and use the display. CTRL/A, a command that does not echo on the terminal, pages output if you use it after CTRL/S. The system permits console output to resume until the screen is completely filled; text currently displayed scrolls upward off the screen. CTRL/A has no effect if the keyboard monitor command GT ON is not in effect.
CTRL/B	Causes the system to direct all keyboard input to the background job. The FB monitor echoes B> on the terminal. The system takes at least one line of output from the background job. The foreground or system job, however, has priority, so the system 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 an SJ monitor or when a SET TT NOPB command is in effect.
CTRL/C	Terminates program execution and returns control to the keyboard monitor. CTRL/C echoes ^C on the terminal. You must type two CTRL/Cs 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 FB monitor, 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/R, CTRL/F, or CTRL/X before typing CTRL/C.
CTRL/E	Causes all terminal output to appear on the graphics display screen and the console terminal simultaneously. CTRL/E is valid after you type the monitor GT ON command and use the display. The command does not echo on the terminal. A second CTRL/E disables console terminal output. CTRL/E has no effect if GT ON is not in effect.

(Continued on next page)

Table 3–4: Special Function Keys (Cont.)

Keys	Function
CTRL/F	Causes the system to direct all keyboard input to the foreground job and take all output from the foreground job. The FB monitor echoes F> on the terminal unless output is already coming from the foreground job. If no foreground job exists, the monitor prints 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 an SJ monitor, or when a SET TT NOFB command is in effect.
CTRL/O	Suppresses terminal output while continuing program execution. CTRL/O echoes as ^O on the terminal. RT-11 resumes terminal output when one of the following occurs: 1. You type a second CTRL/O. 2. You return control to the monitor by typing CTRL/C or by issuing the .EXIT request in your program. 3. The running program issues a .RCTRLO or .MTRCTO programmed request (see the *RT–11 Programmer's Reference Manual*). RT–11 system programs reset CTRL/O to the echoing state each time you enter a new command string. When you are using CTRL/O under the SJ monitor, the system may print an extraneous character after the monitor echoes the CTRL/O and a carriage return/line feed.
CTRL/Q	Resumes printing characters on the terminal from the point printing previously stopped because of a CTRL/S. CTRL/Q echoes but has no effect under a multiterminal SJ or FB monitor if a SET TT NOPAGE command is in effect.
CTRL/R	Redisplays the current line if you are using the CTRL/W single-line editor. This function is useful to verify a line you have edited, to verify that your screen is displaying information correctly, or if another job prints a message on your screen while you are typing input to the console.
CTRL/S	Temporarily suspends output to the terminal until you type a CTRL/Q. CTRL/S does not echo. Under a multiterminal SJ or FB monitor, CTRL/S is not intercepted by the monitor if SET NOPAGE is in effect.
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 SJ is running, CTRL/U deletes the current input line. When SJ is not running, CTRL/U echoes as ^U, followed by a carriage return/line feed at the terminal.
CTRL/W	Redisplays the current line if you are using the single-line editor. This function is useful to verify a line that you have edited, to verify that your screen is displaying information correctly, or if another job prints a message on your screen while you are typing input to the console.
CTRL/X	Causes the system to prompt you for a job name, then to direct all keyboard input to the system job you specify. When you type CTRL/X, the system prints *job?* at the terminal. 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

(Continued on next page)

Table 3–4: Special Function Keys (Cont.)

Keys	Function
	not exist the system prints a question mark (?); otherwise it prints 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). CTRL/X has no effect when used with a monitor that does not have system job support or when a SET TT NOSB command is in effect.
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.
DELETE or RUBOUT	Deletes the last character from the current line and echoes a backslash plus the character deleted. Each succeeding DELETE deletes and echoes another character. The system prints an enclosing backslash when you type a key other than DELETE. This erasure is performed from right to left up to the beginning of the current line. If you are using a video display terminal and you have issued the SET TT SCOPE command, DELETE erases characters with a backspace, space, backspace sequence. Your corrections appear on the screen; pressing the DELETE key does not enclose them with backslash characters.

3.7 Foreground/Background Terminal I/O

Console input and output under the FB monitor are independent functions; therefore, you can type input to one job while another job prints output. You may be in the process of typing input to one job when the system is ready to print output from another job on the terminal. In this case, the job that is ready to print interrupts you and prints the message on the terminal; the system does not redirect input control to this job, however, unless you type a CTRL/B, CTRL/F, or CTRL/X, whichever applies. If you type input to one job while another has output control, the system suppresses the echo of the input until the job accepting input gains output control; at this point, all accumulated input echoes.

If the two jobs are ready to print output at the same time, the job with the higher job number has priority. For example, in an FB system, the system prints output from the foreground job until it encounters a line feed. Each time the system prints a line feed, it checks to see if the foreground job (or, in a monitor with system job support, any higher priority job) has output; if so, the system gives control to the highest priority job that is ready to print.

When the foreground job terminates, control reverts automatically to the background job, or to the highest priority job if you are running system jobs.

3.8 Type-Ahead Feature

The monitor has a type-ahead feature that lets you enter terminal input while a program is executing. For example:

```
.DIRECTORY/PRINTER
.DATE
```

While the first command line is executing, you can type the second line. Although the system echoes the characters you type immediately after you type them, the system stores this terminal input in a buffer and uses it when the system completes the first operation.

If type-ahead input exceeds the input buffer capacity (usually 134 characters), the terminal bell rings and the system 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 specify multiple command lines to system programs.

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

If you type a single CTRL/C while the system is in this mode, the system puts CTRL/C into the buffer. The program currently executing exits when it makes a terminal input request. Typing a double CTRL/C returns control to the monitor immediately. If you terminate a job by typing two CTRL/Cs, the system discards any unprocessed type-ahead input.

Chapter 4
Keyboard Commands

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

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

4.1 Command Syntax

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

The general syntax for a command is:

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

or

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

where:

command	is the command name
/option	represents a command qualifier that specifies the exact action to be taken. Any option you supply immediately following the command applies to the entire command string
prompt1 prompt2	represents the keyboard monitor prompt for more information. The keyboard monitor prints an appropriate prompt only if you omit input and/or output file or device specifications in the initial command line. (Note that not all keyboard monitor commands print prompts, and some print more than two prompts.)

input-filespec	represents the file on which the action is to be taken
/option	represents a file qualifier that specifies more detailed information about that particular file or action to be taken. Some of the command and file options are mutually exclusive. You should avoid using combinations of options that give contradictory instructions to the system. For example:
output-filespec	represents the file that is to receive the results of the operation
/option	represents a file qualifier that specifies more detailed information about that particular file or action to be taken

```
.DELETE/QUERY/NOQUERY TEST.DAT
```

You can use a hyphen the end of a 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 keyboard command copies three input files to the output file DK:OUTFIL.TXT.

```
.COPY/CONCATENATE MYFILE.TXT-
MYFILO.TXT DK:OUTFIL.TXT
```

In the alphabetical listing of keyboard monitor commands in Section 4.5, each command begins with a graphic presentation of the syntax involved (see Figure 4–1 for an illustration of a typical command). These presentations provide a ready-reference list of the options that the commands accept, as well as information that makes the commands easier to use. The following list describes the conventions used

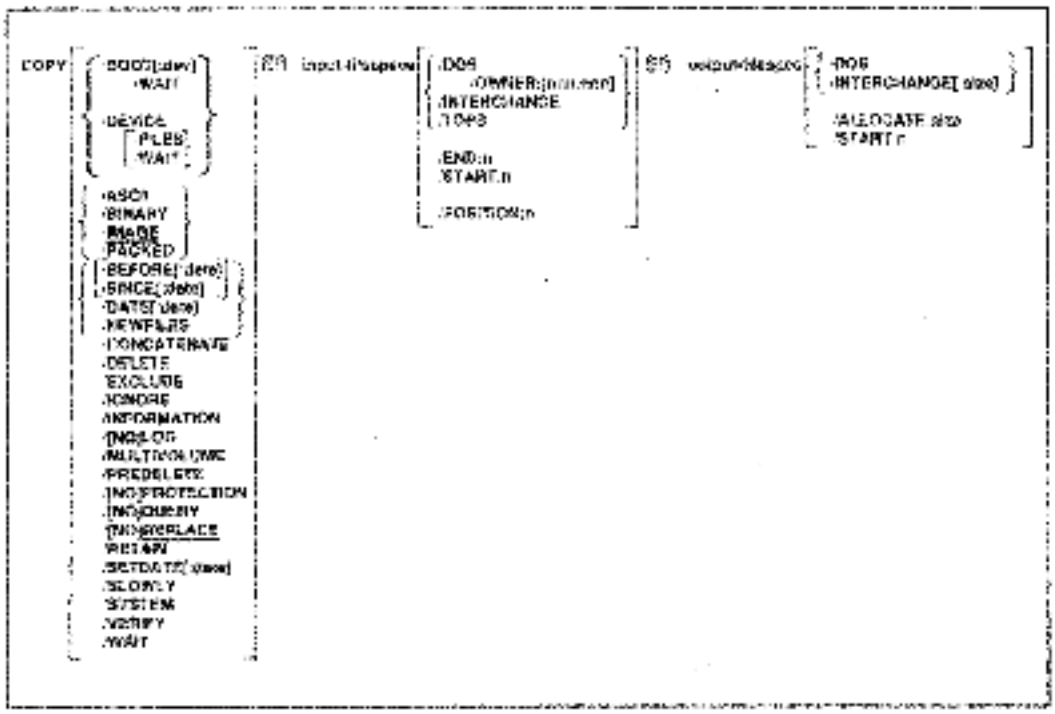
1. Capital letters represent command names or options, which you must type as shown. (Abbreviations are discussed later in this section.)
2. Lowercase letters represent arguments or variables, for which you must supply values. For options that accept numeric arguments, the system interprets the values as decimal, unless otherwise stated. Some values, usually memory addresses, are interpreted as octal; these cases are noted in the accompanying text.
3. Square brackets ([]) enclose options; you can include the item enclosed in the brackets or you can omit it, as you choose. If a vertical list of items is enclosed in square brackets, you can combine the options that appear in the list. However, an option set off from the others by blank lines (see /BOOT and /DEVICE in Figure 4–1) indicates that you cannot combine that option with any other option in the list.
4. Braces ({}) enclose options that are mutually exclusive. You can choose only one option from a group of options that appear in braces.

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

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

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

Figure 4–1: Sample Command Syntax



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

dev:filnam.typ

where:

dev:	represents either a logical device name or a physical device name, which is a two- or three-character name from Table 3–1
filnam	represents the one- to six-character alphanumeric name of the file

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

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

```
DX1:TEST.LST
```

You can omit the device name:

```
TEST.LST
```

If you omit the device name, the system assumes that the file is stored on the default device, DK:.

4.1.1 Factoring File Specifications

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

```
DY0:(TEST,A,B).LST
```

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

```
DY0:TEST.LST,DY0:A.LST,DY0:B.LST
```

When you use factoring the device name outside the parentheses applies to each file specification inside the parentheses. Without factoring, the system interprets each file you specify to reside on DK: (the default device) unless you explicitly specify another device name.

Factoring is useful for complicated command lines. It is a general method of string replacement that you can use in many different situations. The following example shows how a command line expands after factoring. Note that the /SYSTEM option appears only once in the resulting output line.

Original command line:

```
.COPY DK:FILE(,2,3).SYS/SYSTEM PR1:
```

Resulting command line (after factoring):

```
.COPY DK:FILE.SYS,DK:FILE2.SYS,DK:FILE3.SYS/SYSTEM PR1:
```

NOTE

The command string that results from the expansion of the line you enter must not exceed 80 characters. 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.

4.1.2 File Type Specification

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

```
.RUN TEST.
```

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

4.1.3 Abbreviating Keyboard Commands

Although the keyboard monitor commands are all English-language words and therefore easy to use, it can become tedious to type words like CROSSREFERENCE and ALLOCATE frequently. You can use as abbreviations the minimum number of characters that are needed to make the command or option unique. Table A–1 in Appendix A lists the minimum abbreviations for the commands and options.

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

CONCATENATE can be shortened to CONC

NOCONCATENATE can be shortened to NOCONC

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

```
.C TEST.LST
```

4.1.4 Keyboard Prompts

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

```
.COPY/CONCATENATE
From?
```

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

```
From? DX1:[TEST.LST]/LST,[DATA.LST]
```

The system prompts you for an output file specification:

```
To ?
```

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

```
To ? DSK:TEST.LST
```

The command now executes.

The system continues to prompt for an input and output file specification until you provide them. If you respond to a prompt by entering only a carriage return, the prompt prints again.

You can combine the normal form of a command with the prompting form, as this example shows:

```
.COPY    ASC.LST
To   ?   DET.LST
```

The system always prompts you for information if any required part of the command is missing.

You can also enter just an option in response to a prompt. The two following examples are equivalent.

```
.COPY
From ?   CHI.MAC/NBLOC
To   ?   CHI.BAK

.COPY
From ?   /NBLOC
From ?   CHI.MAC
To   ?   CHI.BAK
```

4.2 Wildcards

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

An asterisk (*) can replace a file name:

```
*.MAC
```

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

An asterisk (*) can replace a file type:

```
TEST.*
```

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

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

```
*.*
```

The system selects all files on device DK:.

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

```
A*B.MAC
```

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

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

```
A%B.MAC
```

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

Table 4–1 lists commands that support wildcards.

Table 4–1: Commands Supporting Wildcards

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

Note that wildcards work differently with the DIFFERENCES command. See the description of the DIFFERENCES command in Section 4.5 of this manual for more information.

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

Table 4–2: Wildcard Defaults

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

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

```
.COPY/QUERY MYPROG  DX1:
```

The system interprets this command to mean:

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

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

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

```
.DIRECTORY
```

The system interprets this command to mean:

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

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

```
.PRINT  DK:
```

The system interprets this command to mean:

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

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

```
.DELETE/NOQUERY  *.*/PROG
```

The system interprets this to mean:

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

You can use the SET WILDCARDS EXPLICIT command (described in Section 4.5) to change the way the system interprets these commands.

4.3 Editing Command Lines and Terminal Input

The single-line editor (SL) allows you to change a monitor command line, CSI string, or other lines you type at the console. Without using SL, there are only two ways that you can change a line: you can use the delete key to erase characters to the left of the cursor one at a time; or you can type CTRL/U, which erases the entire line, and retype the line. With SL, you can edit parts of a line by moving the cursor to different positions and inserting or deleting characters.

You can use SL only if you have a VT100-compatible video terminal; SL is not available for hard-copy terminals. SL is always available at keyboard monitor level and for background programs. Refer to the *RT-11 Software Support Manual* for information on how to use SL with foreground programs.

To use SL, you must perform the following steps:

1. Make sure the SL handler is installed. If it is not, type:

   ```
   .INSTALL SL
   ```

   If the INSTALL command fails, make sure the characteristics of the SL handler match those of the current monitor by typing:

   ```
   .SET SL SYSGEN
   ```

   Then type INSTALL SL once again.

2. Allow SL to determine what type of terminal you are using, or tell SL what type of terminal you are using. It is recommended that you allow SL to determine your terminal type, by typing:

   ```
   .SET SL ASK
   ```

   If instead you want to specify your terminal type to SL, type:

   ```
   .SET SL VTxxx
   ```

   where xxx represents your terminal type.

3. You must then enable SL by typing:

```
.SET SL ON
```

Then, you can use SL functions as you type on the terminal. When you finish using SL, disable SL by typing the following command:

```
.SET SL OFF
```

4. SLMIN.SYS

SLMIN.SYS is a smaller SL handler for use under SJ/FB monitors. The HELP frame has been removed. SLMIN.SYS cannot be used with the VT52 terminal. To use SLMIN.SYS:

```
.RENAME SL.SYS SLMAX.SYS
.RENAME SLMIN.SYS SL.SYS
.REMOVE SL
.INSTALL SL
```

After you have enabled SL, you can edit console input. If you want the ability to edit responses to prompts printed by the system utilities, you must issue the .SET SL TTYIN command.

See the SET command in Section 4.5 for more information on these commands.

The following sections describe the functions you can perform with SL. Table 4-3 shows the function keys that SL uses. You must perform all edits before you type a line terminator, such as a carriage return (RET). Also, you can use the up-arrow <↑> function to recall the last line.

In the examples, the position of the cursor is indicated by an underline character.

4.3.1 The GOLD Key (PF1)

The GOLD or PF1 key on a VT100 performs no function by itself. This key is always used in combination with another function key to direct SL to perform an alternate function.

If you type a function key without first typing the PF1 key, the regular function is performed. However, if you type the PF1 key and immediately you type a function key, an alternate function is performed. For example, if you type only ←, the cursor moves one position to the left. However, if you type the PF1 key immediately followed by ←, (PF1 ←), the cursor moves to the beginning of the line.

4.3.2 The Help Key (PF2)

The help or PF2 key provides error message and function key information. If you press the PF2 key once, SL prints an error message for the last error that occurred. If you press the PF2 key a second time, SL functions keys are displayed. Return to the original screen by typing any key except PF2. Note that any key you type will then perform its single-line editor function.

To help you learn to use SL, you can lock the display of the SL functions keys on the upper half of your screen by typing the command SET SL

LEARN, then pressing the PF2 key twice (PF2 PF2). You can then use the lower half of your screen to type and edit command lines. To unlock the display, type the command SET SL NOLEARN.

Table 4–3: Single-Line Editor Function Keys

Key(s)	Function
PF1	GOLD prefix key; used with other function keys to perform alternate function operations
PF2	SL help key
←	Move cursor one character to left
→	Move cursor one character to right
PF1 ←	Move cursor to beginning of line
PF1 →	Move cursor to end of line
↑	Reproduce last line terminated with a carriage return (CR)
PF1 ↑	Reproduce line before last
PF4	Delete line from cursor to end of line
PF1 PF4	Restore last line deleted
DELETE	Delete one character to left of cursor
PF1 DELETE	Restore last character deleted
BACKSPACE	Trade positions of two characters; character under cursor switched with character to right
PF1 BACKSPACE	Trade positions of two characters; character under cursor switched with character to left
CTRL/U	Deleted all characters to left of cursor
PF1 CTRL/U	Restore last line deleted
CTRL/R	Redisplay current line
CTRL/W	Redisplay current line
PF1 RETURN	Truncate and execute command line
RETURN	Execute entire command line

4.3.3 Moving the Cursor

Use the ← key to move the cursor one or more characters to the left. For example:

```
.COPY DL0:A.MAC DL1:B.MSC_ ← ←
```

produces:

```
.COPY DL0:A.MAC DL1:B.MSC
```

Use the → key to move the cursor one or more characters to the right. In the example below, the cursor is moved from the T in DELETE to the X in DX0: by typing the → key four times.

```
$DELETE DX0:FILE.TXT → → → →
```

produces:

```
$DELETE DX0:FILE.TXT
```

Instead of moving the cursor to the left one character at a time, you can use the PF1 and ← keys to move the cursor directly to the beginning of a line. For example:

```
$RENAME DY0:FILE.TXT DY1:FILE.BAK PF1 ←
```

produces:

```
$RENAME DY0:FILE.TXT DY1:FILE.BAK
```

Similarly, you can use the PF1 and → keys to move the cursor directly to the end of a line. In the example below, the cursor is moved from the I of FIL1.MAC to the end of the command line by typing the PF1 key followed by the → key.

```
$COPY FIL1.MAC,FIL2.MAC,FIL3.MAC  FILES.MAC PF1 →
```

produces:

```
$COPY FIL1.MAC,FIL2.MAC,FIL3.MAC FILES.MAC
```

4.3.4 Reproduce Last Command Executed

Use the up-arrow (↑) key in response to the monitor prompt ($) or CSI prompt (*) to reproduce the last command that you terminated with a carriage return.

Lines that contain only a carriage return are not stored as the last line. Therefore, if the last line you typed contains only a carriage return, the previous line is reproduced.

After you reproduce the line, you can edit it and execute the new command. The cursor is placed at the end of the line reproduced on the screen.

For example:

```
$LINK RTN1,RTN2,RTN3,RTN4,PROGRM/EXECUTE/MAP<CR>

$↑

$LINK RTN1,RTN2,RTN3,RTN4,PROGRM/EXECUTE/MAP_
```

4.3.5 Delete Line from Cursor to End of Line

Use the PF4 key to delete characters from the cursor to the end of a line. In the example below, typing the PF4 key deletes the D of DY1: and all characters that immediately follow it in the command line.

```
.RENAME FIL1.MAC/FIL2.MAC DY1:MYFILE.MAC[PF4]
```

produces:

```
.RENAME FIL1.MAC/FIL2.MAC
```

4.3.6 Restore Last Line Deleted

Use the PF1 key with the PF4 key, or the PF1 key with CTRL/U, to restore all the characters that you have just deleted on a line. In the example below, the characters after the D in DY1: are deleted by typing the PF4 key, then restored by typing the PF1 and PF4 keys.

```
.RENAME FIL1.MAC/FIL2.MAC DY1:MYFILE.MAC[PF4]
```

produces:

```
.RENAME FIL1.MAC/FIL2.MAC D
```

Then:

```
.RENAME FIL1.MAC/FIL2.MAC D[PF1][PF4]
```

produces:

```
.RENAME FIL1.MAC/FIL2.MAC DY1:MYFILE.MAC
```

Typing [ESC][CTRL/U] instead of [PF1][PF4] produces the same result.

4.3.7 Delete One Character to Left of Cursor

Use the DELETE key to delete the character to the left of the cursor. For example:

```
.COPY A.MAC B.MAC[DELETE]
```

produces:

```
.COPY A.MAC B.MA
```

You can use the PF1 key with the DELETE key to restore the last character that you deleted.

4.3.8 Switch Positions of Two Characters

Use the BACKSPACE key to switch the positions of the character under the cursor and the character to the right of the cursor. The cursor remains on the same character in its new position. For example:

```
$COPY MYFIL1.SAV MYFIL2.SAV[BACKSPACE][RETURN]
```

produces:

```
$COPY MYFIL1.SAV MYFIL2.SAV
```

You can use the PF1 and BACKSPACE keys to switch the positions of the character under the cursor and the character to the left of the cursor. For example:

```
$COPY MYFIL1.SAV MYFIL2.SAV[PF1][BACKSPACE][RETURN]
```

produces:

```
$COPY MYFIL1.SAV MYFIL2.SAV
```

This function is useful to restore the positions of two characters you switched with the BACKSPACE key.

4.3.9 Delete All Characters to Left of Cursor

Use the CTRL/U command to delete all characters from the character to the left of the cursor to the monitor prompt ($) or CSI prompt (*). In the example below, all characters to the left of the colon (:) in DL0: are deleted by typing CTRL/U.

```
$RENAME DL0:MAIN.MAC DL1:SUB1.MAC[CTRL/U]
```

produces:

```
$MAIN.MAC DL1:SUB1.MAC
```

You can type [CTRL/U] to restore the deleted characters.

4.3.10 Truncate and Execute Command Line

Use the PF1 key and the RETURN key to truncate the command line at the cursor and execute the remaining line. In the example below, a command line has been edited and characters have been inserted before DL1:. Type [PF1][RET] to truncate the command line beginning with DL1: and execute the command.

```
$COPY DX0:*.MAC DK1:*.BAK DL1:FILES.BAK[PF1][RET]
```

executes:

```
$COPY DX0:*.MAC DX1:*.BAK
```

4.3.11 Execute Entire Command Line

Use the RETURN key to execute the entire command line, regardless of the position of the cursor. In the example below, the original command line did not include the file LIB2.OBJ. The cursor has been moved back to the middle of the line and LIB2.OBJ has been inserted after LIB1.OBJ. To execute the entire command line, type RETURN.

```
$LINK MYPROG,LIB1.OBJ,LIB2.OBJ,SUB1,SUB2<RET>
```

executes:

```
$LINK MYPROG,LIB1.OBJ,LIB2.OBJ,LIB2.OBJ,SUB1,SUB2
```

4.3.12 Redisplay Current Line

Both CTRL/R and CTRL/W redisplay the current line you are typing or the last line you typed. This function is useful if you are not sure that your screen is displaying information accurately, or if another job prints a message on your screen while you are typing input. When you type either CTRL/R or CTRL/W, the interrupting messages and data are removed and the line you are typing (or the line you just typed) is redisplayed, unaltered.

4.3.13 Redisplay Line Before Last Line Executed

When you press the PF1 (GOLD) key, then the ↑ (UPARROW) key, SL reproduces the line before the last line terminated with a carriage return. This extends the functionality of the ↑ (UPARROW) key by letting you recall an older line for editing.

In the following example, pressing the <GOLD> ↑ recalls the next-to-last command line for editing:

```
$RENAME FILE1.MAC FILE1.BAK<RET>
$<RET>
$ASSIGN DLO: LOG:<RET>
$<GOLD> ↑ RENAME FILE1.MAC FILE1.BAK
```

4.4 Defining Commands with Distributed UCL.SAV

The user command linkage (UCL) feature lets you define your own commands. You can write your own UCL.SAV command parser or you can use the distributed file UCL.SAV.

4.4.1 Command Definition Syntax

To define a command using the distributed UCL.SAV, type a command line with the following syntax:

```
newcommand  :==  oldcommand[/options]"oldcommand'oldcommand...'"
```

where:

newcommand	represents the name of the new command you want to define. The command name can include up to 16 letters and numbers.
,== ==	are the command definition symbols that separate the new command from old commands.
oldcommand	represents an RT-11 keyboard command or a command you have previously defined with UCL.SAV. To include several old commands in a command definition, separate the old commands with a backslash character (\), and terminate the string of old commands with a double backslash (\\). You can include up to four old commands in a command definition. You cannot call indirect files (@filename) in a command definition. The keyboard monitor (KMON) preprocesses anything following an @ sign for expansion as part of its command processing.
/options	represents any valid options for that command.

You can type command definitions in response to the keyboard monitor prompt (.), or you can include them in indirect files. Command definitions are stored in the file UCL.DAT on the system disk SY:. As UCL processes and stores command definitions, it prints:

```
?UCL-I-Processing...
?UCL-I-Successfully entered new command
```

After defining a command, you can use it like any keyboard command in response to the keyboard monitor prompt (.) or in indirect files.

The following defines the new command FILES as the RT-11 keyboard command DIRECTORY including the /VOLUMEID and /COLUMNS:1 options.

```
.FILES ,== DIRECTORY/VOLUME/COLUMNS:1\\
?UCL-I-Processing...
?UCL-I-Successfully entered new command
```

The FILES command processes the RT-11 keyboard command DIRECTORY/VOLUMEID/COLUMNS:1 to produce a single-column directory listing including volume identification and owner name.

The next example defines the command WHEN using two RT-11 keyboard commands, DATE and TIME.

```
.WHEN ,== DATE\TIME\\
```

4.4.2 Defining Command Abbreviations

When you define a command using UCL.SAV, you can choose a command abbreviation by inserting an asterisk (*) in the command name. Characters in the command name that appear before the asterisk constitute the shortest valid abbreviation for the command.

For example, the following defines the new command STATUS, whose shortest valid abbreviation is STAT.

```
.STAT*US :== SHOW ALL\SHOW QUEUE\\@0
```

4.4.3 Displaying Command Definitions

To display a list of UCL commands, run UCL.SAV. The following shows a sample list of UCL commands.

```
.UCL/D
                    User Command Language (UCL)

FILES           :== DIRECTORY/VOLUME:0/COLUMNS:1
WHEN            :== DATE\TIME\\
STAT*US         :== SHOW ALL\SHOW QUEUE\\
```

4.4.4 Adding Information from a Command Line

Some information that you include in a command, such as a file specification, can change with each command. You plan for this kind of information change by including the append symbol (') in command definitions. The append symbol (') lets you tell UCL where to add information given at command time.

For example, the following command definition includes two append symbols where additional information is required.

```
.KILL :== UNPROTECT '\DELETE/NOQUERY '\WHEN\\@0
```

When the KILL command is issued, any text following the word KILL is inserted in place of the append symbols when the command is processed. Suppose you typed this command:

```
.KILL DZ1:*.*/@0
```

The command would perform the following operations:

`UNPROTECT DZ1:*.*`	Unprotect all files on DZ1:
`DELETE/NOQUERY DZ1:*`	Delete, without confirmation, all files on DZ1:.
`WHEN`	Show date and time. Since no append symbol follows the WHEN command, no additional information is included.

4.4.5 Redefining Commands

To redefine a UCL command, type a new command definition. UCL prompts:

```
%UCL-I-Processing...
Command already exists. Replace, Are you sure?
```

Type Y to replace the old definition with the new one. Type N to keep the old definition and ignore the new one.

4.4.6 Deleting Commands

To delete a UCL command, type the command and the definition symbols :==, and press the RETURN key. For example, the following deletes the command WHEN.

```
.WHEN :==
```

UCL prompts:

```
?UCL-I-Processing...
Delete command? Are you sure?
```

Type Y to delete the command or N to save the command.

4.5 Indirect Files

You can group together, as a file, a collection of keyboard commands that you want to execute sequentially. This collection is called an indirect command file, or indirect file. Indirect files are best suited to perform tasks that require a significant amount of computer time and that do not require your supervision or intervention. Any series of commands that you are likely to type often can also run easily as an indirect file.

The indirect file concept is similar to BATCH processing. Although indirect files lack some of the capabilities of BATCH, they are easier to use, use the same commands as normal operations, and generally require less memory overhead than the BATCH processor. (RT–11 BATCH is described in Appendix A of the *RT–11 System Utilities Manual*.)

Another type of file contains a collection of keyboard commands and IND directives that you want to execute. This collection is called an indirect control file, or control file. Chapter 5 of this manual describes IND directives and explains how to create and execute indirect control files.

This section describes how to create indirect command files and how to execute them.

4.5.1 Creating Indirect Files

Create an indirect file by using the EDIT/CREATE command described in Section 4.6. It is conventional to use a .COM file type for an indirect file, but you can choose any file name that you wish. Structure the lines of text to look like keyboard input, placing one command on each line of the file and terminating each line with a carriage return. Do not include the prompt character (.) in the line.

Any keyboard monitor command you can type at the terminal can also be included in an indirect file. The following file, for example, prints the date and time, and creates backup copies of all FORTRAN source files:

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

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

In addition to using the keyboard monitor commands, you can also run one of the RT-11 system utility programs in an indirect file. In this case, structure your input to conform to the Command String Interpreter (CSI) syntax described in Chapter 1 of the *RT-11 System Utilities Manual*. Do not include the CSI asterisk (*) in any line that provides input or output to a utility program.

The following file starts the directory system utility program and lists the directory of two devices on the line printer.

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

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

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

NOTE

If you have a minimal configuration (16K) or a very large indirect command file, use frequent ^C sequences in your indirect files. When the system processes an indirect file, it first places each line in a special memory buffer. This memory

buffer must expand to accommodate each line in an indirect file, and if there are too many lines before the system reaches a ^C, the processor's memory area may become filled. Placing a ^C every 10 or so lines avoids this problem.

Some commands normally require a response from you as they execute. The INITIALIZE command, for example, prints the *Are you sure?* message and waits for you to type Y and a carriage return before it executes. The DELETE command also requests confirmation from you before it deletes a file if you use wildcards in the file specification.

There are three ways to control interaction with the executing command. One way is to use the /NOQUERY option on each command that allows it. This option suppresses the confirmation messages entirely when you use the command in an indirect file.

Another method of interacting applies to a command like DELETE. This command can have a variable number of confirmation queries, if you use a wildcard in the file specification. (If you use no wildcards in the file specification, the DELETE command does not query before deleting the specified file(s).) This type of command accepts your responses directly from the terminal and allows you to make a decision before deleting each file. However, in this case the indirect file cannot operate unattended.

There is yet another way to deal with commands that require a response from you. Both the INITIALIZE and LINK commands have options that cause the system to prompt you for data. This section describes two methods of responding to these prompts, where more than just a Y response is required.

The INITIALIZE command with the /VOLUMEID option permits you to specify a volume ID and owner name for a device. You can place your responses in the indirect file, as this example shows:

```
INITIALIZE/NOQUERY/VOLUMEID DT:
TAPE8
PAYROLL
```

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

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

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

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

```
LINK/TRANSFER MYPROG/OPT
0,OPT
```

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

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

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

NOTE

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

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

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

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

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

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

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

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

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

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

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

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

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

4.5.2 Executing Indirect Files

You can use indirect files to supply input to jobs running under the SJ monitor and to background jobs running under the FB or XM monitor. Indirect files are unavailable for foreground or system jobs.

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

@ filespec (when SET KMON NOIND is in effect)

or,

$@filespec (always)

where:

$	is the command that causes KMON to execute a file as an indirect file when SET KMON IND is enabled
@	is the monitor command that indicates an indirect file
filespec	represents the name and file type of the indirect file, as well as the device on which it is stored. The default file type is .COM

If you omit the device specification, DK: is assumed. If you specify any other block-replaceable device, the monitor automatically loads the handler for that device.

After you type the SET KMON IND command, the command syntax @filespec causes IND (the Indirect Control File Processor) to execute the specified file as an IND control file. Once SET KMON IND is in effect, you can force KMON to execute a file as an indirect command file by typing a dollar sign ($) before the at sign (@) in the command line as follows:

```
$@filespec
```

Use the SET KMON NOIND command to disable IND processing. See Section 4.6 of this manual for more information on the SET KMON [NO]IND command.

Note that indirect control files are quite different from indirect command files. Chapter 5 of this manual describes IND and indirect control files in detail.

It is conventional to type the indirect file command directly in response to the monitor's prompt, as this example shows:

```
.@INDCT
```

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

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

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

```
*.BAK
```

This is the command that will actually execute:

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

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

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

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

each operation and can abort the indirect file as soon as it detects an error. A better way to perform the same operations as the indirect file shown above is as follows:

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

You can use the SET ERROR command, described in Section 4.6, to define the severity of error that causes an indirect file to stop executing.

Normally, as each line of an indirect file executes, it echos on the console terminal so that you can observe the progress of the job. However, you can use the SET TT QUIET command, described in Section 4.6, to suppress this printout. In this case, only the prompting messages, if any, print.

You can stop execution of an indirect file at any time by typing two CTRL/C characters. Control returns to the monitor and you can enter a new command. You can also abort the indirect file by typing a single CTRL/C in response to a query or prompt. If you use an indirect file to execute a MACRO program, read the appropriate section in the *RT-11 Programmer's Reference Manual* to learn about certain restrictions on using the .EXIT call with indirect files.

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

```
.@FIRST
```

The file FIRST.COM contains these lines:

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

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

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

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

4.5.3 Start-Up Indirect Files

Section 3.1 introduced the start-up indirect command files: STARTS.COM (for SJ), STARTF.COM (for FB), and STARTX.COM (for XM). Each monitor automatically invokes its own indirect command file when you bootstrap the system, and you can modify these files to perform standard system configurations. Since many of the system parameters are reset by a bootstrap operation (see the SET command, Section 4.6), you should use the start-up indirect files to set the system parameters you normally use.

For example, if you use the FB monitor and have a visual display console terminal that supports hardware tabs, add the SET TT: SCOPE and SET TT: TAB commands to the file STARTF.COM. You could also include a SET TT: QUIET command at the beginning of STARTF.COM and a SET TT: NOQUIET command at the end to suppress extra type-out at bootstrap time. If you have a list of commands that you need to execute, regardless of the monitor you bootstrap, include these commands in a separate indirect file, such as COMMON.COM, and invoke this file from all three start-up indirect files. The following example shows a typical STARTF.COM file.

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

You can also use the start-up indirect files to run your own programs, set the date, or do other file maintenance operations. You can use IND to run an interactive dialog to assign devices and load handlers.

4.6 Keyboard Monitor Commands

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

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

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

NOTE

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

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

RT–11 permits you to remove some of the monitor commands at system generation time. If you type a command that the system does not recognize as a keyboard monitor command, the system checks the concise command language (CCL) and user command linkage (UCL) tables. If the command is not part of your system, the system prints an error message. See Section 4.7 for instructions on using CCL. Refer to Section 4.4 for more information on UCL.

ABORT

The ABORT command, typed on the system console, aborts a foreground or system job that has been assigned a private terminal for communication. (See the description of the FRUN or SRUN /TERMINAL:n option in this chapter for more information.)

```
ABORT [/F] [jobname]
```

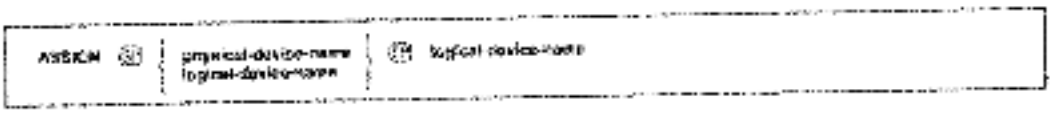
When you type the ABORT command at the system console, the foreground or system job is immediately aborted.

You can enter the ABORT command as one line, or you can rely on the system to prompt you. If you type ABORT followed by a carriage return, the system prompts *Jobname?*.

When running under an FB or XM monitor, if your monitor 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.

ASSIGN

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



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

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

If the logical name you supply is already associated with a physical device, the system disassociates the logical name from that device and assigns it to the current device. You can assign only one logical name with each ASSIGN command, but you can use several ASSIGN commands to assign different logical names to the same device. Note that SA and SY are always invalid as logical device names.

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

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

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

The following command redirects the printer output to the terminal.

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

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

The next command redefines the default file device.

```
$ASSIGN DL1: SY:
```

If after executing this command you supply a file specification in a command and omit the device name, it now defaults to DL1:. Note that this does not affect the default system device, SY:.

The last example is typical for a system that uses a dual-drive diskette device. Several users can share the same system software on 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: SY:
```

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

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

```
B [ @ address]
```

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

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

The following command sets the base to 0.

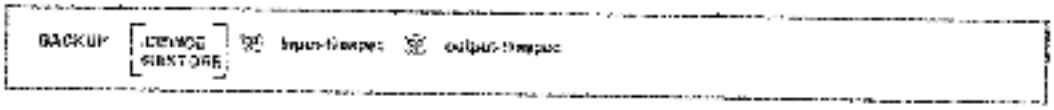
```
.B
```

The next two commands both set the base to 1000.

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

BACKUP

The BACKUP command backs up and restores RT-11 files or volumes.



The BACKUP command copies the contents of a large file or an entire volume to a set of smaller volumes. Since you can not use the file or volume while it is fragmented on several smaller volumes, you should use the BACKUP command only as a means of storing information.

The BACKUP command also performs the reverse operation of restoring the fragmented file or volume to its original form on a single large volume so you can again use the file or volume.

In the command syntax shown above, input-filespec represents the data to copy. Output-filespec represents the device or file to receive the data. You cannot use wildcards with the BACKUP command; when storing a file with the BACKUP command you must specify the input file name and device. The output file name is the same as the input file unless otherwise specified.

When copying an entire volume, the output file name is the two-letter device mnemonic of the volume you are copying, unless otherwise specified. In either case, the default output file type is .BUP. For example, the system assigns the name DL.BUP to the output file on the diskettes to which the RL02 is copied.

```
.BACKUP/OBJECT DL0: DY0:
```

You can use random-access volumes 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. If you use TSV05 magtapes as backup volumes, you must set the Extended Features Switch (switch S0 on switch pack E58) if you want the tape to stream at 100 in/s. See Appendix A of either the *TSV05 Installation Guide* or the *TSV05 User's Guide* for more information on setting the Extended Features Switch.

You can use the BACKUP command to store a file or volume only if the input file is larger than the type of output volume you are using unless the output volume is magtape. Otherwise, if you use the BACKUP command to copy a file or volume that fits on only one of the selected output volumes, an error occurs.

When you use the BACKUP command, the system copies as much of the input as will fit on the first output volume. When that volume becomes full, the system prompts you to mount another volume in the same drive unit. As each output volume is filled, the system notifies you which volume is being created, so you can label the volumes accordingly. The process continues until the entire file or volume has been copied.

The output volumes must be specially initialized as backup volumes (see the INITIALIZE command). If you mount a volume that is not a backup volume, is not an RT–11 block-replaceable volume, or already contains files, the system notifies you. You can then choose to replace that output volume with another backup volume, or you can allow the system to initialize the output volume that is already mounted without disrupting the backup operation.

When you use the BACKUP command without any option, the system backs up the specified input file to a set of volumes successively mounted in the specified output device. In the following example, the file MYPROG.MAC is backed up to several double-density diskettes. The system detects that the second output diskette has not been initialized as a backup volume.

```
BACKUP DL0:MYPROG.MAC DY0:
Mount output volume in DY0:?  Continue? Y
?BUP-I-Creating volume <1>
Mount next output volume in DY0:?  Continue? Y
?BUP-W-Not a backup volume DY0:
DY0:/?BUP Initialize?  Are you sure? Y
?BUP-I-Bad block scan started...
?BUP-I-No bad blocks detected
?BUP-I-Creating volume <2>
Mount next output volume in DY0:?  Continue? Y
?BUP-I-Creating volume <3>
```

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

/DEVICE Use this option to back up an entire volume to several smaller volumes, or use /DEVICE with /RESTORE to restore a volume from a set of backup volumes. In the following example, an RL02 disk is backed up to several double-density diskettes. The system detects that the second output diskette has not been initialized as a backup volume.

```
BACKUP/DEVICE DL0: DY0:DL.BUP
Mount output volume in DY0:?  Continue? Y
?BUP-I-Creating volume <1>
Mount next output volume in DY0:?  Continue? Y
?BUP-W-Not a backup volume DY0:
DY0:/?BUP Initialize?  Are you sure? Y
?BUP-I-Bad block scan started...
?BUP-I-No bad blocks detected
?BUP-I-Creating volume <2>
Mount next output volume in DY0:?  Continue? Y
?BUP-I-Creating volume <3>
```

/RESTORE This option restores to its original state a file you have backed up using the BACKUP command. Use /DEVICE with /RESTORE to restore an entire volume.

When restoring a file, if you specify no input file name the system uses the name of the file on the volume you specify. The default file type is .BUP. If you specify no output file, the system uses the input file name and type.

The system prompts you to mount each volume of the set that contains the full volume or file, and copies the contents of each volume to the volume you specify. If you mount the input volumes in the wrong order, or if you mount a volume that contains the wrong file, the system notifies you and reprompts you to mount the correct volume. The system also notifies you when it has finished the restore operation.

The following command restores the volume DL: from several RX02 diskettes to a single RL02 disk.

```
$BACKUP/DEVICE/RESTORE DY0:DL.FDF DL1:
```

Note:

The BASIC command no longer exists.

BOOT

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

In the command syntax illustrated above, filespec represents the device or monitor file to be bootstrapped. If you omit filespec, the system prompts you with *Device or file?*. The BOOT command can perform either of two operations: a hardware bootstrap of a specific device, or a direct bootstrap of a particular monitor file without using the bootstrap blocks on the device. When you bootstrap a volume, make sure that the appropriate device handler is present on that volume.

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

DD0:-DD1:	DX0:-DX1:
DK:	DZ0:-DZ1:
DL0:-DL3:	DY0:-DY1:
DM0:-DM7:	DU0.-DU7:
DS0:-DS7:	RK0:-RK7:
DW:	SY:

NOTE

The following unsupported devices are also valid for the BOOT command:

DP0:-DP7:
DT0:-DT7:
PD0:-PD1:
RF:

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

```
.BOOT DK:

RT-11SJ  V05.00
```

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

You can use the BOOT command to alternate between the SJ and FB monitors. When you use the BOOT command to change monitors you do not have to reenter the date and time. The system clock, however, may lose a few seconds during a reboot. The next example bootstraps the FB monitor on device SY:.

```
.BOOT SY:FB

RT-11FB V05.00
```

NOTE

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

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

/WAIT This option 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 you use this option, the system initiates the BOOT procedure but then pauses and waits for you to mount the volume you want to bootstrap. When the system pauses, it prints *Mount input volume in <device>; Continue?* at the terminal, where <device> represents the device into which you mount the volume. Mount the volume you want to bootstrap, then type Y or any string beginning with Y followed by a carriage return. Type N or any string beginning with N, or two CTRL/Cs, to abort the operation and return control to the original monitor. Any other response causes the message to repeat. Make sure DUP is on the system volume when you use the /WAIT option.

The following sample command line boots an RK05 disk:

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

CLOSE

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

```
close
```

The CLOSE command accepts no options or arguments.

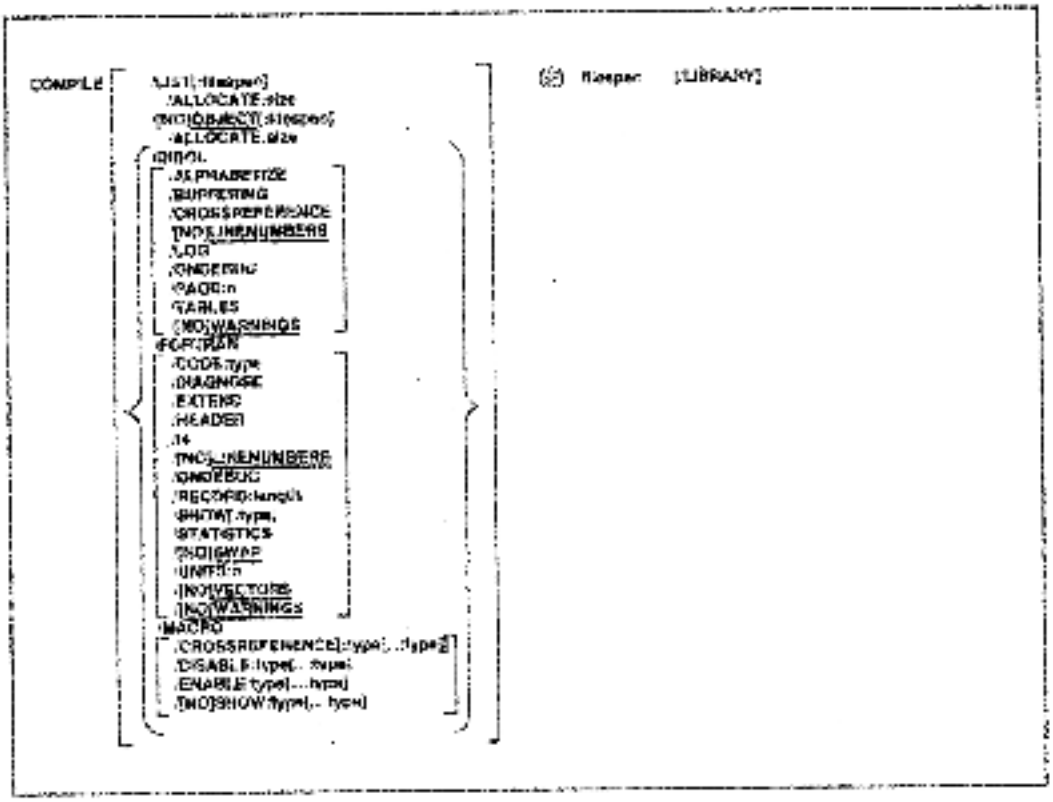
You can use the CLOSE command to make tentative open files permanent; otherwise, they do not appear in a normal directory listing and the space associated with the files is available for reuse. The CLOSE command is particularly useful after you type a CTRL/C to abort a background job. You can also use it after an unexpected program termination to preserve any new files that were being used by the terminated program. 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.

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

```
.R PREC

.

.

CTRL/C
.CLOSE
```

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



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

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

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

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

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

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

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

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

/ALLOCATE:size Use this option with /LIST or /OBJECT to reserve space on the device for the output file. The argument size represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 65535. A value of –1 is a special case that creates the largest file possible on the device.

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

/BUFFERING Use this option with /DIBOL to direct the compiler to use single buffering for I/O. Normally the compiler uses double buffering.

/CODE:type Use this option with /FORTRAN to produce object code that is designed for a particular hardware configuration. The argument type represents a three-letter abbreviation for the type of code to produce. The 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.

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

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

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

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

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

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

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

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

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

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

/LIBRARY Use this option with /MACRO to identify a macro library file; use it only after a library file specification in the command line. The MACRO assembler looks first in 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/MACRO:C/LIBRARY:MYLIB,SYSMAC/LIST/OBJECT
```

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

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

/LIST[:filespec] You must specify this option to produce a compilation or assembly listing. The /LIST option has different meanings depending on its position in the command line. Note that anytime you type a colon after the /LIST option (/LIST:) you must specify a device or a file specification after the colon.

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 follow /LIST with a device name, the system creates a listing file on that device. If the device is a file-structured device, the system stores the listing file on that device, assigning it the first input file name and a .LST file type. The following command produces a listing on the terminal:

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

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

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

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

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

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

```
*COMPILE/DIBOL A+B/LIST:RX3:
```

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

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

Both the commands shown above generate 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) 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 Use this option with /DIBOL to create a log of error messages generated by the compiler.

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

/OBJECT[:filespec] Use this option to specify a file name or device for the object file. Note that anytime 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 two commands have the same meaning:

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

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

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

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

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

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

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

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

/ONDEBUG Use this option with /DIBOL to include 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 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.

/PAGE:n Use this option with /DIBOL to override the default listing page length of 66 lines. The meaningful range of values for the decimal argument n is 1 to 32768 (decimal).

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

/SHOW:type Use this option with /FORTRAN to control FORTRAN listing format. The argument type represents a code that indicates which listings the compiler is to produce. Table 4–6 summarizes the codes and their meaning.

Use this option with /MACRO to specify any MACRO .LIST directive. Table 4–13 summarizes the valid arguments and their meaning. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives.

/NOSHOW:type Use this option with MACRO to specify any MACRO .NLIST directive. Table 4–13 summarizes the valid arguments and their meaning. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives.

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

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

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

/TABLES Use this option with /DIBOL to generate 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.

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

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

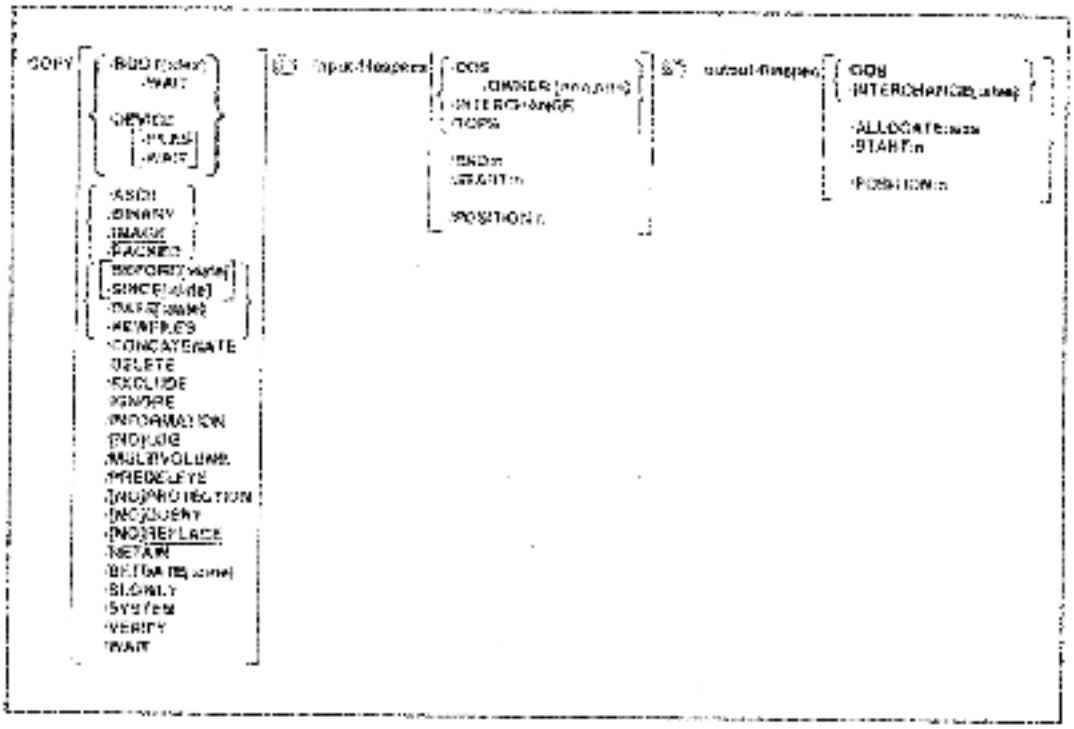
/NOVECTORS This option directs FORTRAN to use multiplication operations to access multidimensional arrays.

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

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

COPY

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



The COPY command performs the following 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 to a volume
- The contents of a volume to a file and vice versa
- The contents of a device to another device

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

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

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

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

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

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

```
.COPY *.MAC *.BAK
```

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

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

The system has a special way of handling system (.SYS) files and files that cover bad blocks (.BAD files). The system requires you to use the /SYSTEM option when you need to copy system files and wildcards are used in the input file type, or when you use the /EXCLUDE option. You cannot copy system files simply by placing wildcards in file specifications. To copy a .BAD file, you must specify it by explicitly giving its file 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.

The following sections describe the COPY command options and include command examples. Some of the options accept a date as an argument. The syntax for specifying the date is:

[dd][:mmm][:yy]

where:

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

The default value for the date is the current system date. If you 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 :82 and the current

system date is May 4, 1983, the system uses the date 4-MAY-82. 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.)

/ALLOCATE:*size* Use this option after the output file specification to reserve space on the device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 65535. A value of –1 is a special case that creates the largest file possible on the device.

/ASCII This option copies files in ASCII mode, ignoring and eliminating nulls and rubout characters. It converts data to the ASCII 7-bit format and treats CTRL/Z (32 octal) as the logical end-of-file on input. Files that consist of ASCII-format data include source files you create with the editor, map files, and list files. The /ASCII option cannot be used with /VERIFY.

The following example copies a FORTRAN source program from DY0: to DY1:, giving it a new name, and reserving 50 blocks of space for it.

```
.COPY/ASCII DY0:MATRIX.FOR DY1:TEST.FOR/ALLOCATE:50
```

/BEFORE[:*date***]** This option 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, 1983.

```
.COPY/BEFORE:4-FEB-83 *.MAC DLO:*.MAC
Files copied:
DK:A.MAC                to DLO:A.MAC
DK:B.MAC                to DLO:B.MAC
DK:C.MAC                to DLO:C.MAC
```

/BINARY Use this option to copy formatted binary files, such as .OBJ files produced by the assembler or the FORTRAN compiler, and .LDA files produced by the linker. The system verifies checksums and prints a warning if a checksum error occurs. If this happens, the copy operation does not complete. The /BINARY option cannot be used with /VERIFY.

The following command copies a binary file from DK: to a diskette.

```
.COPY/BINARY ANALYZ.OBJ DY1:*.*
```

Note that you cannot copy library files with the /BINARY option because a checksum error occurs. Copy them in image mode.

/BOOT[:dev] This option 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. The optional argument dev represents a two-letter target system device. This argument is especially useful when you are creating a bootable RX01 system while the current system is on an RX02 diskette. Note that you cannot combine /BOOT with any other option, and that your input and output volume must be the same. Also note that you can name your monitor file any name you wish; 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 RL02 disk, you must have the handler file DL.SYS on that RL02.

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

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

The following example creates a system diskette.

```
.INITIALIZE DY1:
DY1:/Initialize: Are you sure? Y
.COPY/SYSTEM DY0:*.* DY1:*.*
 Files copied:
DY0:RT11FB.SYS        to DY1:RT11FB.SYS
DY0:SWAP.SYS          to DY1:SWAP.SYS
DY0:DT.SYS            to DY1:DT.SYS
DY0:DX.SYS            to DY1:DX.SYS
DY0:LP.SYS            to DY1:LP.SYS
DY0:DIR.SAV           to DY1:DIR.SAV
DY0:DUP.SAV           to DY1:DUP.SAV
DY0:ABC.MAC           to DY1:ABC.MAC
DY0:ABS.MAC           to DY1:ABS.MAC
DY0:DT.SYS            to DY1:DT.SYS
DY0:PIP.SAV           to DY1:PIP.SAV
DY0:MT.SYS            to DY1:MT.SYS
DY0:MN.SYS            to DY1:MN.SYS
DY0:COMB.CAT          to DY1:COMB.CAT

.COPY/BOOT DY1:RT11FB.SYS DY1:
```

The following example creates a bootable RX01 system diskette on an RX02 drive:

```
.COPY/BOOT:DX  DY0:RT11SJ.SYS  DY0:
```

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 Use this option to combine several input files into a single output file. This option is particularly useful to combine several object modules into a single file for use by the linker or librarian.

The following command combines all the .FOR files on DY1: into a file called MERGE.FOR on DY0:.

```
.COPY/CONCATENATE DY1:*.FOR DY0:MERGE.FOR
 Files copied:
DY1:A.FOR               to DY0:MERGE.FOR
DY1:B.FOR               to DY0:MERGE.FOR
DY1:C.FOR               to DY0:MERGE.FOR
```

Wildcards are invalid in the output file specification.

/DATE[:date] Use this option to copy only those files with a certain creation date. If no date is specified the current system date is used. The following command copies all .MAC files created on February 20, 1983 from DL0: to DL1:

```
.COPY/DATE:20-FEB-83 DL0:*.MAC DL1:*.*
 Files copied:
DL0:A.MAC               to DL1:A.MAC
DL0:B.MAC               to DL1:B.MAC
DL0:C.MAC               to DL1:C.MAC
```

/DELETE Use this option to delete 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 example copies JSPROG.SAV to DY1:, then deletes it from device DK:.

```
.COPY/DELETE JSPROG.SAV DY1:JSPROG.SAV
```

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

If one device is smaller than the other, the system copies only as many blocks as the smaller device contains. For example, if you copy a large volume to a smaller one, you 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 1K-word 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 DY0: to DY1:.

```
.COPY/DEVICE DY0:DY1:
DY1:/COPY? Are you sure? Y
```

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

NOTE

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

The /DEVICE and /POSITION options are mutually exclusive.

/DOS Use this option to transfer files between RSTS/E or DOS–11 format and RT–11 format. The option must appear in the command line after the file to which it applies. Valid 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:[70,6,SRC/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
```

/ENDn Use with /STARTn and /DEVICE to specify the last block of the volume you are copying. The /ENDn notation must follow the input file

specification. The argument n represents a decimal block number. The following example copies blocks 0 to 500 from DL0: to DL1:, starting at block 501, in a file named ADAM.MAC:

```
.COPY/DEVICE/FILES DL0:/START:0/END:500  DL1:ADAM.MAC/START:501
```

/EXCLUDE This option copies all the files on a device except the ones you specify. The following command copies all files from DY0: to DY1: except .OBJ and .SAV files.

```
.COPY/EXCLUDE DY0:[*,*](*.OBJ) DY1:*.*
```

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

/FILES Use with /DEVICE to copy a volume to a file on another volume or vice versa. If you use 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 /FILES option cannot be used with /WAIT.

The following example copies diskette DY0: to DL1: as file FLOPPY.BAK:

```
.COPY/DEVICE/FILES DY0: DL1:FLOPPY.BAK
```

The following example copies file DECTAP.BAK to DD0:

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

/IGNORE Use this option to ignore errors during a copy operation. /IGNORE forces a single-block data transfer, which you can invoke at any other time with the /SLOWLY option. Use /IGNORE if an input error occurred when you tried to perform a normal copy operation. This procedure can sometimes recover a file that is otherwise unreadable. If there is still an error, an error message prints on the terminal, but the copy operation continues. This option is invalid with /DOS, /TOPS, and /INTERCHANGE.

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

1. For ASCII data, there is no check for nulls.

2. For binary data, there is no checksum consideration.

This command copies a text file to a double density diskette for storage:

```
.COPY LETTER.SAV DY0:*.*
```

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

/INFORMATION Use this option to change the severity level of the error message that prints when not all of the input files you specified are found. If you do not use /INFORMATION, the system prints an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, the system prints 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 copied to DL1:. However, since the system is unable to find DL0:FILE2.TXT, the system prints a message to inform you.

```
.COPY/INFORMATION DL0:(FILE1,FILE2,FILE3).TXT  DL1:*.*
PIP-I-File not found DL0:FILE2.TXT
```

/INTERCHANGE[:size] This option transfers data in interchange format between interchange diskettes that are compatible with IBM 3741 format and RT–11 block-replaceable devices or the console. The option must appear in the command line after the file to which it applies. If the output file is to be in interchange format, you can specify the length of each record. The argument size represents the record length in characters (the default record length is 80 bytes).

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 WAIT.MAC from device DK: to device DX1: in interchange format, giving it the name WAIT.MA. The record length is set to 128 (decimal) bytes.

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

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

The following example shows a copy command line and the resulting log.

```
.COPY/LOG DY1:FILE.MAC  DY0:FILE.MAC
 Files copied:
DY1:FILE.MAC          to DY0:FILE.MAC
```

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

/MULTIVOLUME Use this option to copy 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 the system copies files to the output volume until the system finds a file that will not fit. The system 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, the system prompts you to mount the next output volume and prints the *Continue?* message. Mount another output volume of the same type and type Y or any string beginning with Y to continue. The system begins the copy operation with the first file that did not fit on the previous output volume. If you type N or any string beginning with N, or two CTRL/C, the operation is not completed and the monitor prompt (.) appears. Any other response causes the prompt to repeat. The system 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, the system 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 DL0: being copied to several double-density diskettes:

```
.COPY/MULTIVOLUME DL0:*.* DY0:
(Log of files copied)
Mount next output volume in DY0:!  Continue? Y
(Log of files copied)
Mount next output volume in DY0:!  Continue? Y
(Log of files copied)
Mount next output volume in DY0:!  Continue? Y
(Log of files copied)
Mount next output volume in DY0:!  Continue? Y
```

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

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

```
.COPY/NEWFILES *.* DY1:*.*
 Files copied:
DK:A.FOR                to DY1:A.FOR
DK:B.FOR                to DY1:B.FOR
DK:C.FOR                to DY1:C.FOR
```

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

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

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

The /POSITION and /DEVICE options are mutually exclusive.

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

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

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

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

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

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

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

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

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

Chapter 13 of the *RT-11 System Utilities Manual*, Section 13.2.1, contains more detailed information about operations involving magtape.

/PREDELETE This option 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, the system 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. Be careful when you use the /PREDELETE option; if for any reason the input file is unreadable, the output file will already have been deleted and you are left with no usable version of the file.

/PROTECTION Use this option to give 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 Use this option to enable 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 and active console log files are protected. You should not use this option to remove protection from an active logical disk file.

/QUERY If you use this option, the system requests confirmation from you before it performs the operation. /QUERY is particularly useful on operations that involve wildcards, when you may not be sure which files the system selected for an operation. The /QUERY option is valid on the COPY command only if both input and output are in RT-11 format.

Note that if you specify /QUERY in a copy command line that also contains a wildcard in the file specification, the confirmation messages that print on the terminal replace the log messages that would normally appear. You must respond to a query message by typing Y (or any string that begins with Y) and a carriage return. The system interprets any other response to mean NO, and it does not copy the file.

The following example copies three of the four .MAC files stored on DK: to DY1:.

```
.COPY/QUERY DK:*.MAC DY1:*.*
 Files copied:
DK:A.MAC                to DY1:A.MAC      ? Y
DK:B.MAC                to DY1:B.MAC      ? Y
DK:C.MAC                to DY1:C.MAC      ? N
DK:DEMOF1.MAC           to DY1:DEMOF1.MAC? Y
```

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

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

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

/RETAIN Use this option with the /DEVICE option to preserve the bad block table of the output volume. The input and output volumes must be alike and must support bad block replacement. You must have initialized the output volume by using the INITIALIZE/REPLACE command before you can use this option with /DEVICE. The /RETAIN option is invalid with the /START, /END, and /FILES options.

The following example copies the volume DL0: to DL1:, but preserves DL1:'s bad block replacement table.

```
.COPY/DEVICE/RETAIN DL0: DL1:
```

/SETDATE[:date] This option causes the system to put the date you specify on all files it transfers. 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, the system preserves the existing file creation date when it copies a file block for block.

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

/SINCE[:date] This option 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, 1983.

```
*COPY/SINCE:24-FEB-83 *.MAC DL0:*.MAC
 Files copied:
DK:A.MAC                to DL0:A.MAC
DK:B.MAC                to DL0:B.MAC
DK:C.MAC                to DL0:C.MAC
```

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

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

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

The following example copies blocks 500 to 550 of DL0: to DL1: starting at block 100:

```
*COPY/DEVICE DL0:/START:500/END:550  DL1:/START:100
```

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

/SYSTEM Use this option if you need to copy system (.SYS) files and 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 printed on the terminal to remind you.

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

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 RT-11 device RK0:.

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

/VERIFY Use this option to verify 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 printed on the terminal. This option cannot be used with /ASCII or /BINARY.

/WAIT Use this option to copy 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 your system has dual drives but the system volume is neither the input nor output volume. When you use this option, the system initiates execution of a command but then pauses and prints the message *Continue?*. At this time, you can remove the system disk and mount the disk on which you want the operation to take place. Mount the new disk and type a Y or any string beginning with Y, followed by a carriage return, to resume the operation. If you type N, or any string beginning with N, or two CTRL/Cs, and the system volume is still in place, the operation is not performed and the keyboard monitor prompt (.) appears. If the system volume is not in place, the system prompts you to remount the system volume before the system aborts the operation. Any other response causes the message to repeat.

When the operation completes the system prints the *Continue?* message again. Mount the system volume and type a Y or any string beginning with Y followed by a carriage return. If you type any other response the system prompts you to mount the system volume until you type Y. The system then prints the keyboard monitor prompt. Make sure PIP, DUP, and FILEX (if necessary) are on your system volume when you use the /WAIT option.

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.

The /WAIT option cannot be used with /FILES.

Single-Volume Operation

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

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

   COPY/WAIT input-filespec output-filespec

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

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

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

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

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

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

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

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

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

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

Double-Volume Operation

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

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

   .COPY/WAIT input-filespec output-filespec

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

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

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

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

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

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

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

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

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

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

CREATE

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

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

If you attempt to create a file over a tentative file (one that was opened but never closed) and the foreground job is loaded, the system 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; this will corrupt the file and cause unpredictable results.

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

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

/ALLOCATE:size Use this option following the file specification to allocate 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. If you do not use /ALLOCATE, the system assumes one block.

/EXTENSION:n Use this option to extend an existing file by the number of blocks you specify; n 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).

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

```
.DIRECTORY/FULL DK0:
 08-SEP-83
MYPROG.MAC          SCP 18-JAN-83        YN      .MAC      25   27-JAN-83
UTMAC .MAC;         7   18-JAN-83        SYSMAC.MAC        41   18-JAN-83
< UNUSED >          22                   RT11SJ.SYS        67   18-JAN-83
```

```
TT     .SYS          2   19-JAN-83       DX     .SYS      3   19-JAN-83
LELB   .LBM          1   09-FEB-83       BUILD  .MAC     80   19-JAN-83
< UNUSED >         199
 3 Files, 282 Blocks
 224 Free blocks
```

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

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

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

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

```
.DIRECTORY/DELETED DX0:
 09-FEB-83
SWAP   .SYS     25   19-JAN-83       17  EMPTY.FIL  178   31-JAN-83  315
 0 Files, 0 Blocks
 204 Free blocks
```

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

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

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

D

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

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

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

The D 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.

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

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

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

The next command sets the base address to 0.

```
.B
```

The following command deposits 3705 into location 1000.

```
.D 1000=3705
```

The next command sets the relocation base to 1000.

```
.B 1000
```

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

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

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

```
DATE [dd-mmm-yy]
```

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

To enter a date into the system, 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 the system.

The following example enters the current date.

```
.DATE 18-May-83
```

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

```
.DATE
18-May-83
```

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

DEASSIGN

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

```
DEASSIGN [ logical-device-name]
```

In the command syntax illustrated above, logical-device-name represents 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.

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:
```

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

```
.DEASSIGN INP:

%DCL-W-Logical name not found INP:
```

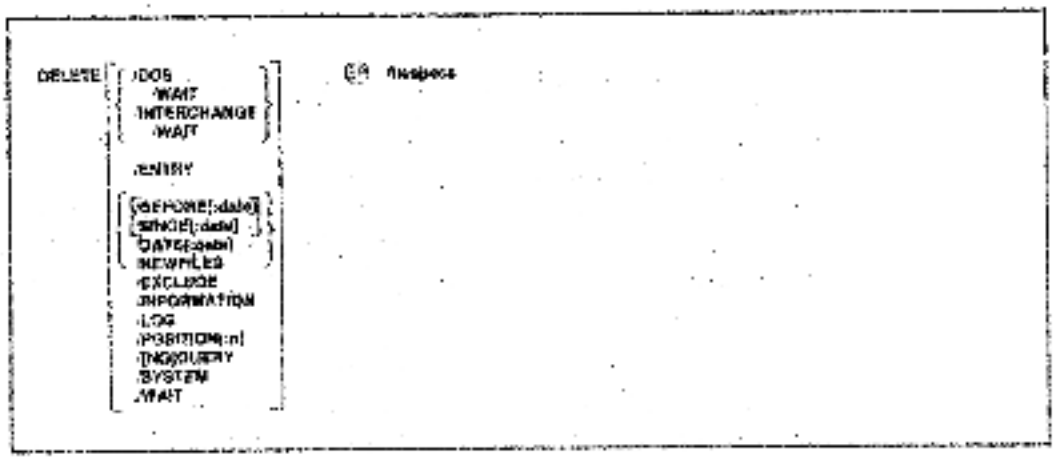
To disassociate all logical names from physical devices, type the DEASSIGN command without an argument. The following example disassociates all logical device names (except SY:) from physical devices and resets the logical names DK: and SY: to represent the system volume.

```
.DEASSIGN
```

If DK: is assigned to a nonsystem device (such as DY1:), the following command disassociates DK: from DY1: and restores the default association of DK: to SY:, the system device.

```
.DEASSIGN DK:
```

The DELETE command deletes the files you specify.



In the command syntax shown above, filespec represents the file(s) to be deleted. You can specify up to six files; separate them with commas. You can enter the DELETE command as one line, or you can rely on the system to prompt you for information. If you omit the file specification, the DELETE command prompts *Files?*. If you delete a file accidentally, it may be possible to recover the file if you act immediately (see CREATE). A procedure for doing this is described in Chapter 6 of the *RT–11 System Utilities Manual*.

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

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

Another feature of the DELETE command is that, unless you use /LOG or /NOQUERY, the system 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 followed by a carriage return in order to execute the command.

The following sections describe the options you can use with the DELETE command. Some of the options accept a date as an argument. The syntax for specifying the date is:

[dd][:mmm][:yy]

where:

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

The default value for the date is the current system date. If you 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 :82 and the current system date is May 4, 1983, the system uses the date 4:MAY:82. 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.)

/BEFORE[:date] Use this option to delete 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 DY0: that were created before March 20, 1983.

```
.DELETE/LOG/BEFORE:20:MAR:83 DY0:*.SAV
  Files deleted:
DY0:A.SAV
DY0:B.SAV
DY0:C.SAV
```

/DATE[:date] Use this option to delete 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, 1983.

```
.DELETE/LOG/DATE:20:FEB:83 DK:*.MAC
  Files deleted:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

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

/ENTRY Use this option to delete a job from the queue. Use /ENTRY when QUEUE is running as a foreground or system job (see Chapter 17 of the *RT–11 System Utilities Manual*, Queue Package).

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

```
$DELETE/ENTRY MILLER
```

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

/EXCLUDE This option deletes all the files on a device except the ones you specify. The following command, for example, deletes all files from DY0: except .SAV files:

```
$DELETE/EXCLUDE DY0:*.SAV
%PIP-W-No ,SYS option
 Files deleted:
DY0:ABC.OLD    ? Y
DY0:AAF.OLD    ? Y
DY0:COM3.      ? Y
DY0:MERGE.OLD  ? Y
```

/INFORMATION Use this option to change the severity level of the error message that prints when not all of the input files you specified are found. If you do not use /INFORMATION, the system prints an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, the system prints 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 the system is unable to find DLO:FILE2.TXT, the system prints a message to inform you.

```
$DELETE/INFORMATION DLO:[FILE],FILE2,FILE3].TXT
%PIP-I-File not found DLO:FILE2.TXT
```

/INTERCHANGE Use this option to delete from a diskette a file that is in interchange format. /WAIT is the only option you can use with /INTERCHANGE.

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

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

```
$DELETE/NEWFILES DY1:*.BAK
 Files deleted:
DY1:MERGE.BAK ? Y
```

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

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

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

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

/QUERY Use this option to request confirmation before the system deletes each file. This option is particularly useful on operations that involve wildcards, when you may not be completely sure which files the system selected for the operation. This is the default mode of operation 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, and a carriage return to initiate execution of a particular operation. The system interprets any other response as NO; it does not perform the operation.

The following example shows querying. Only the file DX1:AAF.MAC is deleted.

```
.DELETE/QUERY DY1:*.*
 Files deleted:
DY1:ABC.MAC    ? N
DY1:AAF.MAC    ? Y
DY1:MERGE.FOR  ? N
```

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

/SINCE[:date] Use this option to delete 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 DY0: that were created on or after March 20, 1983.

```
.DELETE/LOG/SINCE:20:MAR:83 DY0:*.SAV
 Files deleted:
DY0:A.SAV
DY0:B.SAV
DY0:C.SAV
```

/SYSTEM Use this option if you need to delete system (.SYS) files and you use wildcards in an input file type. If you omit this option, the system files are excluded from the DELETE operation, and a message is printed on the terminal. (Note that the system prints this message only when system files would have been included in the operation.)

/WAIT 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 you use this option, the system initiates the DELETE operation but then pauses for you to mount the volume that contains the files you want to delete.

When the system pauses, it prints *Mount input volume in <device>; Continue?*, where <device> represents the device into which you mount the volume. Mount the volume and type Y or any string beginning with Y, followed by a carriage return. Type N or any string beginning with N, or two CTRL/Cs, to abort the operation and return control to the keyboard monitor. Any other response causes the message to repeat.

When the operation completes the system prints the *Continue?* message again. Mount the system volume and type a Y or any string beginning with Y, followed by a carriage return. If you type any other response the system prompts you to mount the system volume until you type Y. The system then prints the keyboard monitor prompt.

Make sure that PIP (and FILEX, if necessary) are on the system volume when you use the /WAIT option.

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

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

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

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

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

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

/ALLOCATE:size Use this option with /LIST or /OBJECT to reserve space on the device for the output file. The argument size represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 65535. A value of –1 is a special case that creates the largest file possible on the device.

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

/BUFFERING Use this option to direct the compiler to use single buffering for I/O. Normally the compiler uses double buffering.

/CROSSREFERENCE This option generates a symbol cross-reference section in the listing to which it adds as many as four separate sections to the listing. These sections are: symbol cross-reference table, label cross-reference table, external subroutine cross-reference table, and COMMON cross-reference table. Note that the system does not generate a listing by default. You must also specify /LIST in the command line to get a cross-reference listing.

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

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

/LIST[:filespec] You must specify this option to produce a DIBOL compilation listing. The /LIST option has different meanings depending on where you place it in the command line. Note that anytime you type a colon after the /LIST option (/LIST:) you must specify a device or a file specification after the colon.

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

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

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

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

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

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

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

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

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

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

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

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

The command shown above compiles A.DBL and B.DBL together, producing files DK:A.OBJ and RK3:B.LST.

If you specify a file name on a /LIST option following a file specification in the command line, it has the same meaning as when it follows the command. The following two commands have the same results:

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

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

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

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

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

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

/LOG Use this option to create a log of error messages generated by the compiler.

/OBJECT[:filespec] Use this option to specify a file name or device for the object file. Note that anytime you type a colon after the /OBJECT option (/OBJECT:) you must specify a device or a file specification after the colon.

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 KK1:.

```
.DIBOL/OBJECT:KK1: A+B
```

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

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

/NOOBJECT Use this option to suppress creation of an object file. As a command option, /NOOBJECT suppresses all object files; as a file option, it suppresses only the object file produced by the related input files.

In this command, for example, the system compiles A.DBL and B.DBL together, producing files A.OBJ and B.LST. It also compiles C.DBL and produces C.LST, but does not produce C.OBJ.

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

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

/PAGEn Use this option to override the default listing page length of 66 lines. The meaningful range of values for the decimal argument n is 1 to 32768.

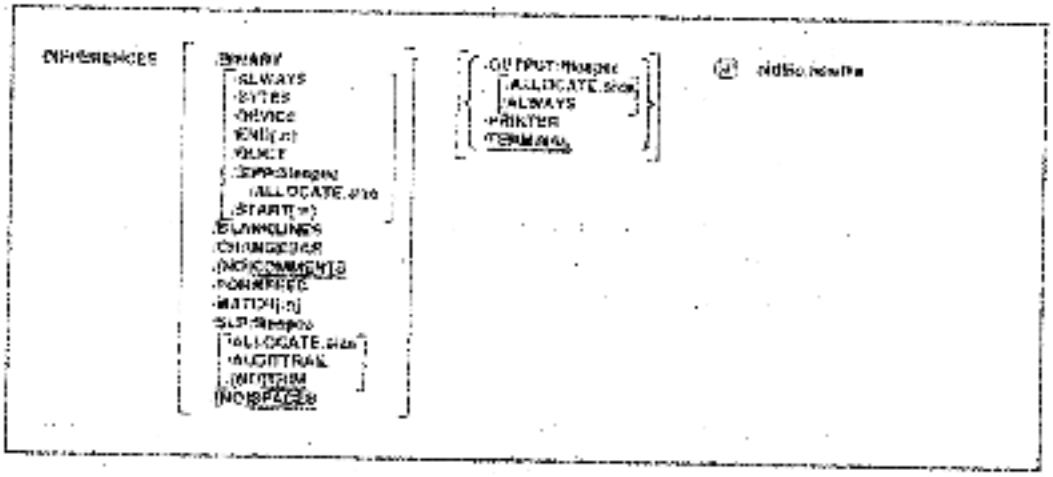
/TABLES Use this option to generate a symbol table and label table as part of the listing. This information is useful for program maintenance and debugging. Note that the system does not generate a listing by default. You must also specify /LIST in the command line to produce an assembly listing.

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

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

DIFFERENCES

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



In the command syntax shown above, oldfile represents the first file to be compared and newfile represents the second. The default output device is the console terminal. The default file type for input files is .MAC; for output listing files it is .DIF. (Default file types do not apply when you use wildcards in a DIFFERENCES command line.) You can specify the entire command on one line, or you can rely on the system to prompt you for information. The DIFFERENCES command prompts are *File 1?* and *File 2?*.

You can use wildcards in either input file specification to perform multiple source file and binary file comparisons. When you use wildcards, the system prints 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 upon 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, the system compares the file TEST1.MAC on device DY0: to all files on device DY1: with the file name TEST2:

```
*DIFFERENCES/MATCH:1/OUTPUT:TEST.DIF DY0:TEST1.MAC,DY1:TEST2.*
```

You can send the results of all the comparisons to a file on a volume rather than to the console by specifying a file name with the /OUTPUT option. In the above example, all differences from the comparisons are sent to the file TEST.DIF on device DK:.

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; each input file is compared to only 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 DY0:PROG1.*/DY1:PROG2.*
```

The system searches for the first file on DY0: with the file name PROG1, and takes note of its file type. Then, the system searches DY1: for a file with the file name PROG2 and the same file type as PROG1. If a match is found, the system compares the two files and lists the differences on the console (or sends the differences to an output file if one is specified). The system then searches DY0: for more files with the file name PROG1 and DY1: for PROG2 files with matching file types.

The DIFFERENCES command is particularly useful when you want to compare two similar versions of a source or binary program, typically an updated version against a backup version. A file comparison listing highlights the changes made to a program during an editing session.

The 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 files created by these options is .COM. You cannot use wildcards when creating SLP or SIPP command files.

The following sections describe the various options you can use with the DIFFERENCES command. Following the descriptions of the options is a sample listing and an explanation of how to interpret it.

/ALLOCATE:size Use this option 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 65535. A value of -1 is a special case that creates the largest file possible on the device.

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

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

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

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

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

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

```
Audit trail?
```

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

/BINARY When you use this option, the system compares two binary files and lists the differences between them. This option is useful for comparing 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, the system prints output at the terminal according to the following general syntax:

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

where:

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

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

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

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

/CHANGEBAR Use this option to create an output file that contains newfile with a changebar character next to the lines in newfile that differ from oldfile. The system inserts a vertical bar next to each line that has been added to newfile, and a bullet (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 line printer. Specify an output file with the /OUTPUT:filespec option.

The sample that follows 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
```

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

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

/DEVICE Use this command with /BINARY to compare two entire volumes starting with block 0. If one input volume is longer than the other the system prints *?BINCOM-W-Device is longer DEV:*. The system prints 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[:n] Use this option with /BINARY to specify the ending block number of the file comparison, where n is an octal number that represents the ending block number. If you do not supply a value with /END, the system defaults to the last block of the file or volume.

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

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

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

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

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

/SIPP:filespec Use this option with /BINARY to output a file that you can use as an input command file to the save image patch program (SIPP), where filespec represents the name of the output file. The file you create with /SIPP can patch oldfile so it matches newfile.

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

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

To execute the input command file created by /SIPP, see Chapter 22 of the *RT–11 System Utilities Manual*, Save Image Patch Program (SIPP).

/SLP[:filespec] Use this option to create a command file that, when run with the source language patch utility (SLP), patches oldfile to match newfile. The default file type is .SLP. If you do not supply a file specification with /SLP, the system prints the command file at the console.

The sample that follows creates the command file PATCH.COM. PATCH.COM 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 Chapter 23 of the *RT–11 System Utilities Manual*, Source Language Patch Program (SLP).

You cannot use wildcards in a command line with the /SLP option.

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

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

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

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

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

```
.TYPE FILE1.TXT
HERE'S A BOTTLE AND AN HONEST FRIEND!
  WHAT WAD YE WISH FOR MAIR, MAN?
WHA KENS, BEFORE HIS LIFE MAY END,
  WHAT HIS SHAME MAY BE O' CARE, MAN?
THEN CATCH THE MOMENTS AS THEY FLY,
  AND USE THEM AS YE OUGHT, MAN: --
BELIEVE ME, HAPPINESS IS SLY:
  AND COMES NOT AY WHEN SOUGHT, MAN.

                          --SCOTTISH SONG
```

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

                          --SCOTTISH SONG
```

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

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

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

The following listing shows file DIFF.TXT.

```
.TYPE DIFF.TXT
1) DK:FILE1.TXT
2) DK:FILE2.TXT
**********
1,1        WHAT HIS SHAME MAY BE O' CARE, MAN?
1)       THEN CATCH THE MOMENTS AS THEY FLY,
****
2)1        WHAT HIS SHARE MAY BE O' CARE, MAN?
2)       THEN CATCH THE MOMENTS AS THEY FLY,
**********
```

```
1)1        BELIEVE ME, HAPPINESS IS SLY,
1)       AND COMES NOT AT WHEN SOUGHT, MAN.
****
2)1        BELIEVE ME, HAPPINESS IS SHY,
2)       AND COMES NOT AT WHEN SOUGHT, MAN.
**********
```

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

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

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

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

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

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

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

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

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

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

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

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

```
%SRCCOM-I-No differences found
```

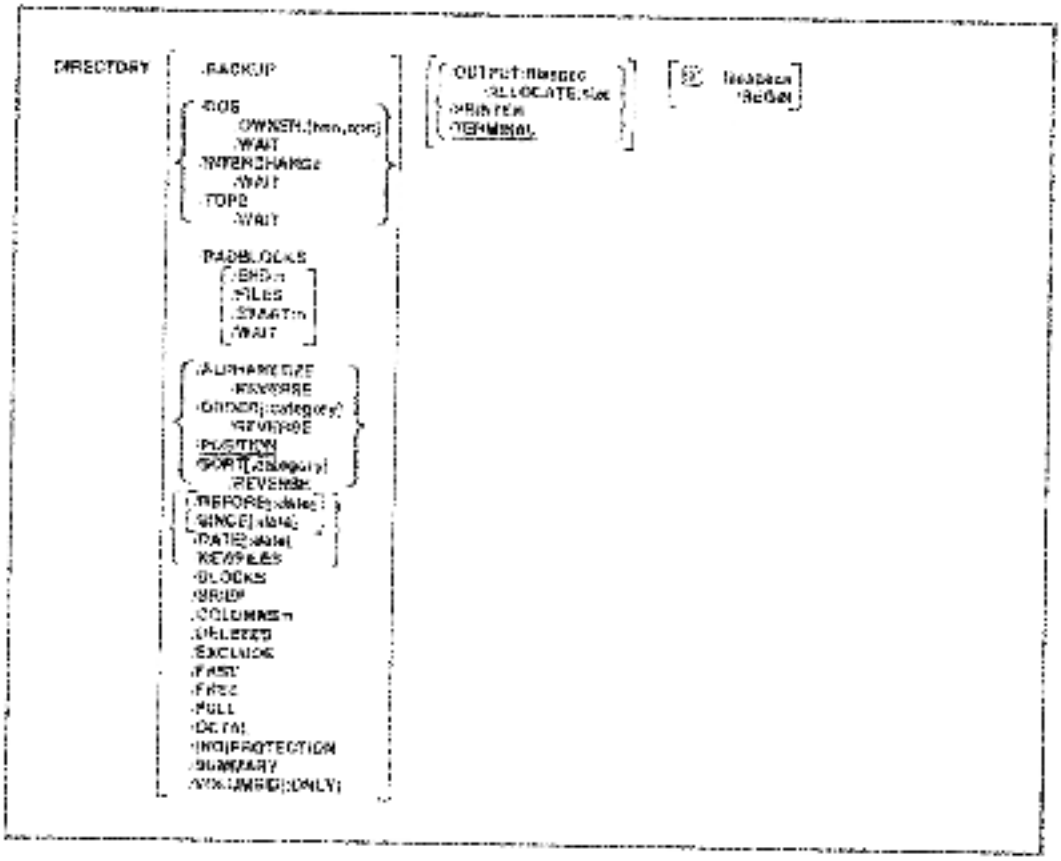
If you use wildcards in the command line, the system always prints the specifications of the files being compared regardless of whether or not there are differences.

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

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

DIRECTORY

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



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

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

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

```
>DIRECTORY
 2-Apr-83
RT11SJ.SYS     67P 03-Mar-83   RT11FB.SYS     60P 13-Feb-83
RT11XL.SYS     60P 15-Mar-83   DX     .SYS     3P 13-Feb-83
BKAF   .SYS    25P 13-Feb-83   TT     .SYS     2P 13-Apr-83
DF     .SYS     3P 15-Mar-83   DY     .SYS     4P 13-Apr-83
LP     .SYS     2P 27-Jan-83   PIP    .SAV    16  25-Mar-83
DUP    .SAV    41  28-Mar-83   RESORC.SAV    15  15-Apr-83
EDIT   .SAV   .19  13-Feb-83   STARTS.COM     1  27-Jan-83
DIFF   .SAV    14  13-Feb-83
 15 Files, 413 Blocks
 73 Free blocks
```

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

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

```
.DIRECTORY DY0:RT11FB.SYS
 10-Jan-83
RT11FB.SYS     80P 3-Jan-83
 1 File, 80 Blocks
 4 Free blocks
```

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

{dd[-mmm[-yy]

where:

dd	represents the day (a decimal integer in the range 1–31)
mmm	represents the first three characters of the name of the month
yy	represents the year (a decimal integer in the range 72–99)

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 -82 and the current system date is May 4, 1983, the system uses the date 4-MAY-82. 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.)

/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 65535. A value of –1 is a special case that creates the largest file possible on the device.

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

/BACKUP This option lists the directory of the backup volume you specify. This option lists only backup information about a volume created with the /BACKUP command.

The listing for a random-access volume begins with the system date and the volume number of the specified backup volume, which indicates its position within the set of volumes that compose a single file or volume. The volume number is followed by a four-column listing of information about each volume in the set. The first column lists the volume numbers. The second column lists the name of the file, part of which resides on that volume. The third column lists the number of blocks from the file each volume contains. The last column lists the date on which the file or volume was backed up. Underneath the four columns the system prints the number of free blocks on the specified volume. Since this directory information is determined when you first begin a backup operation, all the predetermined backup directory information prints when you use this option even if you do not complete the backup operation.

The following command lists the backup information for backup volume 3 of the four-volume set that composes the file CAFE.TXT.

```
.DIRECTORY/BACKUP DV0:
23-Jan-83

VOLUME 3 OF 4

VOLUME      FILENAME        BLOCKS      DATE

 V1         CAFE.BUP         980        23-Jan-83

 V2         CAFE.BUP         980        23-Jan-83
```

```
03          CAFIL.BUP          960          23-Jan-83

02          CAFIL.BUP          400          23-Jan-82

1 file, 960 blocks
0 free blocks
```

For magtapes, the listing appears in the same four-column format. However, only the current system date, and information for the one magtape, is displayed. The third column lists the total number of blocks used in the set of magtapes that compose the file or volume.

The next command lists the backup information for a magtape.

```
.DIRECTORY/BACKUP MT1:
23-Jan-83

VOLUME      FILENAME           BLOCKS      DATE

 V1         DL1    .BUF         20480      23-Jan-83
```

/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. The system prints the absolute block number of these blocks on the volumes that return hardware errors when the system 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 prints on the terminal.

```
.DIRECTORY/BADBLOCKS DY1:
?DUP-I-No bad blocks detected DY1:
```

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

/BEFORE[:date] This option prints a directory of files created before the date you specify. The following command lists on the terminal all files stored on device DY1: created before February 1983.

```
.DIRECTORY/BEFORE:14-FEB-83 DY1:
 14-Feb-83
MYPROG.MAC       36P  19-Nov-82  TP      .MAC   25  27-Nov-82
VTSAC .MAC        7   19-Nov-82  SYSMAC.MAC    41  19-Nov-82
RT11A.SYS         0   19-Nov-82  RT11SJ.SYS    67  19-Nov-82
T1    .SYS        2   19-Nov-82  DK     .SYS    3  19-Nov-82
BUILD .MAC      100   19-Nov-82
 9 files, 281 blocks
 190 free blocks
```

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

The following example lists the file VTMAC.MAC on device DY0: and all the files that follow it in the directory.

```
.DIRECTORY DY0:VTMAC.MAC/BEGIN
 10-Mar-83
VTMAC .MAC      15  19-Feb-83      DIP    .SAV 17  03-Feb-83
RK     .SYS      3  19-Feb-83      EDIT   .SAV 18  03-Feb-83
STARTS.COM      1  27-Feb-83      DD     .SYS  5  19-Feb-83
SRCCOM.SAV     19  18-Feb-83      DIRCOM.SAV 11  05-Jan-83
SLP    .SAV      6  19-Feb-83      DIFF   .SAV 14  05-Jun-82
 19 Files, 107 Blocks
 73 Free blocks
```

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

```
.DIRECTORY/BLOCKS DY0:
 14-Dec-82
FSM    .SYS     31P 18-Nov-82    2955    BATCH  .MAC    102P 18-Nov-82    2986
BLOCPY.MAC      8P 18-Nov-82    3088    BLINIT.MAC     75P 18-Nov-82    3096
ELTASK.MAC     15P 18-Nov-82    3111    ERRCUT.MAC     08P 18-Nov-82    3126
ERRTXT.MAC      9P 18-Nov-82    3174    SYCND  .BL      3P 18-Nov-82    3183
SYSTBL.BL       4P 18-Nov-82    3186    SYCND  .DIS     5P 18-Nov-82    3190
SYSTBL.DIS     4P 18-Nov-82    3195    SYCMD  .MD      5P 18-Nov-82    3199
PROLOD.SAV     4B  13-Mar-82    3204    CHECS  .SAV    00  17-Aug-82    3252
PETAL  .SAV    3B  11-Dec-82    3252    LAMP   .SAV    29  16-Mar-82    3329
NUMBUS.SAV     20  16-Mar-82    3357
 17 Files, 348 Blocks
 10A Free blocks
```

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

```
.DIRECTORY/BRIEF DK1:
 14-Dec-82
BMAP   .SYS     RT11SJ.SYS      RT11FB.SYS      RT11SL.SYS      TT     .SYS
DT     .SYS     DP     .SYS     DX     .SYS     DY     .SYS     RF     .SAV
RX     .SYS     DL     .SYS     DM     .SYS     DS     .SYS     DD     .SYS
LP     .SYS     LS     .SYS     DR     .SYS     MS     .SYS     MTMD   .SYS
DISMTL.COM      MTMD   .SYS     NUMBER.PAS      NLOCK  .SAV     MYPROG.MAC
PROG   .MAP     ANTONY.BAK      MSMD   .SYS     NL     .SYS     PC     .SYS
PD     .SYS     CT     .SYS     SA     .SYS     MYPROG.SAV      DDT    .SAV
 35 Files, 406 Blocks
 78 Free blocks
```

/COLUMNS:n Use this option to list a directory in a specific number of columns. The value n represents an integer in the range 1–9. Normally, the

system uses two columns for regular listings and five columns for brief listings. The following example lists the directory information for device DY1: in one column.

```
.DIRECTORY/COLUMNS=1 DY1:
 28-Jan-83
SWAP   .SYS    25P 19-Jan-83
RT11SJ.SYS    67P 19-Jan-83
RT11FB.SYS    80P 19-Jan-83
RT11SL.SYS    64P 19-Jan-83
TT     .SYS     2P 19-Jan-83
DT     .SYS     3P 19-Jan-83
DP     .SYS     3P 19-Jan-82
 7 Files, 244 Blocks
 240 Free blocks
```

/DATE[:date] Use this option to include in the directory listing only those files with a certain creation date. The following command lists all the files on device DY0: that were created on April 21, 1983.

```
.DIRECTORY/DATE:21-APR-83 DY0:
 26-Apr-83
RT11SJ.SYS     67P 21-Apr-83     RT11FB.SYS   80P 21-Apr-83
RT11SL.SYS     64P 21-Apr-83     DX     .SYS   3P 21-Apr-83
SWAP   .SYS    25P 21-Apr-83     TT     .SYS   2P 21-Apr-83
DP     .SYS     3P 21-Apr-83     DY     .SYS   4P 21-Apr-83
LP     .SYS     2P 21-Apr-83     PIP    .SAV  16  21-Apr-83
DUP    .SAV     6  21-Apr-83     RESORC.SAV  13  21-Apr-83
DIR    .SAV    17  21-Apr-83     RK     .SYS   3  21-Apr-83
FDIT   .SAV    15  21-Apr-83     DD     .SYS   5  21-Apr-83
SRCCOM.SAV     13  21-Apr-83     BINCOM.SAV  11  21-Apr-83
SLP    .SAV     9  21-Apr-83     DIFF   .SAV  14  21-Apr-83
 20 Files, 412 Blocks
 73 Free blocks
```

/DELETED This option lists a directory of files that have been deleted from a specific device, but whose file name information has not been destroyed. The listing includes the file names, types, sizes, creation dates, and starting block numbers in decimal of the files. The file names that print also represent tentative files. The listing can be useful in recovering files that have been accidentally deleted. Once you identify the file name and location, you can use DUP or the CREATE command to rename the area (see Section 6.2.1 of the *RT–11 System Utilities Manual* for this procedure).

The following command lists files on device DY0: that have been deleted.

```
.DIRECTORY/DELETED DY0:
 14-Jan-83
SYSGEN.CND      11  19-Nov-82   1403   T5     .MAC     2  27-Nov-82 2305
TM     .MAC     26  10-Nov-82   2926   M1     .SYS    32  27-Nov-82 3413
A660AT.DIR       1  14-Dec-82   3701   A50CCL.DIR    227  14-Dec-82 3704
NUM2   .MAC      4  21-Nov-82   4231   NUM2   .LST    365  05-Dec-82 4235
 0 Files, 0 Blocks
 1164 Free blocks
```

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

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

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

/EXCLUDE This option lists a directory of all the files on a device except those files you specify. The following example lists all files on DY0: except the .SAV and .SYS files.

```
.DIRECTORY/EXCLUDE DY0:(*.SAV,*.SYS)
 26-Oct-82
RT11SJ.MAC      67P 06-Sep-82     RT11FB.MAC   80P 06-Sep-82
RT11BL.MAC      63P 06-Sep-82     DX      .MAC    3P 06-Sep-82
SWAP   .MAC     25P 06-Sep-82     TT      .MAC    2P 06-Sep-82
DW      .MAC     3P 06-Sep-82     DY      .MAC    4P 06-Sep-82
LP      .MAC     2P 06-Sep-82     RK      .MAC    0  06-Sep-82
STARTS.COM       1  27-Aug-82     DD      .MAC    5  06-Sep-82
 12 Files, 250 Blocks
 73 Free blocks
```

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

/FILES Use this option with /BADBLOCKS to print the file names of bad blocks. If the system does not find any bad blocks, it prints only the heading, as the following example shows.

```
.DIRECTORY/BADBLOCKS/FILES DY1:
?DUP-I-No bad blocks detected DY1:
```

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

```
.DIRECTORY/FREE
 14-Jan-83
< UNUSED >     11          < UNUSED >      2
< UNUSED >     20          < UNUSED >     32
< UNUSED >      1          < UNUSED >    525
< UNUSED >      0          < UNUSED >    550
 0 Files, 0 Blocks
 1132 Free blocks
```

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

```
.DIRECTORY/FULL DX0:
 14-Dec-82
SWAP   .SYS       256  20-Oct-82     RT11SJ.SYS       87P 23-Oct-82
RT11FB.SYS        80P 18-Nov-82     RTLLBL.SYS       64P 18-Nov-82
F1     .SYS        2P 18-Nov-82     DT     .SYS       3P 18-Nov-82
DP     .SYS        3P 23-Dec-82     DX     .SYS       3P 18-Nov-82
DY     .SYS        4P 18-Nov-82     DF     .SYS       3P 18-Nov-82
RK     .SYS        3P 18-Nov-82     DL     .SYS       4P 23-Oct-82
DM     .SYS        5P 23-Oct-82     DS     .SYS       3P 13-Nov-82
DD     .SYS        5P 23-Oct-82     LP     .SYS       2P 23-Oct-82
LS     .SYS        2P 18-Nov-82     CR     .SYS       3P 18-Nov-82
MS     .SYS        8P 27-Nov-82     MTAC   .SYS       3P 23-Oct-82
DIGITAL.COM        4P 27-Nov-82     MMAC   .SYS       4P 18-Nov-82
NUMBER.PAS         1  11-Dec-82     TONY   .AGT      14  17-Aug-82
NUMB   .LST        1  10-Dec-82     < UNUSED >      184
 25 Files, 322 Blocks
 184 Free blocks
```

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

/NEWFILES This option 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 19 May 1983.

```
.DIRECTORY/NEWFILES DY1:
 19-May-83
FILE1  .TXT      1  18-May-83        FILE2  .TXT      1  19-May-83
 2 Files, 2 Blocks
 958 Free blocks
```

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

```
.DIRECTORY/OCTAL DX0:
 14-Dec-82 Octal
MYPROG.MAC      44P 12-Nov-82     TM     .MAC       3;  27-Nov-82
.TMP   .MAC       7  16-Dec-82     SYSMAC.MAC       5;  10-Nov-82
SWAP   .SYS     3;  05-Sep-82     ARTCM  .MAC       4  10-Nov-82
RT11SJ.SYS    103  10-Nov-82     TT     .SYS       2  10-Nov-82
DX     .SYS       2  20-Aug-82     BUILD  .MAC     144  15-Nov-82
 10 Files, 462 Blocks
 234 Free blocks
```

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

Table 4–4: DIRECTORY Sort Categories

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

The following examples list the directory of device DY0:, according to each of the categories.

```
.DIRECTORY/ORDER:DATE DY0:
 14-Dec-82
BUILD .MAC     100  06-Sep-82     SYSMAC.MAC      41  10-Nov-82
DX     .SYS      3  06-Sep-82     TT     .SYS      2  10-Nov-82
MYPROG.MAC      36P 12-Oct-82     UTMAC .MAC       7  10-Nov-82
RFUNCT.MAC       4  18-Nov-82     TM     .MAC      25  27-Nov-82
RTIISJ.SYS      67  19-Nov-82     SWAP   .SYS      25  05-Dec-82
 10 Files, 306 Blocks
 180 Free blocks


.DIRECTORY/ORDER:NAME DY0:
 14-Dec-82
BUILD .MAC     100  06-Sep-82     SWAP   .SYS      25  03-Dec-82
DX     .SYS      3  06-Sep-82     SYSMAC.MAC      41  10-Nov-82
MYPROG.MAC      36P 12-Oct-82     TM     .MAC      25  27-Nov-82
RFUNCT.SYS       4  18-Nov-82     TT     .SYS      2  10-Nov-82
RTIISJ.SYS      67  19-Nov-82     UTMAC  .MAC      7  10-Nov-82
 10 Files, 306 Blocks
 180 Free blocks


.DIRECTORY/ORDER:POSITION DY0:
 14-Dec-82
RTIISJ.SYS      67  19-Nov-82     BUILD .MAC     100  06-Sep-82
DX     .SYS      3  06-Sep-82     SYSMAC.MAC      41  10-Nov-82
MYPROG.MAC      36P 12-Oct-82     TM     .MAC      25  27-Nov-82
SWAP   .SYS      25  05-Dec-82     UTMAC .MAC       7  10-Nov-82
RFUNCT.SYS       4  18-Nov-82     TT     .SYS      2  10-Nov-82
 10 Files, 306 Blocks
 180 Free blocks
```

```
.DIRECTORY/ORDER:SIZE /TO:
 14-Dec-82
TT     .SYS       2  19-Nov-82     TM     .MAC      25  27-Nov-82
DK     .SYS       3  06-Sep-82     MYPROG.MAC      35P 12-Oct-82
RFUNCT.SYS        4  19-Nov-82     SYSMAC.MAC      41  19-Nov-82
VTMAC .MAC        7  19-Nov-82     RTIISJ.SYS      67  19-Nov-82
SWAP   .SYS      25  05-Dec-82     BUILD .MAC     100  06-Sep-82
 10 Files, 305 Blocks
 180 Free blocks


.DIRECTORY/ORDER:TYPE /TO:
 14-Dec-82
BUILD .MAC     100  06-Sep-82     DK     .SYS       3  06-Sep-82
MYPROG.MAC      35P 12-Oct-82     RFUNCT.SYS        4  19-Nov-82
SYSMAC.MAC      41  19-Nov-82     RTIISJ.SYS       67  19-Nov-82
TM     .MAC      25  27-Nov-82     SWAP   .SYS      25  05-Dec-82
VTMAC .MAC        7  19-Nov-82     TT     .SYS       2  19-Nov-82
 10 Files, 306 Blocks
 180 Free blocks
```

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

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

/POSITION Use this option to list the file sequence numbers of files stored on a magtape.

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

/PROTECTION This option 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
 14-Dec-82
BUILD .MAC     100P 06-Sep-82     TM     .MAC      25P 27-Nov-82
RTIISJ.SYS      67P 19-Nov-82     VTMAC .MAC       7P 19-Nov-82
SYSMAC.MAC      41P 19-Nov-82     RFUNCT.SYS       4P 19-Nov-82
MYPROG.MAC      35P 12-Oct-82     DK     .SYS      3P 06-Sep-82
SWAP   .SYS     25P 05-Dec-82     TT     .SYS      2P 19-Nov-82
 10 Files, 306 Blocks
 180 Free blocks
```

/NOPROTECTION This option includes in the directory listing only those files on the specified volume that are not protected against deletion.

/REVERSE This option lists a directory in the reverse order of the sort you specify with /ALPHABETIZE, /ORDER, or /SORT. The following example sorts the directory of DY0: and lists it in reverse order by size.

```
.DIRECTORY/BRIEF/SIZE/REVERSE DY0:
 14-Dec-82
BUILD .MAC      100   03-Sep-82     TM      .MAC      25   27-Nov-82
RT11SJ.SYS       67   19-Nov-82     %TMAC   .MAC       7   10-Nov-82
SYSMAC.MAC       41   15-Nov-82     RFUNCT.SYS         4   10-Nov-82
MYPROC.MAC      38P   12-Oct-82     DX      .SYS       3   08-Sep-82
SWAP   .SYS      25   05-Dec-82     TT      .SYS       2   12-May-82
 10 Files, 305 Blocks
 190 Free blocks
```

/SINCE[:date] This option 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 15, 1982.

```
.DIRECTORY/SINCE:15-AUG-82
 16-Dec-82
RT11SJ.SYS      87P  19-Aug-82     RT11FB.SYS  80P  02-Sep-82
RT11SL.SYS      83P  19-Aug-82     DX      .SYS    3P  10-Sep-82
SWAP   .SYS     25P  02-Sep-82     TT      .SYS    2P  15-Sep-82
SYPP   .SAV      1A  02-Sep-82
 7 Files, 150 Blocks
 342 Free blocks
```

/SORT[:category] This option sorts the directory of a device according to the category you specify. It is the same as /ORDER[:category]. (See Table 4–4.)

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

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

```
.DIRECTORY/SUMMARY
 14-Mar-83

   44 Files in segment 1

   46 Files in segment 6

   37 Files in segment 2

   34 Files in segment 5

   28 Files in segment 3
```

```
  18 Available segments, 5 in use

125 Files, 3047 Blocks
1115 Free blocks
```

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

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

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

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 DX1:

```
.DIRECTORY/VOLUMEID DX1:
 14-Dec-82
 Volume ID: BACKUP2
 Owner:     [ Mary ]
SWAP   .SYS      25P 18-Nov-82        RT11SJ.SYS    87P 18-Nov-82
RT11FB.SYS      80P 18-Nov-82        RT11BL.SYS    34P 18-Nov-82
ZT     .SYS       2P 18-Nov-82        DT     .SYS    2P 18-Nov-82
DP     .SYS       3P 18-Nov-82        DX     .SYS    3P 18-Nov-82
DY     .SYS       4P 18-Nov-82        RF     .SYS    2P 18-Nov-82
RK     .SYS       3P 18-Nov-82        DL     .SYS    4P 18-Nov-82
 12 Files, 271 Blocks
 215 Free blocks
```

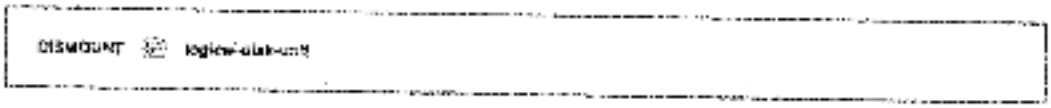
/WAIT Use with the /BADBLOCKS option when you want the system to initiate a bad block scan but to pause for you to mount the input volume. This option is particularly useful if you have a single-disk system. When you use this option, and the system volume is mounted, the system initiates the operation you specify, then prints *Mount input volume in <device>; Continue?*. The prompt <device> represents the device into which you mount the volume. Mount your input volume and type Y or any string beginning with Y, followed by a carriage return. Type N or any string beginning with N, or two CTRL/Cs, to abort the operation and return control to the keyboard monitor. Any other response causes the message to repeat. Make sure that DUP or FILEX is on the system volume when you use the /WAIT option.

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

The DISMOUNT command disassociates a logical disk unit from a file.



In the command syntax illustrated above, logical-disk-unit represents the logical disk unit that you want to free from its current assignment. Specify a logical disk unit number in the form LDn:, where n is an integer in the range 0–7. If the logical disk has been assigned a logical device name, you can free the logical disk unit by specifying the logical device name. You can specify the entire command on one line, or you can rely on the system to prompt you for information. The DISMOUNT command prompt is *Device?*.

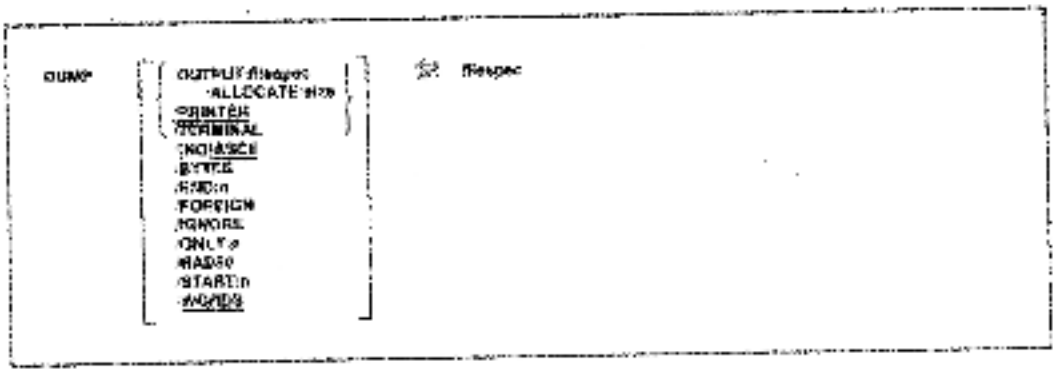
The following example frees logical disk unit 3 (LD3:) from it's current file assignment.

```
>DISMOUNT LD3:
```

The following example shows another way of freeing logical disk unit 3, once it has been assigned the logical device name DAT.

```
>ASSIGN LD3: DAT
>DISMOUNT DAT
```

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



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

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

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

NOTE

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

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

/ALLOCATEsize 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 65535. A value of –1 is a special case that creates the largest file possible on the device.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

.PRINT MACLIB.DMP

SY:SYSMAC.MAC
BLOCK NUMBER 00001
000/ 120 040 117 106 040 040 124 110 105 040 040 123 117 105 124 127
       P       O   F           T   H   E           S   O   F   T   W
020/ 101 122 105 040 040 111 123 040 040 110 105 122 105 102 131 015
       A   R   E           I   S       H   E   R   E   B   Y   .
040/ 012 073 040 126 122 101 116 125 106 105 122 122 105 104 053 015
       .   ;       V   R   A   N   S   F   E   R   R   E   D   .   .
060/ 012 073 015 012 073 040 124 110 105 040 111 116 106 117 122 115
       .   ;   .   .   ;       T   H   E       I   N   F   O   R   M
100/ 101 124 111 117 116 040 111 116 040 124 110 111 123 040 123 117
       A   T   I   O   N       I   N       T   H   I   S       S   O
120/ 106 124 127 101 122 105 040 111 123 040 120 125 102 114 105 103
       F   T   W   A   R   E       I   S       S   U   B   L   E   C
140/ 124 040 124 117 040 103 110 101 116 107 105 040 040 127 111 124
       T       T   O       C   H   A   N   G   E           W   I   T
160/ 110 117 125 124 040 040 116 117 124 111 103 105 015 012 073 040
       H   O   U   T           N   O   T   I   C   E   .   .   ;
200/ 101 116 104 040 040 123 110 117 125 114 104 040 040 116 117 124
       A   N   D           S   H   O   U   L   D       N   O   T
220/ 040 040 102 105 040 040 103 117 116 123 124 122 125 103 104 040
           B   E       C   O   N   S   T   R   U   C   E   D
240/ 040 101 123 040 040 101 040 122 117 115 115 111 124 115 105 110
       A   S       A       R   O   M   M   I   T   M   E   N
260/ 124 040 102 131 040 104 111 107 111 124 101 114 040 105 121 125
       T       B   Y       D   I   G   I   T   A   L       E   Q   U
300/ 111 120 115 105 116 124 015 012 073 040 103 117 122 120 117 122
       I   P   M   E   N   T   .   .   ;       C   O   R   P   O   R
320/ 101 124 111 117 116 056 015 012 073 015 012 073 040 104 111 107
       A   T   I   O   N   .   .   .   ;   .   .   ;       D   I   G
340/ 111 124 101 114 040 101 123 123 123 115 105 123 040 115 117 040
       I   T   A   L       A   S   S   U   M   E   S       M   O
360/ 122 105 123 120 117 116 123 111 102 111 114 111 124 131 040 103
       R   E   S   P   O   N   S   I   B   I   L   I   T   Y       F
400/ 117 122 040 124 110 105 040 125 123 105 040 117 122 040 040 122
       O   R       T   H   E       U   S   E       O   R           R
420/ 105 124 111 101 102 111 114 111 124 131 040 040 117 103 040 040
       E   L   I   A   B   I   L   I   T   Y           O   C
440/ 111 124 123 015 012 073 040 123 117 106 124 127 101 122 105 040
       I   T   S   .   .   ;       S   O   F   T   W   A   R   E
460/ 117 116 040 103 101 125 111 120 115 105 116 124 040 127 110 111
       O   N       C   A   U   I   P   M   E   N   T       W   H   I
500/ 103 110 040 101 123 040 116 117 120 040 123 125 120 120 114 111
       C   H       A   S       N   O   T       S   U   P   P   L   I
520/ 105 104 040 102 131 040 104 111 107 111 124 101 114 056 015 012
       E   D       B   Y       D   I   G   I   T   A   L   .   .   .
540/ 104 056 115 101 103 122 117 040 056 056 123 031 056 056 015 012
       .   .   M   A   C   R   O       .   .   S   .   .   .   .   .
```

```
580/ 056 115 103 101 114 114 011 056 056 056 103 115 060 054 056 056
       .   M   C   A   L   L   	   .   .   .   C   M   0   ,   .   .
600/ 056 103 115 061 054 056 056 056 103 115 062 054 056 056 056 103
       .   C   M   1   ,   .   .   .   C   M   2   ,   .   .   .   C
620/ 115 063 054 056 056 056 103 115 064 054 056 056 056 103 115 065
       M   3   ,   .   .   .   C   M   4   ,   .   .   .   C   M   5
640/ 054 056 056 056 103 115 063 015 012 050 056 033 126 051 075 061
       ,   .   .   .   C   M   3 015 012   (   .   .   V   )   =   1
660/ 015 012 056 103 116 130 115 015 012 015 012 050 115 101 103 122
     015 012   .   C   N   X   M 015 012 015 012   (   M   A   C   R
700/ 117 040 056 056 126 062 056 056 015 012 056 215 103 101 114 114
       O       .   .   V   2   .   .   .   .   .   M   C   A   L   L
720/ 011 000 056 056 103 115 000 054 056 056 056 103 115 051 054 056
       	       .   .   C   M   0   ,   .   .   .   C   M   )   ,   .
740/ 056 056 103 115 062 054 056 056 056 056 103 115 000 054 056 056 056
       .   .   C   M   2   ,   .   .   .   .   C   M   3   ,   .   .   .
760/ 103 115 064 054 056 056 056 056 103 115 065 054 056 056 056 103 115
       C   M   4   ,   .   .   .   .   C   M   5   ,   .   .   .   C   M
```

In the printout above, the heading shows which file was dumped and which block of the file follows. The numbers in the leftmost column indicate the byte offset from the beginning of the block. Remember that these are all octal values, and that there are two bytes per word. The octal bytes that were dumped appear in the next 16 columns. The ASCII equivalent of each octal byte appears underneath the byte. The system substitutes a dot (.) for nonprinting codes, such as those for control characters.

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

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

RK0:/N/X/O:6
BLOCK NUMBER 000006
000/ 000000 000002 000006 000000 000046 002000 075131 052000
       P      B      F                 &      VX     5MA      P
020/ 075273 000031 000000 027147 002000 017767 142000 075273
     SYS       Y           CPG      VX     RTI      15J     SYS
040/ 000103 000000 027147 002000 071677 141052 075273 000120
       A4           CPG      VX     RT1     IPS     SYS       6
060/ 000000 027147 002000 071577 141054 075273 000100 000000
           CPG      VX     RT1     15L     SYS       @
100/ 027147 002000 136040 000000 075273 000002 000000 027147
     CPG      VX     TT           SYS       B           CPG
120/ 002000 016040 000000 075273 000003 000000 027147 002000
       VX     07          SYS       C           CPG      VX
140/ 016000 000000 075273 000003 000000 027147 002000 016000
     DP          SYS       C           CPG      VX     DX
160/ 000000 075273 000003 000000 027147 002000 016370 000000
           SYS       C           CPG      VX     DY
200/ 075273 000004 000000 027147 002000 075350 000000 075273
     SYS       D           CPG      VX     RP          SYS
220/ 000003 000000 027147 002000 071070 000000 075273 000003
       E           CPG      VX     RA          SYS       C
240/ 000000 027147 002000 016040 000000 075273 000004 000000
           CPG      VX     DL          SYS       D
```

```
2E0/ 027147 002000 015410 000000 075273 000005 000000 027147
       GPB     YX     DM            SYS     E              GPB
300/ 002000 015770 000000 075273 000003 000000 027147 002000
       YX     DZ            SYS     C              GPB     YX
320/ 014540 000000 075273 000005 000000 027147 002000 045800
       DU            SYS     E              GPB     YX     LP
340/ 000000 075273 000002 000000 027147 002000 043770 500000
             SYS     B              GPB     YX     LS
360/ 075273 000002 000000 027147 002000 012520 000000 075273
       SYS     B              GPB     YX     CP            SYS
400/ 000003 000000 027147 002000 052070 000000 075273 000011
       C              GPB     YX     MS            SYS     I
420/ 000000 027547 002000 052150 014400 075273 000003 000000
             GWD     YX     MTH     D     SYS     C
440/ 027147 002000 015172 052177 012445 000011 000000 027547
       GPB     YX     DIS     MTI     CDM     I              GWD
460/ 002000 051520 014400 075273 000004 000000 027147 002000
       YX     MNH     D     SYS     C              GPB     YX
500/ 015175 052200 012445 000010 000000 027547 002000 052100
       DIS     MTZ     CDM     H              GWD     YX     MBH
520/ 014400 075273 000004 000000 027147 002000 054540 000000
       D     SYS     D              GPB     YX     NL
540/ 075273 000002 000000 027147 002000 052170 000000 075273
       SYS     B              GPB     YX     FC            SYS
560/ 000002 000000 027147 002000 052240 000000 075273 000003
       B              GPB     YX     FO            SYS     C
600/ 000000 027147 002000 012740 000000 075273 000005 000000
             GPB     YX     CT            SYS     E
620/ 027147 002000 005250 000000 075273 000007 000000 027147
       GPB     YX     BA            SYS     G              GPB
640/ 002000 015130 000000 073378 000051 000000 027147 002000
       YX     DUA           SAV     AR             GPB     YX
660/ 023732 050574 073375 000023 000000 027147 002000 070533
       FDR     MAT     SAV     S              GPB     YX     RES
700/ 050225 073378 000017 000000 027147 002000 075273 015172 000000
       ORD     SAV     O              GPB     YX     SYS     DIR
720/ 073378 000021 000000 027147 002000 075273 050533 074324
       SAV     J              GPB     YX     SYS     MAC     SPL
740/ 000052 000000 027147 002000 017751 075400 073378 000023
       AS             GPB     YX     EDI     I      SAV     S
760/ 000000 027147 002000 042514 000000 073378 000023 000000
             GPB     YX     KED            SAV     AE
```

E

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

```
E  address[-address2]
```

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

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

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

Note that you cannot examine addresses outside the background.

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

```
.E 1000

127401
```

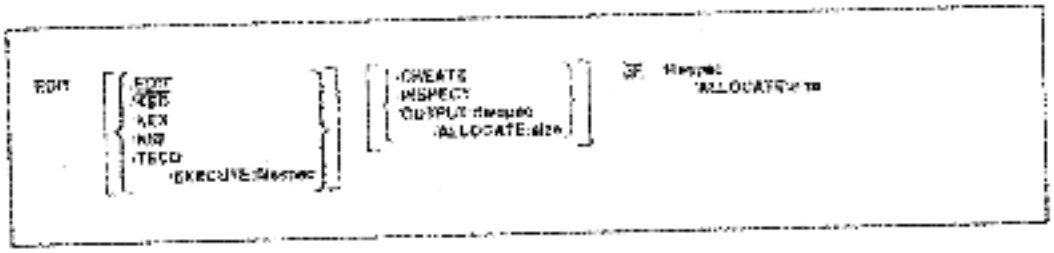
The next command sets the relocation base to 1000.

```
.B 1000
```

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

```
.E 1001-1005

127401 007824 127400
```

The EDIT command invokes the text editor.



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

You can also use the keypad editor (KED or KEX for VT100-compatible terminals, K52 for VT52 terminals) as an alternative to EDIT if you have a video terminal. You can invoke the keypad editor with the /KED, /KEX, or /K52 options described below. For more information on the keypad editor, see the *PDP–11 Keypad Editor User's Guide*.

NOTE

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

EDIT considers a file to be divided into logical units called pages. A page of text is generally 56–60 lines long (delimited by form feed characters) and corresponds approximately to a physical page of a program listing. EDIT reads one page of text at a time from the input file into its internal buffers where the page becomes available for editing. You can then use editing commands to:

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

In the command syntax illustrated above, filespec represents the file you wish to edit. You can enter the EDIT command on one line, or you can rely on the system to prompt you for information. If you do not supply a file specification for the file to edit, the system prompts *File?*. If you do not specify

any option with the EDIT command, the text editor performs the edit backup operation. To do this, it changes the name of the original file, giving it a file type of .BAK when you finish making your editing changes. The actual file renaming occurs when you successfully exit from the editor.

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

```
.EDIT MYFILE.TXT
*R33
```

When you issue an EDIT command, the system invokes the text editor. (You can use the SET EDIT command to set the default editor. If you do not use the SET EDIT command, the system assumes EDIT.SAV each time you issue the EDIT command. See the SET EDIT command for more information.)

It is possible to receive an error or warning message as a result of the EDIT command. If, for example, the file you need to edit with EDIT does not exist on device DK:, the editor issues an error message and remains in control. For example:

```
.EDIT/INSPECT EXAMPB.TXT
?EDIT-F-File not found
*^C^C[33]
```

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

NOTE

To perform any edit operations on a protected file, you must disable the file's protected status (see the descriptions of the UNPROTECT command, the COPY/NOPROTECTION command, or the RENAME/NOPROTECTION command).

The following sections describe the options you can use with the EDIT command. A complete description of EDIT is contained in Chapter 6.

/ALLOCATE:size Use this option with /OUTPUT or after the file specification to reserve space on the device for the output file. The argument size represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 65535. A value of -1 is a special case that creates the largest file possible on the device.

/CREATE Use this option to build a new file. With EDIT you can also create a new file while you are working with the text editor by using the Edit Write (EW) command, described in Chapter 6.

The following example creates a file called NEWFIL.TXT on device DK:, inserts one line of text, and then closes the file.

```
.EDIT/CREATE NEWFIL.TXT
*I
THIS IS A NEW FILE.
^Z
*EX
```

To create a file using KED, KEX, or K52, use the /KED, /KEX, or /K52 options with /CREATE. See the *PDP-11 Keypad Editor User's Guide* for more information on creating files with /KED, /KEX, or /K52.

/EDIT This option invokes the editor EDIT. This is the default editor.

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

/INSPECT Use this option to open a file for reading. This option does not create any new output files. You can also open a file for inspection while you are working with EDIT by using the Edit Read (ER) command, which is explained in Chapter 6 of this manual.

The following commands open an existing file for inspection, list its contents, and then exit.

```
.EDIT/INSPECT NEWFIL.TXT
*R
*/L
THIS IS A NEW FILE.
*EX
```

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

/KEX This option invokes a specialized version of the keypad editor (KEX). KEX is a version of KED for use only as a background job under the XM monitor. Use /KEX only if you are using a VT100-compatible terminal.

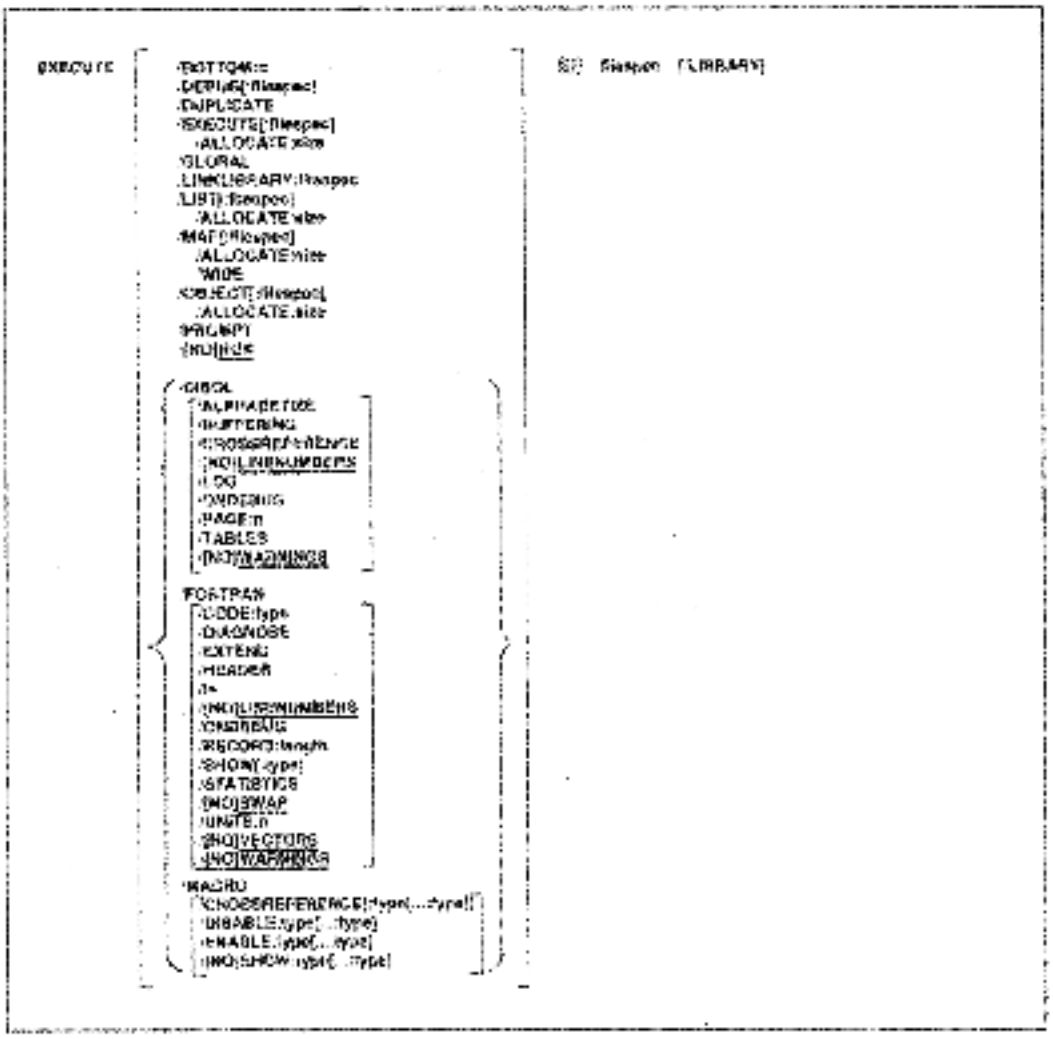
/K52 This option invokes the Keypad Editor. Use /K52 only if you are using a VT52 terminal. For more information on the Keypad Editor, see the *PDP-11 Keypad Editor User's Guide*.

/OUTPUT:filespec This option directs the text you edit to the file you specify, leaving the input file unchanged. You can also write text to an output file while you are working with EDIT by using the Edit Write (EW) command, explained in Chapter 6. The following command reads file ORIG.TXT, and writes the edited text to file CHANGE.TXT.

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

/TECO This option invokes the TECO editor. (TECO is not supported by DIGITAL.)

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



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

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

To compile multiple files in independent compilations, separate the files by commas (,) in the command line. This generates a corresponding object file for each input file. The system then links together all the object files and creates a single executable file.

You can combine up to six files for a compilation producing a single object file. You can specify the entire EXECUTE command as one line, or you can rely on the system to prompt you for information. The EXECUTE command prompt is *Files?*.

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

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

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

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

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

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

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

/ALLOCATE:*size* Use this option with /EXECUTE, /LIST, /MAP, or /OBJECT to reserve space on the device for the output file. The argument *size* represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 65535. A value of -1 is a special case that creates the largest file possible on the device.

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

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

/BUFFERING Use this option with /DIBOL to direct the compiler to use single buffering for I/O. Normally the compiler uses double buffering.

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

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

With MACRO, this option takes an optional argument. The argument type represents a one-character code that indicates which sections of the cross-reference listing the assembler should include. Table 4–11 summarizes the valid arguments and their meaning.

/DEBUG[:filespec] Use this option to link ODT (On-Line Debugging Technique, described in Chapter 20 of 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 Use this option with /FORTRAN to help analyze an internal compiler error. /DIAGNOSE expands the crash dump information to include internal compiler tables and buffers. Submit the diagnostic printout to DIGITAL with a software performance report (SPR) form. The information in the listing can help DIGITAL programmers locate the compiler error and correct it.

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

/DISABLE:type[...:type] Use this option with /MACRO to specify a .DSABL directive. Table 4–12 summarizes the arguments and their meaning. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all valid types.

/DUPLICATE Use this option to place 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, the system 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 a carriage return. Type a carriage return after the last global symbol you want to duplicate.

See Chapter 11 of the *RT–11 System Utilities Manual* for more information on duplicating library modules.

/ENABLE:type[...:type] Use this option with /MACRO to specify an .ENABL directive. Table 4–12 summarizes the arguments and their meaning. See the *PDP–11 MACRO Language Reference Manual* for a description of the directive and a list of all valid types.

/EXECUTE[:filespec] Use this option to specify a file name or device for the executable file. Note that anytime 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 when it follows the command and when it follows the file specification. The following command creates an executable file called PROG1.SAV on device DL1:.

```
.EXECUTE/EXECUTE:DL1:  PROG1,PROG2
```

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

```
.EXECUTE  SYS:ARTHZ/MYPROG/EXECUTE
```

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

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

/GLOBAL Use this option to generate 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 Chapter 11 of 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 the system 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/DL1:  MYPROG
```

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

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

/LIBRARY Use this option with /MACRO to identify the file the option qualifies as a macro library file. Use it only after a library file specification in the command line.

The MACRO assembler looks first to the library associated with the most recent /LIBRARY option to satisfy references (made with the .MCALL directive) from MACRO programs. It then looks to any libraries you specified earlier in the command line, and it looks last to SYSMAC.SML.

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

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

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

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

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

/LIST[:filespec] You must specify this option to produce a compilation or assembly listing. Note that anytime you type a colon after the /LIST option (/LIST:) you must specify a device or a file specification after the colon.

The /LIST option has different meanings depending on where you put it in the command line. If you specify /LIST without filespec in the list of options that immediately follows the EXECUTE command, the system generates a listing that prints on the line printer. If you follow /LIST with a device name, the system creates a listing file on that device. If the device is a file-structured device, the system stores the listing file on that device, assigning it the same name as the input file with a .LST file type.

The following command produces a listing on the terminal:

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

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

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

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

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

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

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

The command shown above compiles A.DBL and B.DBL together, producing files DK:A.OBJ and RK3:B.LST. It then links A.OBJ (using SYSLIB.OBJ as needed) and produces DK:A.SAV

If you specify a file name on a /LIST option following a file specification in the command line, it has the same meaning as when it follows the command. The following two commands have the same results:

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

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

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

```
.EXECUTE/NORUN A.MAC/LIST:S.TXT+B
```

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

/LOG Use this option with /DIBOL to create a log of error messages generated by the compiler.

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

/MAP[:filespec] You must specify this option to produce a load map after a link operation. The /MAP option has different meanings depending on where you put it in the command line. It follows the same general rules outlined above for /LIST.

/OBJECT[:filespec] Use this option to specify a file name or device for the object file. Note that anytime you type a colon after the /OBJECT option (/OBJECT:) you must specify a device or a file specification after the colon.

Because the EXECUTE command creates object files by default, the following two commands have the same meaning:

```
.EXECUTE/FORTRAN A

.EXECUTE/FORTRAN/OBJECT A
```

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

/PAGEn Use this option with /DIBOL to override the default listing page length of 66 lines. The meaningful range of values for the decimal argument n is 1 to 32768.

/PROMPT Use this option to enter additional lines of input for the link operation. The system continues to accept lines of linker input until you enter two slashes (//). Chapter 11 of 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 Chapter 1, 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:

```
.EXECUTE/PROMPT COSINE
*TAN/O:1
*COS1/O:1
*SIN3/O:2
*LOG3/O:2//
```

The /PROMPT option also gives you a convenient way to create an overlaid program from an indirect file. The file LCP.COM contains these lines:

```
@/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.

```
.EXECUTE/MAP  @LOP
```

/RECORD:length Use this option with /FORTRAN to override the default record length of 132 characters for ASCII sequentially formatted input and output. The meaningful range for length is from 4 to 4096.

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

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

/SHOW[:type] Use this option with /FORTRAN to control the FORTRAN listing format. The argument type represents a code that indicates which listings the compiler is to produce. Table 4–6 summarizes the codes and their meaning.

Use this option with /MACRO to specify any MACRO .LIST directive. Table 4–13 summarizes the valid arguments and their meaning. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives.

/NOSHOW:type Use this option with /MACRO to specify any MACRO .NLIST directive. Table 4–13 summarizes the valid arguments and their meaning. The *PDP–11 MACRO Language Reference Manual* explains how to use these directives.

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

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

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

/TABLES Use this option with /DIBOL to generate 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.

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

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

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

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

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

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

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

In the command syntax described above, device represents the storage volume you wish to format and/or verify.

You can FORMAT and /VERIFY the following devices:

RK05	RP02/RP03
RK06/RK07	RX02

You can only FORMAT the following devices:

RD50/RD51

You can only /VERIFY the following devices:

RL01/RL02	TU56 (DECtape I)
RX01	TU58 (DECtape II)

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

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

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

When you issue the FORMAT command, the system prints:

```
Device /FORMAT-Are you sure?
```

The variable <dev:> represents the drive name and unit number of the volume you want to format. Type Y to continue the format operation. Type N or any string beginning with N, or CTRL/C, to abort the operation. Any other response causes the prompt to repeat.

NOTE

You can format a diskette (RX01 or RX02) only when you have mounted the diskette in a double-density diskette drive unit (RX02). Unless you use the /SINGLEDENSITY option, the system will format diskettes in double-density format. If you attempt to format a diskette in a single-density drive unit (RX01), the system will print an error message.

When you format an RK05 or RK07 disk, the system lists the block numbers of all the bad blocks in the manufacturer's bad block table and in the software bad block table.

If you try to format a volume while a foreground job is loaded the system prints:

```
Foreground Loaded!
<dev:>/FORMAT-Are you sure?
```

Type Y or any string beginning with Y to continue with the formatting operation. Type N or any string beginning with N, or CTRL/C, to abort the operation. Any other response causes the message to repeat.

NOTE

Although you can format or verify a volume while a foreground job is loaded, it is not recommended. If you try to format or verify a volume that the foreground job is using, data on the volume will be written over and corrupted, which may cause the foreground job or the system to crash.

If you try to format a volume that contains protected files, the system prints:

```
Volume contains protected files! Are you sure?
```

Type Y or any string beginning with Y to continue the formatting operation. Type N or any string beginning with N, or CTRL/C, to abort the operation. Any other response causes the message to repeat.

The options you can use with the FORMAT command follow.

/PATTERN[:value] Use this option with /VERIFY[:ONLY] to specify which 16-bit patterns you want the system to use when it verifies the volume. The optional argument value represents an octal integer in the range 0 to 177777 that denotes which patterns you want used.

Table 4–5 lists the verification patterns FORMAT uses and the corresponding values for the argument value.

Table 4–5: 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	031042
8	7	200	104210
9	8	400	155555
10	9	1000	145454
11	10	2000	146314
12	11	4000	*
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. Table 4–5 gives the equivalent values for each verification bit pattern. If you want 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, the system runs only pattern 8.

After it completes verification, the system prints at the terminal each bad block it found during each verification pass. The format of the verification report is:

```
PATTERN #x
nnnnnn
```

In the example above, x represents the pattern number, and nnnnnn represents the bad block number. The system 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, the system uses pattern 8.

/QUERY Use this option when you want the system to request confirmation before it performs formatting or verification. You must respond to the query message by typing a Y for any string that begins with Y) and a carriage return to continue the operation. The system interprets any other response to mean NO, and it does not continue the operation. /QUERY is the default setting.

/NOQUERY Use this option if you do not want the system to print a confirmation message before it performs formatting or verification. When you use this option in the FORMAT command line, the system prints only the pattern numbers it uses if it performs verification and the informational messages indicating the formatting or verification is complete. The default setting is /QUERY.

/SINGLEDENSITY Use this option to format an RX02 double-density diskette in single-density format. The following example uses the /SINGLEDENSITY option to format 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] Use this option when you want to verify a volume following formatting. Use the optional argument, :ONLY, when you want the system to only verify a volume.

When you use /VERIFY, the system first formats the specified volume, and then writes a bit pattern to each block on the volume. Next, the system reads each pattern. After the verification process is complete, the system prints at the terminal the block number of each bad block it found.

The example that follows uses /VERIFY to format and verify an RL02 disk in drive unit 2.

```
.FORMAT/VERIFY DL2:
DL2:/FORMAT-Are you sure? Y
?FORMAT-I-Formatting complete
PATTERN #8
?FORMAT-I-Verification complete
```

The next example uses /VERIFY:ONLY to only verify an RX02 diskette in drive unit 0.

```
.FORMAT/VERIFY:ONLY DY0:
DY0:/VERIFY-Are you sure? Y
PATTERN #8
?FORMAT-I-Verification complete
```

/WAIT Use this option to initiate the formatting operation, then pause before formatting begins to wait for you to change volumes. The /WAIT option is useful for single drive systems.

After the system accepts your command line, it pauses and prints the message *Continue?*. At this time, you can exchange volumes. When the new disk is loaded, type Y or any string beginning with Y, followed by a carriage return to resume the operation. If you type N or any string beginning with N, or CTRL/C, the operation is not performed and control returns to the keyboard monitor. Any other response causes the message to repeat.

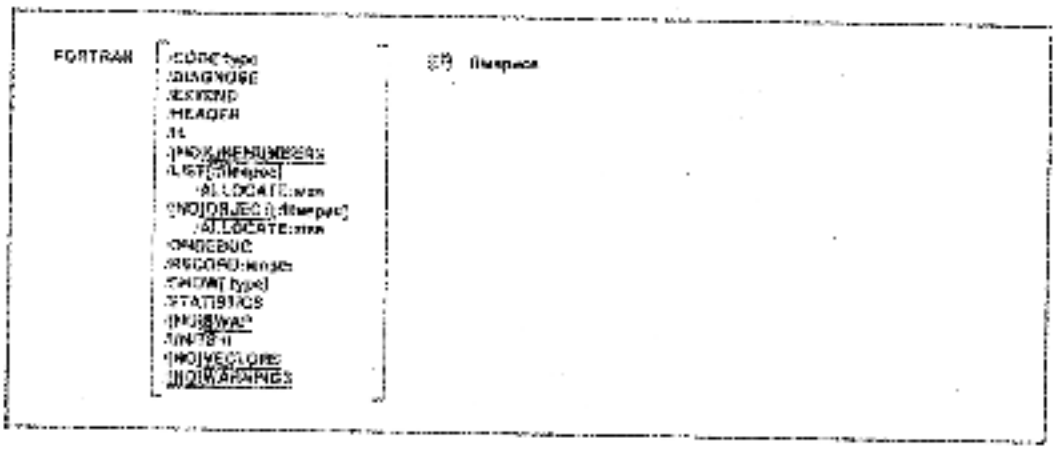
When formatting completes the system pauses again while you remount the system volume. Mount the system volume and type Y or any string beginning with Y, followed by a carriage return, to terminate the formatting operation. If you type any other response the system prompts you to mount the system volume until you type Y. The system then prints the keyboard monitor prompt. Make sure FORMAT is on the system volume when you use the /WAIT option.

The following example uses the /WAIT option to format an RL02 disk.

```
.FORMAT/WAIT DL0:
DL0:/FORMAT-Are you sure? Y
Mount input volume in <device>? Continue? Y
?FORMAT-I-Formatting complete
Mount system volume in <device>? Continue? Y
```

FORTRAN

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



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

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

Language options are position-dependent -- that is, they have different meanings depending on where you place them in the command line. Options that 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 in the command string.

The *RT-11/RSTS/E FORTRAN IV User's Guide* contains detailed information about using FORTRAN. The following sections describe the options you can use with the FORTRAN command.

/ALLOCATE:size Use this option with /LIST or /OBJECT to reserve space on a device for the output file. The argument size represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 65535. A value of –1 is a special case that creates the largest file possible on the device.

/CODE:type Use this option to produce object code that is designed for a particular hardware configuration. The argument type represents a three-letter abbreviation for the type of code to be produced. The 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.

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

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

/HEADER This option includes in the printout a list of options that are currently in effect.

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

/LINENUMBERS Use this option to include internal sequence numbers in the executable program. These are especially useful in debugging a FORTRAN program. They identify the FORTRAN statements that cause run-time diagnostic error messages. This is the default operation.

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

/LIST[:filespec] You must specify this option to produce a FORTRAN compilation listing. Anytime you type a colon after the /LIST option (/LIST:) you must specify a device or a file specification after the colon.

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 FORTRAN command. For example, the following command line produces a line printer listing after compiling a FORTRAN source file:

```
.FORTRAN/LIST MYPROG
```

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

```
.FORTRAN MYPROG/LIST
```

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

```
.FORTRAN/LIST:TT: A
```

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

```
.FORTRAN/LIST:RK3: A
```

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

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

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

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

The above command compiles A.FOR and B.FOR together, producing files DK:A.OBJ and RK3:B.LST.

If you specify a file name on a /LIST option following a file specification in the command line, it has the same meaning as when it follows the command. The following two commands have the same results:

```
.FORTRAN A/LIST:B
```

```
.FORTRAN/LIST:B A
```

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

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

```
.FORTRAN A/LIST+B
```

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

/OBJECT:filespec Use this option to specify a file name or device for the object file. Note that anytime you type a colon after the /OBJECT option (/OBJECT:) you must specify a device or a file specification after the colon.

Because FORTRAN creates object files by default, the following two commands have the same meaning:

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

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

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

```
.FORTRAN/OBJECT:RK1: A+B
```

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

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

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

```
.FORTRAN A+B/LIST:C/NOOBJECT/LIST
```

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

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

/SHOW[:type] Use this option to control FORTRAN listing output. The argument type represents a code that indicates which listings the compiler is to produce. Table 4–6 summarizes the codes and their meaning.

You can combine options by specifying the sum of their numeric codes. For example:

```
/SHOW:7
```

or

```
/SHOW:ALL
```

The two options shown above have the same meaning. If you specify no code, the default value is 3, a combination of SRC and MAP.

Table 4–6: FORTRAN Listing Codes

Code	Listing Content
0	Diagnostics only
1 or SRC	Source program and diagnostics
2 or MAP	Storage map and diagnostics
3	Diagnostics, source program, and storage map
4 or COD	Generated code and diagnostics
7 or ALL	Diagnostics, source program, storage map, and generated code

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

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

/NOSWAP This option keeps the USR resident during execution of a FORTRAN program. This may be necessary if the FORTRAN program uses some of the RT–11 system subroutine library calls (see Chapter 4 of the *RT–11 Programmer's Reference Manual*). If the program frequently updates or creates a large number of different files, making the USR resident can improve program execution. However, the cost for making the USR resident is 2K words of memory.

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

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

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

/WARNINGS Use this option to include warning messages in FORTRAN compiler diagnostic error messages. These messages call certain conditions to your attention, but do not interfere with the compilation. A warning message prints, for example, if you change an index within a DO loop, or if you specify a variable name longer than six characters.

/NOWARNINGS Use this option to exclude warning messages in FORTRAN compiler diagnostic error messages. This is the default setting.

FRUN

The FRUN command initiates foreground jobs. The default file type is .REL; the default device is DK:.

In the command syntax illustrated above, filespec represents the program to execute. Because this command runs a foreground job, it is valid for the FB and XM monitors only.

If a foreground job is active when you issue the FRUN command, an error message prints on the terminal. You can run only one foreground job at a time. If a terminated foreground job is occupying memory, the system reclaims that region for your program. Then, if the system finds your program and if your program fits in the available memory, execution begins.

Note that you can use the FRUN command to run a virtual foreground job, and that you can use FRUN to run a virtual .SAV file in the foreground under the XM monitor.

The following sections describe the options you can use with FRUN. Note that the option must follow the file specification in the command line.

/BUFFER:n Use this option to reserve more space in memory than the actual program size. The argument n represents, in octal, the number of words of memory to allocate. You must use this option to execute a FORTRAN foreground job. If you use /BUFFER for a virtual job linked with the /V option (or /XM), the system ignores /BUFFER because it has already provided a buffer in extended memory.

The following formula determines the space needed to run a FORTRAN program as a foreground job.

$$n = [1\text{'}2[504 + (35*N) + (R-136) + A*512]]$$

where:

- A represents the maximum number of files open at one time. Each file opened as double buffered should be counted as two files.
- N represents the maximum number (decimal) of simultaneously open channels (logical unit numbers). This value is specified when the compiler is built, and can be overridden with the /UNITS option during main program compilation; the default value is 6. Make sure you use a decimal point with this number.

R represents the maximum formatted sequential record length. This value is specified when the compiler is built and can be overridden with the /RECORD option during main program compilation; the default value is 136.

To specify a decimal number for n, include a decimal point (.) after the number you specify. The formula for determining n calculates n as a decimal number, so include the decimal point when you specify the number.

This formula must be modified for certain system subroutine library (SYSLIB) functions.

The IQSET function requires the formula to include additional space for queue elements (qcount) as follows:

n = [1/2[504+(35*N)+(R-136)+A*512]]+[10*qcount]

The ICDFN function requires the formula to include additional space for the integer number of channels (num) as follows:

n = [1/2[504+(35*N)+(R-136)+A*512]]+[6*num]

The INTSET function requires the formula to include additional space for the number of INTSET calls issued in the program as follows:

n = [1/2[504+(35*N)+(R-136)+A*512]]+[25*INTSET]

Any functions, including INTSET, that invoke completion routines must include 64(decimal) words plus the number of words needed to allocate the second record buffer (default is 68 decimal words).

The length of the record buffer is controlled by the /RECORD option to the FORTRAN compiler. If the /RECORD option is not used, the allocation in the formula must be 136(decimal) bytes, or the length that was set at FORTRAN installation time. This modifies the formula as follows:

n = [1/2[504+(35*N)+(R-136)+A*512]]+[64+R/2]

If the /BUFFER option does not allocate enough space in the foreground on the initial call to a completion routine, the following message appears:

```
PERR 0, NON-FORTRAN error call
```

This message also appears if there is not enough free memory for the background job or if a completion routine in the SJ monitor is activated during another completion routine. In the latter case, the job aborts; you should use the FB monitor to run multiple active completion routines.

/NAME:name Use this option to assign a logical name to the foreground job. This option is valid only on a monitor that has system job support, a special feature enabled by the system generation process.

/PAUSE Use this option to help you debug a program. When you type the carriage return at the end of the command string, the system prints the load address of your program and waits. You can examine or modify the program (by using ODT, described in Chapter 20 of 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, prints the load address, and waits for a RESUME command to begin execution.

```
.FRUN DEMOSP/PAUSE
Loaded at 027270
.RESUME
```

/TERMINAL:n This option is meaningful only in a multiterminal system. Use it to assign a terminal to interact with the foreground job. The argument n represents a terminal logical unit number. If you do not use this option, the foreground job shares the console terminal with the background job. By assigning a different terminal to interact with the foreground job, you eliminate the need for the foreground and background jobs to share the console terminal.

Note that the original console terminal still interacts with the background job and with the keyboard monitor, unless you use the SET TT:CONSOL command to change this.

The GET command loads a memory image file into memory.

```
GET /F filespec
```

In the command syntax shown above, filespec represents the memory image file to be loaded. The default file type is .SAV. Note that magtape is not a block-replaceable device and therefore is not permitted with the GET command. Use the GET command for a background job only. You cannot use GET on a virtual program that executes under the XM monitor.

The GET command is useful when you need to modify or debug a program. You can use GET with the Base, Deposit, Examine, and START commands to test changes. Use the SAVE command to make these changes permanent. You can combine programs by issuing multiple GET commands, as the following example shows. This example loads a program, DEMOSP.SAV, loads ODT.SAV (on-line debugging technique, described in Chapter 20 of the *RT-11 System Utilities Manual*), and starts the program using the address of ODT's entry point.

```
.GET DEMOSP

.GET ODT

.START

ODT  005000
*
```

If more than one program requires the same locations in memory, the program you load later overlays the previous program. Note that you cannot use GET to load overlay segments of a program; it can load only the root. If the file you need to load resides on a device other than the system device, the system automatically loads that device handler into memory when you issue the GET command. This prevents problems that occur if you use the START command and your program is overlaid.

GT

The GT command enables or disables the VT11 or VS60 graphics display hardware.

When you issue the GT OFF command, you disable the display hardware. The printing console terminal then becomes the device that prints output from the system.

When you issue the GT ON command, the display screen replaces the printing console terminal. The display screen offers some advantages over the printing terminal: it is quieter than a printing terminal, it is faster than a printing terminal, it does not require a supply of paper, and it is the device for which EDIT's immediate mode is intended.

The display screen can speed up the editing process (see Chapter 6 for information on how to use the text editor). You can use CTRL/A, CTRL/S, CTRL/E, and CTRL/Q to control scrolling. These commands are explained in Chapter 3.

Note that RT–11 does not permit you to use display hardware (with GT ON) if you have multiterminal support (enabled by a user-generated monitor) or if you have an 8K configuration. You cannot use GT ON or GT OFF when a foreground or system job is active; this causes the system to print an error message. Issue the GT ON command before you begin execution of the foreground job.

ODT (on-line debugging technique, described in Chapter 20 of the *RT–11 System Utilities Manual*) is the only system program that cannot use the display screen. Its output always appears on the console terminal. You can use VDT, a variant of ODT, because it can interact with the display hardware.

NOTE

If an indirect command file issues a GT ON command, part of the command may echo on the terminal and the rest may echo on the graphics screen. Also, if you type the GT ON command, followed by CTRL/Z, the initial line on the terminal overprints when you type GT OFF.

The following options control the number of lines that appear on the screen and position the first line vertically.

/L:n Use this option to change the number of lines of text that display on the screen. Table 4–7 shows the valid range for the argument n in decimal. If you do not use this option the system determines the screen size and automatically assigns the largest valid value.

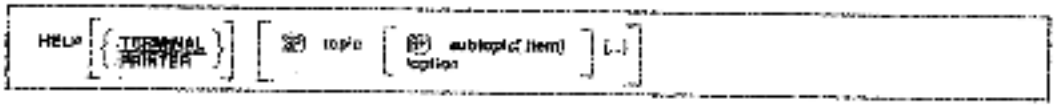
/T:n Use this option to change the top position of the scroll display. Table 4–7 shows the valid range for the argument n in decimal. If you do not use this option, the system determines the screen size and automatically assigns the largest valid value.

Table 4–7: Display Screen Values

Screen Size	Lines	Top Position
12-inch	1–31	1–744
17-inch	1–40	1–1000 (or larger)

HELP

The HELP command lists information related to RT–11 commands to help you remember command syntax, options, and so on, when you are at the console.



In the command syntax shown above, topic represents a subject about which you need information. In the help file supplied with RT–11, the topics are the keyboard monitor commands. The subtopic represents a category within a topic. In the RT–11 help file, the subtopics are SYNTAX, SEMANTICS, OPTIONS, and EXAMPLES. The item represents one member of the subtopic group. You can specify more than one item in the command line if you separate the items by colons (:). If you type HELP followed by a carriage return, the system lists information on the HELP command.

The HELP command permits you to access the HELP text file. The help file distributed with RT–11 contains information about the keyboard monitor commands and how to use them. However, the concept of the help file is a general one. That is, you can create your own help file to supply quick reference material on any subject. For information on how to change the HELP text file, see the *RT–11 Installation Guide*.

There are only two options you can use with the HELP command. They are /PRINTER and /TERMINAL.

/PRINTER Use this option to list information on the line printer.

/TERMINAL This option lists information on the console terminal. This is the default operation. 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 a carriage return to view the next screen.

The following examples all make use of the standard RT–11 help file.

The following command lists all the topics for which assistance is available.

```
.HELP *

ABORT     Terminates a foreground/System job from the console
ASSIGN    Associates a logical device name with a physical device
B         Sets a relocation base
BACKUP    Backs up/Restores large files or random access devices
```

The next command lists all the information about the DATE command.

```
.HELP DATE

DATE      Sets or displays the current system date
  SYNTAX
          DATE [dd-mmm-yy]

  SEMANTICS
          All numeric values are decimal; mmm represents the first
          three characters of the name of the month.  Under RSTS-11,
          the current date can not be changed.

  OPTIONS
          None

  EXAMPLES
          DATE 12-MAR-82
```

The next command lists all the options that are valid with the DIRECTORY command.

```
.HELP DIRECTORY OPTIONS

  OPTIONS
    ALLOCATE:size
        Use with /OUTPUT to reserve space for the output listing file
    ALPHABETIZE
        Sorts the directory in alphabetical order by file name and
        type
            .
            .
            .
```

The next command lists information about the /BRIEF option for the DIRECTORY command.

```
.HELP DIRECTORY OPTIONS/BRIEF

    BRIEF
        Lists only file names and file types of files; same as /FAST
```

The following command lists information about the DIRECTORY command options that begin with B.

```
.HELP DIRECTORY/B

    BADBLOCKS

        Scans the device for bad blocks and types their octal number
    BEFORE[: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
```

INITIALIZE

Use the INITIALIZE command to clear and initialize a device directory.

In the command syntax illustrated above, device represents the volume you need to initialize. The initialize operation must always be the first operation you perform on a new volume after you receive it, formatted, from the manufacturer. If the volume is not formatted, use the FORMAT command (see the FORMAT command description) to format the volume. After you use the INITIALIZE command, there are no files in the directory. If you use the INITIALIZE command with no options, the system simply initializes the device directory. You can enter the INITIALIZE command as one line, or you can rely on the system to prompt you for the name of the device with *Device?*.

The default number of directory segments for RT-11 directory structured volumes is listed in Table 4–8. If any default is too small for your needs, see the *RT-11 Installation Guide* for details on changing this default directory size.

If the volume you are initializing has protected files, the system always requests confirmation as in the following example.

```
.IN:? DL0:
DL0:/Initialize? Are you sure? Y
Volume contains protected files? Are you sure? Y
```

The following sections describe the options you can use with INITIALIZE and give some examples of their use.

/BACKUP Use this option to initialize a backup volume to be used as an output volume with the BACKUP command. Output volumes for the BACKUP command, except magtape, must be initialized by using this option. (The system automatically initializes magtapes during BACKUP operations; therefore, the /BACKUP option is invalid with magtape.)

Since backup output volumes cannot contain badblocks, this option also performs a bad block scan. If bad blocks are detected, the system instructs you to use another volume for your backup operation.

The system also warns you if you attempt to initialize a volume that already contains files, and allows you to replace the volume.

The following command initializes a double-density diskette as a backup volume.

```
$INITIALIZE/BACKUP DY0:
DY0:/BUP Initialize?  Are you sure? Y
%BUP-I-Bad block scan started...
%BUP-I-No bad blocks detected
```

/BADBLOCKS[:RET] Use this option to scan a volume (disk or DECtape) for bad blocks and write .BAD files over them. For each bad block the system encounters on the volume, it creates a file called FILE.BAD to cover it. After the volume is initialized and the scan completed, the directory consists of only FILE.BAD entries to cover the bad blocks. This procedure ensures that the system will not attempt to access these bad blocks during routine operations. If the system finds a bad block in either the boot block or the volume directory, it prints an error message and the volume is not usable.

DIGITAL recommends that you use the DIRECTORY/BADBLOCKS command after using the INITIALIZE/BADBLOCKS command so that you can find out where the bad blocks are, if any.

The following command initializes volume DL1: and scans for bad blocks.

```
$INITIALIZE/BADBLOCKS DL1:
DL0:Initialize? Are your sure? Y
```

If you use /BADBLOCKS:RET, the system will retain through initialization all files with a .BAD file type that it finds on the volume, giving them the name FILE.BAD. The system does not do a bad block scan. The advantage in using :RET is that initializing takes less time.

Note that some volumes support bad block replacement; DIGITAL recommends you use the /REPLACE[:RET] option instead of /BADBLOCKS[:RET] for these volumes when scanning for bad blocks. If you use INITIALIZE/BADBLOCKS with a volume that has been previously initialized with the INITIALIZE/REPLACE command, .BAD files will be written over all bad blocks and the bad block replacement table will be ignored by the system.

If the volume being initialized contains bad blocks, the system prints the locations of the bad blocks in octal and in decimal, as in the following example:

```
$INITIALIZE/BADBLOCKS DL0:
DL0:/Initialize?  Are you sure?  Y
     Block            Type
000320      80.   Hard
000471     313.   Hard
000521     337.   Hard
%BUP-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 Use this option to initialize a DECtape for DOS–11 format.

/FILE:filespec Use this option to initialize a magtape and create a bootable tape. For filespec, substitute dev:MBOOT.BOT. This file is distributed with RT–11 for this purpose only. Consult the *RT–11 Installation Guide* for more information.

The following example creates a bootable magtape.

```
.INITIALIZE/FILE:MBOOT.BOT  MT0:
```

/INTERCHANGE Use this option to initialize a diskette for interchange format. The following example initializes DX1: in interchange format.

```
.INITIALIZE/INTERCHANGE DX1:
DX1:/Init  Are you sure? Y
```

NOTE

The directory of an initialized interchange diskette has a single file entry, DATA, that reserves the entire diskette. You must delete this file before you can write any new files on the diskette. Do this by using the following command:

```
DELETE/INTERCHANGE DX1:DATA
```

This is necessary for IBM compatibility.

/QUERY This option requests confirmation before it initializes a device. Respond by typing Y or any string beginning with Y, followed by a carriage return, to initiate execution of the command. The system interprets a response beginning with any other character to mean NO. /QUERY is the default operation.

/NOQUERY Use this option to suppress the confirmation message the system prints before it proceeds with the initialization.

/REPLACE[:RET] If you have an RK06, RK07, RL01, or RL02 disk, use this option to scan a disk for bad blocks. If the system finds any bad blocks, it creates a replacement table so that routine operations access good blocks instead of bad ones. Thus, the disk appears to have only good blocks. Note, though, that accessing this replacement table slows response time for routine input and output operations.

If you use :RET with /REPLACE, the system initializes the volume and retains the bad block replacement table (and FILE.BAD files) created by the previous /REPLACE command.

Note that if the monitor file resides on a block that contains a bad sector error (BSE) and you are doing bad block replacement, a boot error results when you attempt to bootstrap the system. In this case, move the monitor. Use the DIRECTORY/BADBLOCKS/FILES command to determine which files reside on bad blocks.

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, the system can replace only those bad blocks that generate a bad sector error (BSE). With an RL01 or RL02 disk, the system 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, the system prints a list of replaceable bad blocks as in the following sample:

```
.INITIALIZE/REPLACE DL0:
    Block          Type
030722  12754.  Replaceable
115046  39426.  Replaceable
133617  46991.  Replaceable
136178  48250.  Replaceable
136277  48313.  Replaceable
136401  48383.  Replaceable
140405  49613.  Replaceable
146254  52984.  Replaceable
DUP-I-Bad blocks detected 8.
```

If there is a replacement table overflow, the system prompts you to indicate which blocks you want replaced as follows:

```
?DUP-W-Replacement table overflow DEV:
Type <RET>, 0, or nnnnnn (<RET>)
Replace block #
```

The variable nnnnnn represents the octal number of the block you want the system to replace.

After you enter a block number, the system responds by repeating the *Replace block #* prompt. If you type a 0 at any time you do not want any more blocks replaced, prompting ends and any blocks not placed in the replacement table are marked as FILE.BAD.

If you enter a carriage return at any time, the system places all bad blocks you have not entered into the replacement table, starting with the first on the disk, until the table is full. The system assigns the name FILE.BAD to any remaining bad blocks and prompting ends.

If you use /NOQUERY with /REPLACE, and there is a replacement table overflow, the effect will be as if you had entered a carriage return in response to the first *Replace block #* prompt.

/RESTORE Use this option to uninitialize a volume. That is, you can use this option to restore the directory and files that were present on the volume prior to the previous initialization. You can use /RESTORE only if no files have been transferred to the volume since the last time it was initialized. 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:n Use this option if you need to initialize a disk and also change the number of directory segments. The number of segments in the directory establishes the number of files that can be stored on a device. The system allows a maximum of 72 files per directory segment, and 31 directory segments per device. The argument n represents the number of directory segments. The valid range for n is from 1 to 31 (decimal). Table 4–8 shows the default values of n for standard RT–11 devices.

Table 4–8: Default Directory Sizes

Device	Number (decimal) of Segments in Directory
DD	1
DW (RD50)	16
DW (RD51)	31
DX	1
DY (single-density)	1
DY (double-density)	4
DZ (RX50)	4
PD	1
DL (RL01)	15
DL (RL02)	31
DM	31
DU (Winchester disks)	31
DU (diskettes)	1
RK	28

/VOLUMEID[:ONLY] Use /VOLUMEID to write a volume identification on a device when you initialize it. This identification consists of a volume ID (up to 12 characters long for a block-replaceable device, up to 6 characters long for magtape, or interchange diskette when used with

/INTERCHANGE) and an owner name (up to 12 characters long for a block-replaceable device, up to 10 characters long for 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 device RK1: and writes a volume identification on it.

```
.INITIALIZE/VOLUMEID RK1:
RK1:/Initialize? Are you sure? Y
 Volume ID? FORTRAN VOL
 Owner?     Mary
```

Use /VOLUMEID:ONLY to write a new volume identification on a device without reinitializing the device. You cannot change the volume ID of a magtape without initializing the entire tape.

/WAIT The /WAIT option is useful if you have a single-disk system. When you use this option to initialize a volume, the system begins the procedure but then pauses and waits for you to mount the volume you want to initialize. When the system pauses, it prints the following prompt at the terminal:

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

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 or any string beginning with Y, followed by a carriage return, to continue the initialization operation. Type N or any string beginning with N, or two CTRL/Cs, to abort the operation and return control to the keyboard monitor. Any other response causes the message to repeat.

After the system completes the initialization process, the system prints the following message prompting you to mount the system volume:

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

Mount the system volume and type Y or any string beginning with Y, followed by a carriage return. If you type any other response the system continues to prompt you to mount the system volume until you type Y.

When you use /WAIT, make sure that DUP, and FILEX if necessary, are on the system volume.

INSTALL

The INSTALL command installs the device you specify into the system.

```
INSTALL /SI device[,...device]
```

In the command syntax shown above, device represents the name of the device to be installed. The INSTALL command accepts no options. It allows you to install into the system tables a device that was not installed into the system when it was bootstrapped. (A device handler must exist in the system tables before you can use that device.) The device occupies the first available device slot. Using the INSTALL command does not change the monitor disk image; it only modifies the system tables of the monitor that is currently in memory.

You can enter the command on one line, or you can rely on the system to prompt you for information. The INSTALL command prompt is *Device?*.

When you specify a device name, the system searches the system volume for the corresponding device handler file. For SJ and FB systems, if LP: is to be installed the INSTALL command searches for the file SY:LP.SYS. For XM systems, INSTALL searches for SY:LPX.SYS. The INSTALL command does not allow a device handler built for a different configuration of the system to be installed in a given system. Note that you cannot install the device names SY, DK, or BA, or a logical device name that is the same as an already installed physical device name.

To permanently install a device, include the INSTALL command in the standard, system start-up indirect command file. This file is invoked automatically when you boot the system. The INSTALL command also allows you to configure a special system for a single session without having to reconfigure to revert to the standard device configuration. If there are no free device slots (use the SHOW DEVICES command to ascertain this), you must remove an existing device (with the REMOVE command) before you can install a new device.

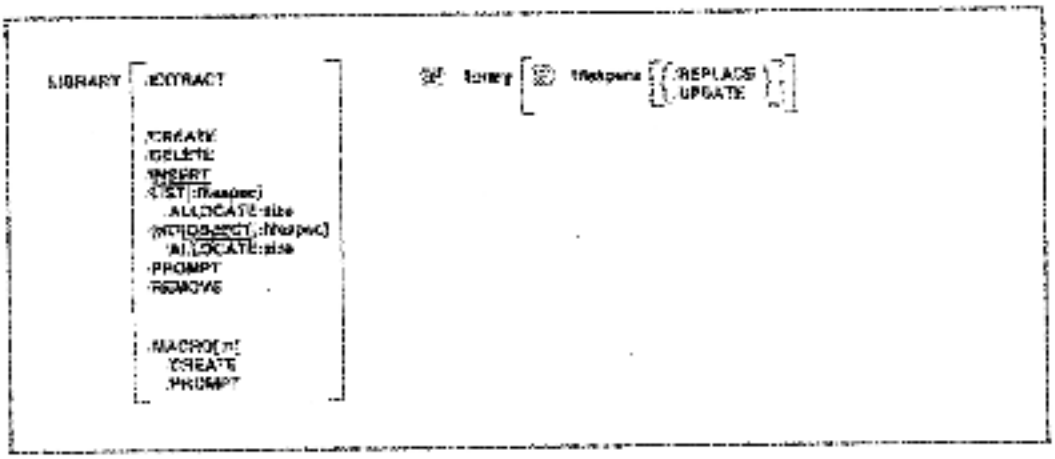
The following command installs the serial line printer into the system tables from the file LS.SYS. (The colon (:) that follows the device handler name is optional.)

```
.INSTALL LS:
```

The next example installs the line printer, RK05, and DY.

```
.INSTALL LP:/RK:/DY:
```

The LIBRARY command lets you create, update, modify, list, and maintain library files.



In the command syntax illustrated above, library represents the library file name, and filespecs represents the input module file names. Separate the library file specification from the module file specifications with a space. Separate the module file specifications with commas.

The system uses .LST as the default file type for library directory listing files. It uses .OBJ as the default file type for object libraries and object input files, and .MAC for macro input files. The default output file type for macro library files is .MLB. Object libraries contain machine-level object modules, and macro libraries contain MACRO source modules. You cannot combine object modules with MACRO modules.

The default operation, if you do not specify an option, is /INSERT. If you do not specify a library file in the command line, the system prompts *Library?*. If you specify /CREATE, /INSERT, or /MACRO and omit the module file specification, the system prompts *Files?*. If you specify /EXTRACT, the system prompts *File?*. Note that no other option causes the *File?* or *Files?* prompt.

The LIBRARY command can perform all the functions listed above on object library files. It can also create macro library files for use with the MACRO–11 assembler. A library file is a direct-access file (a file that has a directory) that contains one or more modules of the same type. The system organizes library files so the linker and MACRO–11 assembler can access them rapidly. Each library is a file that contains a library header, library directory, and one or more object modules. The object modules in a library file can be routines that are repeatedly used in a program, routines that are used by more than one program, or routines that are related and simply gathered together for convenience. An example of a typical object library file is the default system library, SYSLIB.OBJ, used by the linker. An example of a macro library file is SYSMAC.SML.

You access object modules in a library file by making calls or references to their global symbols; you link the object modules with the program that uses them by using the LINK command to produce a single executable module. Each input file for an object library consists of one or more object modules, and is stored on a device under a specific file name and file type. Once you insert an object module into a library file, you no longer reference the module by the file name of which it was a part; reference it by its individual module name. For example, the input file FORT.OBJ may exist on DT2: and can contain an object module called ABC. Once you insert the module into a library, reference only ABC, and not FORT.OBJ.

The input files normally do not contain main programs but only subprograms, functions, and subroutines. The library file must never contain a FORTRAN BLOCK DATA subprogram because there is no undefined global symbol to cause the linker to load it automatically.

The following sections describe the LIBRARY command options and explain how to use them. The last section under this command describes the LIBRARY prompting sequence and order of execution for commands that combine two or more LIBRARY options. Chapter 10 of the *RT-11 System Utilities Manual* contains more detailed information on object and macro libraries.

/ALLOCATE:size Use this option only with /LIST or /OBJECT to reserve space on the device for the output file. The value size represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 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 Use this option to delete an object module and all its associated global symbols from a library file directory. Since the module is deleted only from the directory (the object module itself is not deleted), the module and all global symbols that were previously deleted are restored whenever you update that library, unless you use /DELETE again to delete them. Specify the library name in the command line.

The system prompts you for the names of the modules to delete. The prompt is:

```
Module name?
```

Respond with the name of a module. (Be sure to specify a module name and not a global name.) Follow each module name with a carriage return. Enter a carriage return on a line by itself to terminate the list of module names.

The following example deletes modules SGN and TAN from the library called NEWLIB.OBJ.

```
.LIBRARY/DELETE NEWLIB
Module name? SGN
Module name? TAN
Module name?
```

/EXTRACT Use this option to extract an object module from a library and store it in a file with the same name as the module and a file type of .OBJ. You cannot combine this option with any other option.

The system prompts you for the name of the object module to be extracted. The prompt is:

```
Global?
```

If you specify a global name, the system extracts the entire module of which that global is a part. Follow each global name with a carriage return. Enter a carriage return on a line by itself to terminate the list of global symbols. The following example shows how to extract the module ATAN from the library called NEWLIB.OBJ and store it in file ATAN.OBJ on DX1:.

```
.LIBRARY/EXTRACT
Library? NEWLIB
File    ? DX1:ATAN
Global ? ATAN
Global s?
```

/INSERT Use this option to insert an object module into an existing library. Although you can insert object modules that have duplicate names, this practice is not recommended because of the difficulty involved in replacing or updating these modules. Note that /INSERT is the default operation. If you do not specify any option, insertion takes place.

The following example inserts the modules contained in the files THIRD.OBJ and FOURTH.OBJ into the library called OLDLIB.OBJ.

```
.LIBRARY/INSERT OLDLIB THIRD+FOURTH
```

/LIST[:filespec] Use this option to obtain a directory listing of an object library. Note that anytime you type a colon after the /LIST option (/LIST:), you must include a device or file specification following the colon.

The following example obtains a directory listing of OLDLIB.OBJ on the terminal (the line printer is the default device).

```
.LIBRARY/LIST:TT: OLDLIB
```

The directory listing prints global symbol names. A plus sign (+) in the module column indicates a continued line. See Section 10.2.8 in Chapter 10 of 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 prints a directory listing of the library on the terminal.

```
.LIBRARY/INSERT/LIST:TT: OLDLIB THIRD,FOURTH
```

You cannot obtain a directory listing of a macro library.

Make sure when you use /LIST with LIBRARY that you use it on the command side of the command string, and not after the file specification.

/MACRO[:n] Use this option to create a macro library. The optional argument *n* represents the size (in blocks) of the macro name directory. Note that this is the only valid function for a macro library. You can create a macro library, but you cannot list or modify it. To update a macro library, simply edit the ASCII text file and then reprocess the file with the LIBRARY/MACRO command.

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.

/OBJECT[:filespec] The system creates object library files by default as a result of executing a LIBRARY command. When you modify an existing library, the system actually makes the changes to the library you specify, thus creating a new, updated library that it stores under the same name as the original library. Use this option to give a new name to the updated library file and preserve the original library.

The following example creates a library called NEWLIB.OBJ, which consists of the library OLDLIB.OBJ plus the modules that are contained in files THIRD.OBJ and FOURTH.OBJ.

```
$LIBRARY/INSERT/OBJECT=NEWLIB OLDLIB THIRD,FOURTH
```

/NOOBJECT Use this option to suppress the creation of a new object library as a result of a LIBRARY command.

/PROMPT Use this option to specify more than one line of input file specifications in a LIBRARY command. This option is valid with all other library functions except the /EXTRACT option. You must specify // as the last input in order to terminate the input list. Note that the file specifications you enter after typing the /PROMPT option must conform to Command String Interpreter (CSI) conventions.

The following example creates a macro library called MACLIB.MLB from seven input files.

```
$LIBRARY/MACRO/PROMPT MACLIB A,B,C,D
*E,F,G
*//
```

/REMOVE This option permits you to delete a specific global symbol from a library file's directory. Since globals are deleted only from the directory (and not from the object module itself), all the globals that were previously deleted are restored whenever you update that library, unless you use /REMOVE again to delete them. This feature lets you recover a library if you have inadvertently deleted the wrong global.

The system prompts you for the names of the global symbols to remove. The prompt is:

```
Global?
```

Respond with the name of a global symbol to be removed. Follow each global symbol with a carriage return. Enter a carriage return on a line by itself to terminate the list of global symbols.

The following example deletes the globals GA, GB, GC, and GD from the library OLDLIB.OBJ.

```
$LIBRARY/REMOVE OLDLIB
Global? GA
Global? GB
Global? GC
Global? GD
Global?
```

/REPLACE Use this option to replace modules in an existing object library with modules of the same name contained in the files you specify.

The following example replaces a module called SQRT in the library MATHLB.OBJ with a new module, also called SQRT, from the file called MFUNCT.OBJ.

```
$LIBRARY MATHLB MFUNCT/REPLACE
```

Note that the /REPLACE option must follow each file specification that contains a module to be inserted into the library. Note also that you can use /REPLACE only with modules, and never with library files.

/UPDATE This option combines the functions of /INSERT and /REPLACE. Specify it after each file specification to which it applies. If the modules in the input file already exist in the library, the system replaces those library modules. If the modules in the input file do not exist in the library, the system inserts them.

The following example updates the library OLDLIB.OBJ.

```
$LIBRARY OLDLIB FIRST/UPDATE,SECOND/UPDATE
```

Note that the /UPDATE option must follow each file specification to which it applies, and that you can use this option only with modules, not files.

You can combine the LIBRARY options with the exceptions of /EXTRACT and /MACRO, which you cannot combine with most of the other functions. Table 4–9 lists the sequence in which the system executes the LIBRARY options and prompts you for additional information.

Table 4–9: Execution and Prompting Sequence of LIBRARY Options

Option	Prompt
/CREATE	
/DELETE	Module name?
/REMOVE	Global?
/UPDATE	
/REPLACE	
/INSERT	
/LIST	

The following example combines several options.

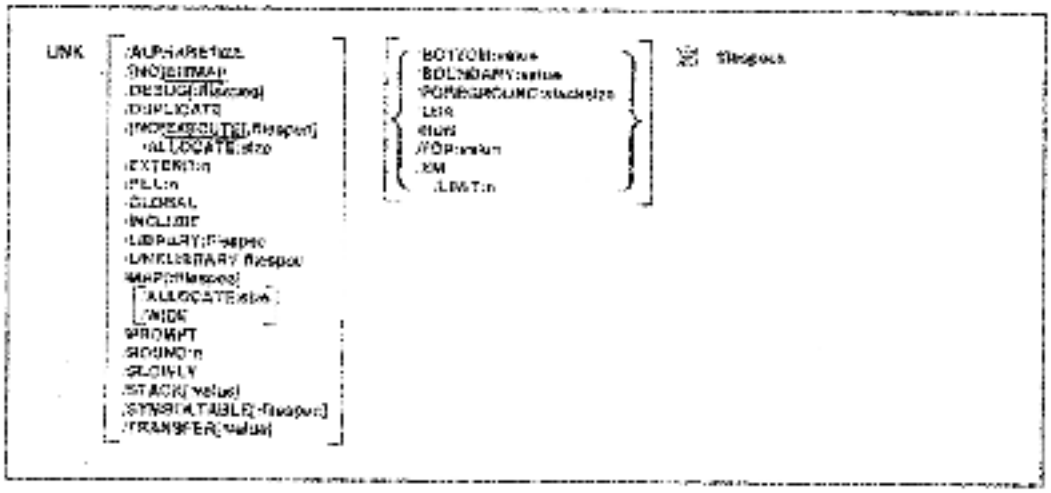
```
$LIBRARY/LIST:TT:/REMOVE/INSERT NEWLIB LIB2/REPLACE,LIB3
Global? SQRT
Global?
RT-11 LIBRARIAN V05.01  FRI 24-JAN-89 00:08:37
NEWLIB                  FRI 24-JAN-89 00:08:35

MODULE          GLOBALS         GLOBALS         GLOBALS

                COS             SIN
                DATAN           DATAN2
                ATAN            ATAN2
                DCOS            DSIN
```

The command executes in the following sequence:

1. Removes global SQRT from NEWLIB
2. Replaces any duplicates of the modules in the file LIB2.OBJ
3. Inserts the modules in the file LIB3.OBJ
4. Lists the directory of NEWLIB.OBJ on the terminal

LINK

The LINK command converts object modules into a format suitable for loading and execution.



The RT–11 system lets you separately assemble a main program and each of its subroutines without assigning an absolute load address at assembly time. The linker can then process the object modules of the main program and subroutines to relocate each object module and assign absolute addresses. It links the modules by correlating global symbols that are defined in one module and referenced in another, and it creates the initial control block for the linked program. The linker can also create an overlay structure (if you specify the /PROMPT option) and include the necessary run-time overlay handlers and tables. The linker searches libraries you specify to locate unresolved global symbols, and it automatically searches the default system subroutine library, SYSLIB.OBJ, to locate any remaining unresolved globals. Finally, the linker produces a load map (if you specify /MAP) that shows the layout of the executable module. The linker also can produce an STB file. See Chapter 11 of the *RT–11 System Utilities Manual* for a more detailed explanation of the RT–11 linker.

In the command syntax illustrated above, filespecs represents the object modules to be linked. Each input module should be stored on a random-access device (disk, diskette, or DECtape II); the output device for the load map file can be any RT–11 device. The output for an .LDA file (if you specify /LDA) can also be any RT–11 device, even those that are not block replaceable such as paper tape.

The default file types are as follows:

Load Module:	.SAV, .REL(FOREGROUND), .LDA(/LDA)
Map Output:	.MAP
Object Module:	.OBJ
Symbol Table File:	.STB

If you specify two or more files to be linked, separate the files by commas. The system creates an executable file with the same name as the first file in the input list (unless you use /EXECUTE to change it).

Table 4–10 summarizes the LINK prompting sequence for commands that combine two or more LINK options.

Table 4–10: Prompting Sequence for LINK Options

Option	Prompt
/TRANSFER	Transfer symbol?
/STACK	Stack symbol?
/EXTEND:n	Extend section?
/BOUNDARY:value	Boundary section?
/ROUND:n	Round section?
/INCLUDE	Library search?
/DUPLICATE	Duplicate symbol?

If you combine any of the options listed in Table 4–10, the system 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, terminate your response with a carriage return. Terminate your list of responses to the *Library search?* and *Duplicate symbol?* prompts by typing an extra carriage return. Note that if the command lines are in an indirect file and the system encounters an end-of-file before all the prompting information has been supplied, it prints the prompt messages on the terminal.

The LINK command options and explanations of how to use them follow.

/ALLOCATE:size Use this option with /EXECUTE or /MAP to reserve space on the device for the output file. The argument size represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 65535. A value of –1 is a special case that creates the largest file possible on the device. When used with /EXECUTE, /ALLOCATE is valid only when you are generating a .REL or .LDA file.

/ALPHABETIZE When you use this option, the linker lists in the load map your program's global symbols in alphabetical order.

/BITMAP Use this option if you want the linker to create a memory usage bitmap. This is the default setting.

/NOBITMAP Use this option if you do not want the linker to create a memory usage bitmap. This option is useful if you are preparing your program for ROM storage and its code lies between locations 360 and 377 inclusive. */BITMAP* is the default setting.

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

/BOUNDARY:value Use the /BOUNDARY option to start a specific program section in the root on a particular address boundary. The system generates a whole-number multiple of the value you specify for the starting address of the program section. The argument value must be a power of 2. The system extends the size of the previous program section to accommodate the new starting address for the specific section.

When you have entered the complete LINK command, the system prompts you for the name of the section whose starting address you need to modify. The prompt is:

```
Boundary section?
```

Respond with the appropriate program section name and terminate your response with a carriage return.

/DEBUG[:filespec] Use this option to link ODT (on-line debugging technique, described in Chapter 20 of 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 system links the debugger low in memory relative to your program.

/DUPLICATE Use this option to place 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, the system 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 a carriage return after each global symbol. Type a carriage return on a line by itself to terminate the list.

See Chapter 11 of the *RT-11 System Utilities Manual* for more information on duplicating library modules.

/EXECUTE[:filespec] Use this option to specify a file name or device for the executable file. Note that anytime you type a colon after the /EXECUTE option (/EXECUTE:), you must include a device or file specification following the colon. Because the LINK command creates executable files by default, the following two commands have the same meaning:

```
.LINK MYPROG

.LINK/EXECUTE MYPROG
```

Both commands link MYPROG.OBJ and produce MYPROG.SAV as a result.

The /EXECUTE option has different meanings when it follows the command and when it follows the file specification. The following command creates an executable file called PROG1.SAV on device DL1:.

```
.LINK/EXECUTE:DL1:PROG1 PROG2
```

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

```
.LINK [376,123]MYPROG/EXECUTE
```

/NOEXECUTE Use this option to suppress creation of an executable file.

/EXTEND:n This option allows you to extend a program section to a specific octal value *n*. The resultant program section size is equal to or greater than the value you specify, depending on the space the object code requires.

When you have entered the complete LINK command, the system prompts you for the name of the program section you need to extend. The prompt is:

```
Extend section?
```

Respond with the appropriate program section name, and terminate your response with a carriage return.

/FILL:n Use this option to initialize unused locations in the load module and place a specific octal value *n* in those locations. Note that the linker automatically initializes to 0 unused locations in the load module; use this option to place another value in those locations. This option can be useful in eliminating random results that occur when a program references uninitialized memory by mistake. It can also help you to determine which locations have been modified by the program and which remain unchanged.

/FOREGROUND[:stacksize] This option produces an executable file in relocatable (.REL) format for use as a foreground job under the FB or XM monitor. You cannot use .REL files under the SJ monitor.

This option assigns the default file type .REL to the executable file. The argument stacksize represents the number of bytes of stack space to allocate for the foreground job. The value you supply is interpreted as an octal number; specify an even number. Follow n with a decimal point (n.) to represent a decimal number. The default value is 128 (decimal) or 200 (octal) bytes of stack space. DIGITAL recommends that you allocate 256 bytes of stack space when linking a FORTRAN program to run in the foreground.

You can use /FOREGROUND[:stacksize] with /XM to link privileged foreground jobs with virtual overlays. See Chapter 11 of the *RT–11 System Utilities Manual* for more detailed information on linking privileged foreground jobs with virtual overlays.

/GLOBAL Use this option to generate 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 Chapter 11 of 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 the system 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:DL1: 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 MACRO cross-reference listing temporary file CREF.TMP, that device will also serve as the default device for the LINK global symbol cross-reference temporary file.

/INCLUDE This option lets you take global symbols from any library and include them in the linked memory image. When you use /INCLUDE, the linker loads modules that are not called by other modules from a library into the root.

When you have entered the complete LINK command, the system prompts you for a list of global symbols to include in the load module. The prompt is:

```
Library search?
```

Respond by typing the global symbols to be included in the load module. Type a carriage return after each global symbol. Type a carriage return in response to the *Library search?* prompt itself to terminate the list.

/LDA This option produces an executable file in LDA format. The LDA-format file can be output to any device, including those that are not block-replaceable. The default file type .LDA is assigned by /LDA to the executable file. This option is useful for files that you need to load with the Absolute Binary Loader.

/LIBRARY This option is the same as /LINKLIBRARY. It is included here only for system compatibility.

/LIMITn Use the /LIMIT:n option with /XM to limit the amount of memory allocated by a .SETTOP programmed request to n (octal) K words. If you do not use the /LIMIT option, a .SETTOP request allocates up to 32K words or, if less than 32K words of physical memory are available, as much memory as is available.

/LINKLIBRARY:filespec You can use this option to include the library file you specify as an object module library in the linking operation. Because the system automatically recognizes library files in the linking operation you do not normally need this option; it is provided for compatibility with the EXECUTE command.

/MAP[:filespec] You must specify this option to produce a load map listing. Note that anytime you type a colon after the /MAP option (/MAP:), you must include a device or file specification following the colon.

The /MAP option has different meanings depending on where you put it in the command line. If you specify /MAP without a filespec in the list of options that immediately follows the command name, the system generates a listing that prints on the line printer. If you follow /MAP with a device name, the system creates a map file on that device. If the device is a file-structured device, the system stores the listing file on that device, assigning it the same name as the first input file and a .MAP file type.

The following command produces a load map on the terminal.

```
.LINK/MAP:TT:  MYPROG
```

The next command creates a map listing file called MYPROG.MAP on RK3:.

```
.LINK/MAP:RK3:  MYPROG
```

If the /MAP option contains a name and file type to override the default of .MAP, the system generates a listing with that name. The following command, for example, links PROG1 and PROG2, producing a map listing file called MAP.OUT on device DK:.

```
.LINK/MAP:MAP.OUT  PROG1,PROG2
```

Another way to specify /MAP is to type it after the file specification to which it applies. To link a file and produce a map listing file with the same name, use a command similar to this one.

```
.LINK  PROG1,PROG2/EXECUTE/MAP
```

The command shown above links PROG1 and PROG2, producing files PROG2.SAV and PROG2.MAP. If you specify a file name on a /MAP option following a file specification in the command line, it has the same meaning as when it follows the command.

/PROMPT Use this option to enter additional lines of input. The system continues to accept lines of linker input until you enter two slashes (//). Chapter 11 of 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 Chapter 1, 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
*TAN/O:1
*COS1/O:1
*SIN0/O:2
*LN13/O:2//
```

The /PROMPT option also gives you a convenient way to create an overlaid program from an indirect file. The file PROMPT.COM contains these lines:

```
@/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:n This option rounds up the section you specify so that the size of the root segment is a whole-number multiple of the value n you supply. The argument n must be a power of 2.

When you have entered the complete LINK command, the system prompts you for the name of the section that you need to round up. The prompt is:

```
Round section?
```

Respond with the appropriate program section name, and terminate your response with a carriage return.

/RUN Use this option to initiate execution of the resultant .SAV file. This option is valid for background jobs only. Do not use /RUN with any option that requires a response from the terminal.

/SLOWLY This option instructs the system to allow the largest possible memory area for the link symbol table at the expense of making the link process slower. Use this option only if an attempt to link a program failed because of symbol table overflow.

/STACK[:value] This option lets you modify 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 argument value is an even, unsigned, six-digit octal number that defines the stack address.

When you have entered the complete LINK command, the system prints the following prompt message if you did not already specify a value:

```
Stack symbol?
```

Respond with the global symbol whose value is the stack address. You cannot specify a number at this point. Terminate your response with a carriage return. If you specify a nonexistent symbol, the system prints an error message. It then sets the stack address to 1000 (for memory image files) or to the bottom address if you used /BOTTOM.

/SYMBOLTABLE[:filespec] When you use this option, the linker creates a file that contains symbol definitions for all the global symbols in the load module. Enter the symbol table file specification as the third output specification in the LINK command line. If you do not specify a file name, the linker uses the name of the first input file and assigns the file type .STB.

Note that anytime you type a colon after the /SYMBOLTABLE option (/SYMBOLTABLE:), you must include a device or file specification following the colon. By default, the system does not create a symbol table file.

The following example creates the symbol table file BTAN.STB:

```
.LINK BOBJ,BOBJ/SYMBOLTABLE:BTAN
```

/TOP:value Use this option to specify the highest address to be used by the relocatable code in the load module. The argument value represents an unsigned, even octal number.

/TRANSFER[:value] The transfer address is the address at which a program starts when you initiate execution with R, RUN, FRUN, or SRUN. The /TRANSFER option lets you specify the start address of the load module. The argument value 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, the system prints 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 a carriage return. If you specify a nonexistent symbol, an error message prints and the linker sets the transfer address to 1 so that the system cannot execute the program.

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

/XM When you use this option, you enable special .SETTOP and .LIMIT features provided in the XM monitor. This option allows a virtual job to map a scratch region in extended memory with the .SETTOP programmed request. See the *RT–11 Programmer's Reference Manual* and the *RT–11 Software Support Manual* for more details on these special features. You can use /XM with /FOREGROUND[:stacksize] to link privileged foreground jobs with virtual overlays.

If you want to create an extended memory overlay structure for your program, use the /PROMPT option. You can then specify on subsequent lines the overlay structure using the LINK /V option (see Chapter 11 of the *RT–11 System Utilities Manual*). Note that when you use /V to create an overlay structure, the linker automatically enables the special .SETTOP and .LIMIT features.

The LOAD command loads a device handler into memory for use with foreground, background, system jobs, or BATCH.

```
LOAD   dd)   device[=jobname][,...device[=jobname]]
```

In the command syntax shown above, device represents the device handler to be made resident; jobname assigns the device handler to the background job if it has the value B, or to the foreground if it has the value F. The jobname specification is invalid with the SJ monitor. Under a monitor that has system job support, jobname can be the logical job name of a system job.

The LOAD command helps control system execution by bringing a device handler into memory and optionally allocating the device to a job. The system allocates memory for the handler as needed. Before you use a device in a foreground program, you must first load the device handler. Also, if you have generated an XM monitor without fetchable handler support, or if your handler is not fetchable, you must load the device handler before the job is executed.

A device can be owned exclusively by either the foreground, background, or system job. (Note that BATCH, if running, is considered to be a background job under the FB and XM monitors.) This exclusive ownership prevents the input and output of two different jobs from being intermixed on the same non-file-structured device.

In the following example, magtape belongs to the background job, while RL02 is available for use by either the background, foreground, or system job; the line printer is owned by the foreground job. All three handlers are made resident in memory.

```
.LOAD DL:,MT:=B,LP:=F
```

For a monitor with system job support, the following example reserves the line printer for the system job QUEUE.

```
.LOAD LP:=QUEUE
```

Different units of the same random-access device controller can be owned by different jobs. Thus, for example, DL1: can belong to the background job, while DL5: 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 non-file-structured device handler, and assign ownership of that handler to a job, all units of that particular device become assigned to that job. This means no other job can use any unit of that particular device.

To change ownership of a device, use another LOAD command. It is not necessary to first unload the device. For example, if the line printer has been loaded into memory and assigned to the foreground job as in the example above, the following command reassigns it to the background job without unloading the handler first.

```
.LOAD LP:/B
```

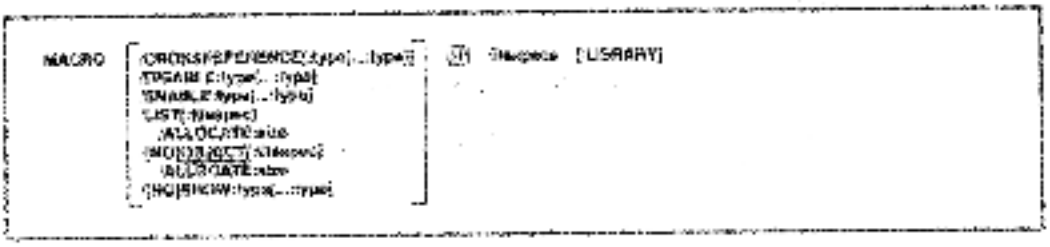
Note, however, that if you interrupt an operation that involves magtape, you must unload (with the UNLOAD command) then load the appropriate device handler (MM, MT, or MS). When using these handlers with the FB monitor, this restriction does not apply.

You cannot assign ownership of the system unit (the unit you bootstrapped) of a system device, and any attempt to do so is ignored. You can, however, assign ownership of other units of the same type as the system device. LOAD is valid for use with logical names. For example:

```
.ASSIGN DL1 XY
.LOAD XY:/F
```

If you are using a diskette, loading the necessary device handlers into memory can improve system performance significantly, since no handlers need to be loaded dynamically from the diskette. Use the SHOW command to display on the terminal the status of device handlers and device ownership.

The MACRO command invokes the MACRO assembler to assemble one or more source files.



In the command syntax shown above, filespecs represents one or more files to be included in the assembly. If you omit a file type for an input file, the system assumes .MAC. Output default file types are .LST for listing files and .OBJ for object files.

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

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

You can enter the MACRO command as one line, or you can rely on the system to prompt you for information. The MACRO command prompt is *Files?* for the input specification. The system prints on the terminal the number of errors MACRO detects during an assembly.

Chapter 12 of the *RT–11 System Utilities Manual* and the *PDP–11 MACRO Language Reference Manual* contain more detailed information about using MACRO. The options you can use with the MACRO command follow.

/ALLOCATE:size Use this option with /LIST or /OBJECT to reserve space on the device for the output file. The argument size represents the number of blocks of space to allocate. The meaningful range for this value is from 1 to 65535. A value of –1 is a special case that creates the largest file possible on the device.

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

get a cross-reference listing. The argument type represents a one-character code that indicates which sections of the cross-reference listing the assembler should include. Table 4–11 summarizes the arguments and their meaning.

Table 4–11: Cross-Reference Sections

Argument	Section Type
S	User-defined symbols
R	Register symbols
M	Macro symbolic names
P	Permanent symbols (instructions, directives)
C	Control sections (.CSECT symbolic names)
E	Error codes
None	Equivalent to /CR:M:E

/DISABLE:type[,...:type] Use this option to specify a MACRO .DSABL directive. See the *PDP-11 MACRO Language Reference Manual* for a description of the directive and a list of all valid types. Table 4–12 summarizes the arguments and their meaning.

Table 4–12: .DSABL and .ENABL Directive Summary

Argument	Default	Enables or Disables
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 8 of the *RT-11 Software Support 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] Use this option to specify a MACRO .ENABL directive. See the *PDP-11 MACRO Language Reference Manual* for a description of the directive and a list of all valid types. Table 4–12 summarizes the arguments and their meaning.

/LIBRARY This option identifies the file it qualifies as a library file; use it only after a library file specification in the command line. The MACRO

assembler looks first to the library file or files you specify and then to the system library, SYSMAC.SML, to satisfy references (made with the .MCALL directive) from MACRO programs.

In the example below, the command string includes two user libraries.

```
.MACRO A+LIB1/LIBRARY+A+LIB2/LIBRARY+B
```

When MACRO assembles file A, it looks first to the library, MYLIB1.MAC, and then to SYSMAC.SML to satisfy .MCALL references. When it assembles file B, MACRO searches MYLIB2.MAC, MYLIB1.MAC, and then SYSMAC.SML, in that order, to satisfy references.

/LIST[:filespec] You must specify this option to produce a MACRO assembly listing. Note that anytime you type a colon after the /LIST option (/LIST:), you must include a device or file specification following the colon.

The /LIST option has different meanings depending on where you place it in the command line.

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

```
.MACRO/LIST MYPROG
```

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

```
.MACRO MYPROG/LIST
```

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

The following command produces a listing on the terminal.

```
.MACRO/LIST:TT: A
```

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

```
.MACRO/LIST:RK3: A
```

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

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

You cannot use a command like the next one. In this example, the second listing file would replace the first one and cause an error.

```
.MACRO/LIST:FILE2 A,B
```

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

```
.MACRO A+B/LIST,RK3:
```

The above command assembles A.MAC and B.MAC, producing files DK:A.OBJ and RK3:B.LST.

If you specify a file name on a /LIST option following a file specification in the command line, it has the same meaning as when it follows the command. The following two commands have the same results:

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

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

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

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

```
.MACRO A/LIST,B
```

This command assembles A.MAC, producing A.OBJ and A.LST. It also assembles B.MAC, producing B.OBJ. However, it does not produce any listing file for the assembly of B.MAC.

/OBJECT[:filespec] Use this option to specify a file name or device for the object file. Note that anytime you type a colon after the /OBJECT option (/OBJECT:), you must include a device or file specification following the colon.

Because MACRO creates object files by default, the following two commands have the same meaning:

```
.MACRO A
```

```
.MACRO/OBJECT A
```

Both commands assemble A.MAC and produce A.OBJ as output.

The /OBJECT option functions like the /LIST option; it can be either a command 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 RK1:.

```
.MACRO/OBJECT:RK1: A+B
```

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 Use this option to suppress 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, the system assembles A.MAC and B.MAC together, producing files A.OBJ and B.LST. It also assembles C.MAC and produces C.LST, but does not produce C.OBJ.

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

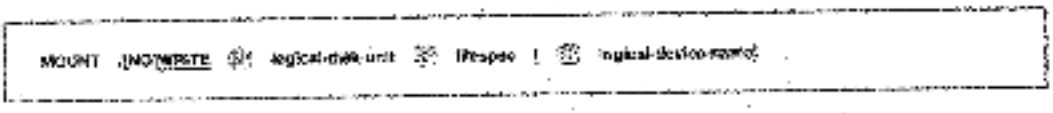
/SHOW:type Use this option to specify any MACRO .LIST directive. The *PDP-11 MACRO Language Reference Manual* explains how to use these directives. Table 4–13 summarizes the arguments and their meaning. Note that you must explicitly request a listing file with the /LIST option.

Table 4–13: .LIST and .NLIST Directive Summary

Argument	Default	Controls
SEQ	List	Source line sequence numbers
LOC	List	Location counter
BIN	List	Generated binary code
BEX	List	Binary extensions
SRC	List	Source code
COM	List	Comments
MD	List	Macro definitions, repeat range expansions
MC	List	Macro calls, repeat range expansions
ME	Nolist	Macro expansions
MEB	Nolist	Macro expansion binary code
CND	List	Unsatisfied conditionals, .IF and .ENDC statements
LD	Nolist	Listing directives with no arguments
TOC	List	Table of contents
TTM	Line printer width	Output format
SYM	List	Symbol table

/NOSHOW:type Use this option to specify any MACRO .NLIST directive. The *PDP-11 MACRO Language Reference Manual* explains how to use these directives. Table 4–13 summarizes the valid arguments and their meaning. Note that you must explicitly request a listing file with the /LIST option.

The MOUNT command associates the logical disk unit you specify with the file you specify.



In the command syntax illustrated above, logical-disk-unit represents the logical disk unit you want to mount. Specify the logical disk unit in the form LDn: (the colon is optional), where n is an integer in the range 0–7; or you can specify a logical device name that has already been assigned to the logical disk unit. The term filespec represents the file to be used as the logical disk. The default file type is .DSK. The optional term logical-device-name represents a logical device name you want to assign 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.

You can specify the entire command on one line, or you can rely on the system to prompt you for information. If you type MOUNT followed by a carriage return, the system prompts *Device?*. If you type the device name followed by a carriage return, the system prompts *File?*. The system does not prompt you for an optional logical device name; enter the logical device name on the same line as the file specification.

The following example associates logical disk unit 5 (LD5:) with the file DATA.DSK on device DL0:.

```
.MOUNT LD5: DL0:DATA
```

Use the SET LD CLEAN command to verify and correct logical disk assignments. See the SET command description for more information on SET LD CLEAN.

The next example associates LD6: with the file DL0: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

You must be careful to avoid accidentally destroying files while performing logical disk subsetting. You can assign logical disk unit numbers to both protected and system (.SYS) files, and write to those files.

The following sections describe MOUNT command options and include command examples.

/WRITE Use this option to write-enable a logical disk. This option allows you read/write access to the logical disk you specify. This is the default mode.

The following example associates LD1: with the file MYFILE.DSK on device DY1:. When the command is executed, the logical disk is write-enabled.

```
$MOUNT/WRITE LD1: DY1:MYFILE.DSK
```

/NOWRITE Use this option to write-protect a logical disk. This option allows you read-only access to the logical disk you specify. The default is /WRITE.

The following example write-protects LD0:.

```
$MOUNT/NOWRITE LD0: DY1:MYFILE.DSK
```

The PRINT command lists the contents of one or more files on the line printer.

In the command syntax illustrated above, filespecs represents the file or files to be printed. You can explicitly specify up to six files as input to the PRINT command. The system prints the files in the order in which you specify them in the command line. You can also use wildcards in the file specification. In this case, the system prints the files in the same order as they occur in the directory of the specified volume. If you specify more than one file, separate the files by commas. If you omit the file type for a file specification, the system assumes .LST. You can specify the entire command on one line, or you can rely on the system to prompt you for information. The PRINT command prompt is *Files?*.

If you are running QUEUE or SPOOL as either a foreground or system job, many of the PRINT commands are executed by these programs; therefore, the keyboard monitor may return the dot prompt (.) immediately. See Chapter 17 (QUEUE) or Chapter 18 (SPOOL) of the *RT-11 System Utilities Manual* for more information. 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.

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

[dd][:mmm][:yy]

where:

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

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), the system uses the values from the current system date. For example, if you specify only the year :82 and the current system date is May 4, 1983, the system uses the date 4-MAY-82. 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.)

The PRINT command options follow; they include command examples.

/BEFORE[:date] Use this option to print 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 DY0: created before April 21, 1983:

```
.PRINT/BEFORE:21-APR-83 DY0:*.MAC
```

/COPIES:n Use this option to print more than one copy of the file. The meaningful range of values for the decimal argument *n* is from 1 to 32 (1 is the default). The following command, for example, prints three copies of the file REPORT.LST on the line printer.

```
.PRINT/COPIES:3 REPORT
```

/DATE[:date] Use this option to print only those files with a certain creation date. If no date is specified the current system date is used. The following command prints all .MAC files created on April 21, 1983.

```
.PRINT/DATE:21-APR-83 *.*;*,MAC
```

/DELETE Use this option to delete a file after it lists on the line printer. This option must appear following the command in the command line. The PRINT/DELETE operation does not ask you for confirmation before it executes. You must use /QUERY for this function. The following example prints PROG1.BAS on the line printer, then deletes it from DY1:.

```
.PRINT/DELETE DY1:PROG1.BAS
```

/FLAGPAGE:n Use this option if you want banner pages for each file being printed, where *n* represents the number of banner pages you want for

each file. This option is valid only if you are running QUEUE. If you specify more than one file to be printed, QUEUE prints a banner page for each file.

If you use the /FLAGPAGE option but specify no value for n, the default number of banner pages set with the QUEMAN/P option will be printed. If the default number of banner pages set with the /P option is 0, n defaults to 1.

The banner page that QUEUE creates consists of a page showing the file name in large, block letters. The banner page also includes a trailer that lists the job name, the date and time the job was output, the copy number and number of copies in the job, and the input file specification.

NOTE

If you use the PRINT command to output files, and QUEUE is running, you may get banner pages even when you do not specify /FLAGPAGE. This condition is due to a default value you can set when you run QUEMAN, the background job that serves as an interface between you and QUEUE. The QUEMAN /P option sets the default number of banner pages for output jobs, so that each time you output a job, you get banner pages. This condition remains in effect until you reset it with the QUEMAN /P option. For more information on QUEMAN and the /P option, see Chapter 17, Queue Package, in the *RT-11 System Utilities Manual*.

If QUEMAN.SAV resides on the system device and you are using the transparent spooler (SPOOL), you can use the /FLAGPAGE:n option to override the default number of banner pages set with the SET SP FLAG=n command.

The following example prints three banner pages for each file in the command line.

```
.PRINT/FLAGPAGE:3 PROG1.PAS,PROG1.LST,PROG1.OTB
```

/NOFLAGPAGE Use this option if you do not want any banner pages printed for each of the files in the job you want printed. Use this option only if you are running QUEUE. This option is useful if you have previously set QUEMAN's /P option to create banner pages each time a job is output (see note above). The default setting is /NOFLAGPAGE unless you specify otherwise with the QUEMAN /P option.

/INFORMATION Use this option to change the severity level of the error message that prints when not all of the input files you specified are found. If you do not use /INFORMATION, the system prints an error message when it is unable to find an input file, and execution halts after the

command is processed. When you use /INFORMATION, the system prints an informational message to tell you which files it cannot find, but execution continues.

In the following example, the system prints input files FILE1.TXT and FILE3.TXT. However, since the system is unable to find DLC:FILE2.TXT, the system prints a message to inform you.

```
.PRINT/INFORMATION DLC:FILE1,FILE2,FILE3.TXT
?PIP-I-File not found DLC:FILE2.TXT
```

/LOG This option lists on the terminal the names of the files that are printed by the current command. Normally, the system prints a log only if there is a wildcard in the file specification. If you specify /QUERY, the query messages replace the log, unless you specifically type /LOG/QUERY in the command line.

The following example shows a PRINT command and the resulting log.

```
.PRINT/LOG/DELETE REPORT
 Files copied/deleted:
DK:REPORT.LST to LP:
```

/NOLOG This option prevents a list of the files copied from typing out on the terminal. You can use this option to suppress the log when you use a wildcard in the file specification.

/NAME:[dev:]jobname Use this option to specify a job name for the files you want printed. This option is valid only if you are running QUEUE. You can use up to six alphanumeric characters for the job name. If you specify a device with the job name, you can send the files to that device, permitting you to send files to any valid RT-11 device. If you send the files to a mass storage volume, the system 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 DX1:. The job name JOB5 appears in the JOBNAME field of the banner page.

```
.PRINT/NAME:DX1:JOB5*FILE1,FILE2,FILE3
```

The files from this example reside on DX1: as JOB5.JOB.

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

```
.PRINT/NEWFILES *.LST
 Files queued:
DK:OUTFIL.LST    to LP:
DK:REPORT.LST    to LP:
```

/PRINTER Use this option to force files to be copied to the line printer. Use this option when you are running QUEUE if you want to perform other tasks while you want print a file. This option causes PIP to copy the file to the line printer, which bypasses QUEUE processing.

/PROMPT Use this option to continue a command string on subsequent lines. This option is valid only if you are running QUEUE. When you use /PROMPT, you can enter file specifications on subsequent lines directly to QUEMAN, described in Chapter 17 of 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 If you use this option, the system requests confirmation from you before it performs the operation. /QUERY is particularly useful on operations that involve wildcards, when you may not be sure which files the system selected for an operation. Note that if you specify /QUERY in a PRINT command line that also contains a wildcard in the file specification, the confirmation messages that print on the terminal replace the log messages that would normally appear. You must respond to a query message by typing Y or any string beginning with Y, followed by a carriage return, to initiate execution of a particular operation. The system interprets any other response to mean NO: it does not perform the specific operation. The following example uses /QUERY.

```
.PRINT/QUERY *.LST
 Files queued?
DX:OUTFIL.LST    to LP:? N
DX:REPORT.LST    to LP:? Y
```

/SINCE[:date] Use this option to print 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 DY0: created on or after April 21, 1983:

```
.PRINT/SINCE:21-APR-83 DY0:*.MAC
```

/WAIT The /WAIT option is useful if you have a single-disk system. When you use this option, the system initiates the PRINT operation, but then pauses and waits for you to mount the volume that contains the files you want to print.

When the system pauses, it prints *Mount input volume in <device>: Continue?*. Mount the input volume and type Y or any string beginning with Y, followed by a carriage return, to continue the print operation. Type N or any string beginning with N, or two CTRL/Cs, to abort the operation and return control to the keyboard monitor. Any other response causes the message to repeat.

After the system completes the PRINT operation the system prints the following message prompting you to mount the system volume:

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

Mount the system volume and type Y or any string beginning with Y, followed by a carriage return. If you type any other response the system continues to prompt you to mount the system volume until you type Y.

The following command line prints ERREX.MAC from DL0:

```
.PRINT/WAIT DL0:ERREX.MAC
Mount input volume in DL0:? Continue? Y
Mount system volume in DL0:? Continue? Y
```

In the case of PRINT, the system prints the file or files you specify before it prints *Mount system volume in <device>: Continue?*. Make sure when you use /WAIT that PIP is on the system volume. This option is invalid if QUEUE is running.

The PROTECT command protects a file so you cannot delete the file until you remove the protection. (See the UNPROTECT command later in this section.)

In the command syntax illustrated above, filespecs represents the file or files you want to protect. You can explicitly specify up to six files. If you specify more than one file, separate the files with commas. You can also use wildcards in the file specifications. You can enter the PROTECT command as one line, or you can rely on the system to prompt you for information. The PROTECT command prompt is *Files?*.

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

[dd][-mmm[-][yy]

where:

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

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), the system uses the values from the current system date. For example, if you specify only the year :82 and the current system date is May 4, 1989, the system uses the date 4-MAY-82. 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.)

The following sections describe options you can use with the PROTECT command and include command examples.

/BEFORE[:date] Use this option to protect 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, 1983.

```
.PROTECT/BEFORE:20-MAR-83 *.MAC
 Files protected:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

/DATE[:date] Use this option to protect 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, 1983.

```
.PROTECT/DATE:20-MAR-83 *.MAC
 Files protected:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

/EXCLUDE This option protects all the files on a device except the ones you specify. The following command, for example, protects all files on DY0: except .SAV files.

```
.PROTECT/EXCLUDE DY0:*.SAV
?PIP-W-No .SYS action
 Files protected:
DY0:ABC.OLD
DY0:RAF.OLDY
DY0:COM8.
DY0:MERGE.OLD
```

/INFORMATION Use this option to change the severity level of the error message that prints when not all of the input files you specified are found. If you do not use /INFORMATION, the system prints an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, the system prints 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 the system is unable to find DL0:FILE2.TXT, the system prints a message to inform you.

```
.PROTECT/INFORMATION DL0:(FILE1,FILE2,FILE3).TXT
?PIP-I-File not found DL0:FILE2.TXT
```

/LOG This option 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, the system does not ask you for confirmation before execution proceeds. Use both /LOG and /QUERY to invoke logging and querying.

/NOLOG This option prevents a list of files being protected from printing on the terminal.

/NEWFILES Use this option to protect only the files that have the current system date. The following example protects the files created today.

```
.PROTECT/NEWFILES DY1:*.BAK
 Files Protected:
DY1:MERGE.BAK
```

/QUERY Use this option to request 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. You must respond to a query message by typing Y or any string beginning with Y, followed by a carriage return, to initiate execution of a particular operation. The system interprets any other response as NO; it does not perform the operation.

The following example shows querying. Only the file DY1:AAF.MAC is protected:

```
.PROTECT/QUERY DY1:*.*
 Files Protected:
DY1:ABC.MAC    ? N
DY1:AAF.MAC    ? Y
DY1:MERGE.FOR  ? N
```

/SETDATE[:date] This option causes the system 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, the system preserves the existing file creation date when it protects a file.

The following example protects files and changes their dates to the current system date.

```
.PROTECT/SETDATE DY0:*.FOR
 Files Protected:
DY0:ABC.FOR
DY0:AAF.FOR
DY0:MERGE.FOR
```

/SINCE[:date] Use this option to protect 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 DY0: that were created on or after April 21, 1983:

```
.PROTECT/SINCE:21-APR-83 DY0:*.MAC
 Files protected:
DY0:A.MAC
DY0:B.MAC
DY0: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, the system files are excluded from the protect operation and a message is printed on the terminal to remind you of this.

This example protects all files on DY0: with the file name MM, including .SYS files.

```
.PROTECT/SYSTEM DY0:MM.*
 Files protected:
DY0:MM.MAC
DY0:MM.OBJ
DY0:MM.SAV
DY0:MM.SYS
```

/WAIT When you use this option, the system initiates the PROTECT operation but then pauses for you to mount the volume that contains the files you want to protect. This option is especially useful if you have a single-disk system.

When the system pauses, it prints *Mount input volume in <device>; Continue?*, where <device> represents the device into which you mount the volume. Mount the volume and type Y or any string beginning with Y, followed by a carriage return. Type N or any string beginning with N, or two CTRL/Cs, to abort the operation and return control to the keyboard monitor. Any other response causes the message to repeat.

When the operation completes the system prints the *Continue?* message again. Mount the system volume and type Y or any string beginning with Y, followed by a carriage return. If you type any other response the system prompts you to mount the system volume until you type Y. The system then prints the keyboard monitor prompt. Make sure PIP is on your system volume when you use the /WAIT option.

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
```

The R command loads a memory image file from the system device into memory and starts execution.

```
R @[ filespec
```

In the command syntax shown above, filespec represents the program to be executed. The default file type is .SAV. The only valid device is SY:. The R command is similar to the RUN command except that the file you specify in an R command string must be on the system device (SY:). Use the R command only with background jobs including privileged jobs under the XM monitor. (Use FRUN to execute a foreground job under the FB or XM monitor.)

The following command loads and executes MYPROG.SAV from device SY:

```
.R MYPROG
```

You can use the R command to execute a background virtual job under the XM monitor. The R command creates a virtual memory partition for the job, creates a region 0 and window 0 definition block, and sets up the user mapping registers.

REENTER

The REENTER command starts the program at its reentry address (the start address minus 2).

```
REENTER
```

The REENTER command accepts no options or arguments. REENTER does not clear or reset any memory areas. Use it to avoid reloading the same program for subsequent execution. You can use REENTER to return to a system program or to any program that allows for a REENTER after the program terminates. You can also use REENTER after you have used two CTRL/Cs to interrupt those programs.

If you issue the REENTER command and it is not valid, the message *?KMON-F-Invalid command* is printed. 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. Two CTRL/Cs interrupt the listing and return to the monitor. REENTER starts DIR at its reentry address, and DIR prompts for a line of input.

```
.R DIR
*LP:=DK:*.*
^C
^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.

The REMOVE command removes a device name from the system tables.

```
REMOVE @ device[,device]
```

In the command syntax shown above, device represents the device to be removed from the system tables. You can enter the REMOVE command on one line, or you can rely on the system to prompt you for information. The REMOVE command prompt is *Device?*.

Using the REMOVE command does not change the monitor disk image; it only modifies the system tables of the monitor currently in core. This allows you to configure a special system for a single session at the computer without having to reconfigure to return to your standard device configuration. Bootstrapping the system device restores the original device configuration. To permanently REMOVE a device, include the REMOVE command in the standard system start-up indirect command 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), or TT (the terminal handler). If you attempt to REMOVE a device that does not exist in the running monitor's system table, the system prints an error message. You can use the INSTALL command to install a new device after using the REMOVE command to remove a device (thus creating a free device slot).

The following command removes the line printer handler and the card reader handler from the system. Note that the colons (:) are optional.

```
.REMOVE  LP,CR
```

Use the SHOW command to display on the terminal a list of devices that are currently available on your system.

RENAME

The RENAME command assigns a new name to an existing file.

In the command syntax illustrated above, input-filespec represents the file(s) to be renamed, and output-filespec represents the new name. You can specify up to six input files, but only one output file. Note that the device specification must be the same for input and output; you cannot rename a file from one device to another. If a file exists with the same name and file type as the output file you specify, the system deletes the existing file unless you use the /NOREPLACE option to prevent this.

So that you do not rename system (.SYS) files by accident, the system requires you to use the /SYSTEM option when you need to rename system files and you use a wildcard in a file type. To rename files that cover bad blocks (.BAD files), you must explicitly give the file name and file type of the specified .BAD file. Since .BAD files cover bad blocks on a device, you usually do not need to rename or otherwise manipulate these files.

Note that because of the file protection feature, you cannot execute any RENAME operations that result in deleting a protected file. For example, you cannot rename a file to the name of a protected file that already exists on the same volume.

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

[dd][:mmm[:yy]]

where:

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

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), the system uses the values from the current system date. For example, if you specify only the year :82 and the current system date is May 4, 1983, the system uses the date 4:MAY:82. 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.)

The options you can use with the RENAME command follow.

/BEFORE[:date] Use this option to rename 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 DY0: created before April 21, 1983:

```
.RENAME/BEFORE:21-APR-83 DY0:*.MAC DY0:*.BAK
 Files renamed:
DY0:A.MAC to DY0:A.BAK
DY0:B.MAC to DY0:B.BAK
DY0:C.MAC to DY0:C.BAK
```

/DATE[:date] Use this option to rename 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, 1982 to .BAK files:

```
.RENAME/DATE:20-MAR-82 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 Use this option to change the severity level of the error message that prints when not all of the input files you specified are found. If you do not use /INFORMATION, the system prints an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, the system prints 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 the system is unable to find DL0:FILE2.TXT, the system prints a message to inform you.

```
.RENAME/INFORMATION DL0:(FILE1,FILE2,FILE3).TXT
?PIP-I-File not found DL0:FILE2.TXT
```

/LOG This option lists on the terminal the files that were renamed by the current command. Normally, the system prints a log only if there is a wildcard in the file specification. If you specify /QUERY, the query messages replace the log (unless you specifically type /LOG/QUERY in the command line).

This example demonstrates logging.

```
.RENAME DY0:[PP,PAD  *.FOO)
 Files renamed:
DY0:ABC.MAC      to DY0:ABC.FOO
DY0:AAF.MAC      to DY0:AAF.FOO
```

/NOLOG This option prevents a list of the files that are renamed from appearing on the terminal.

/NEWFILES Use this option in the command line if you want to rename only those files that have the current date. This is a convenient way to access all new files after a session at the computer.

/PROTECTION Use this option to give a file protected status so that it cannot be deleted until you disable that status. Note that if a file is protected, you cannot delete it implicitly. For example, you cannot perform any operations on a file that result in deleting a protected file. You can change a protected file's name, but not its protected status, unless you also use the /NOPROTECTION option.

/NOPROTECTION Use this option to enable a file for deletion. This option disables a file's protected status.

/QUERY If you use this option, the system requests confirmation before it performs the operation. /QUERY is particularly useful on operations that involve wildcards, when you may not be sure which files the system selected for the operation.

You must respond to a query message by typing Y or any string beginning with Y, followed by a carriage return, to initiate execution of a particular operation. The system interprets any other response to mean NO; it does not perform the specific operation. The following example demonstrates querying.

```
.RENAME/QUERY DY0:[PIP,SAV  PIP.SAV)
 Files renamed:
DY0:PIP.SAV    to DY0: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:PROG3.BAK         to DK:PROG3.MAC ? Y
DK:LNLSA.BAK         to DK:LNLSA.MAC ?
DK:LNLS .BAK         to DK:LNLS .MAC ? Y
```

Note that if you specify /QUERY in a command line that also contains a wildcard in the file specification, the confirmation messages that print on the terminal replace the log messages that would normally appear.

/REPLACE This is the default mode of operation for the RENAME command. If a file exists with the same name as the file you specify for output, the system deletes that duplicate file when it performs the rename operation.

/NOREPLACE This option prevents execution of the rename operation if a file with the same name as the output file you specify already exists on the same device.

The following example uses /NOREPLACE. In this case, the output file already existed and no action occurs.

```
.RENAME/NOREPLACE DY0:TEST.SAV DY0:DIP.SAV
?RIP-W-Output file found, no operation performed DY0:TEST.SAV
```

/SETDATE[:date] This option causes the system 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, the system 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 DY0:[*,*]FOR *.OLD)
 Files renamed:
DY0:ABC.FOR     to DY0:ABC.OLD
DY0:A4F.FOR     to DY0:A4F.OLD
DY0:MERGE.FOR   to DY0:MERGE.OLD
```

/SINCE[:date] This option 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, 1983.

```
.RENAME/SINCE:24-FEB-83 *.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 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, the system files are excluded from the rename operation and a message is printed on the terminal to remind you of this.

This example renames all files on DY0: with the file name MM, including .SYS files, to MX files:

```
.RENAME/SYSTEM DY0:MM.* DY0:MX.*
 Files renamed:
DY0:MM.MAC              to DY0:MX.MAC
DY0:MM.OBJ              to DY0:MX.OBJ
DY0:MM.SAV              to DY0:MX.SAV
DY0:MM.SYS              to DY0:MX.SYS
```

/WAIT The /WAIT option is useful if you have a single-disk system. When you use this option, the system initiates the RENAME operation but then pauses and waits for you to mount the volume that contains the files you want to rename.

When the system pauses, it prints *Mount input volume in <device>; Continue?*. Mount the input volume and type Y or any string beginning with Y, followed by a carriage return, to continue the rename operation. Type N or any string beginning with N, or two CTRL/Cs, to abort the rename operation and return control to the keyboard monitor. Any other response causes the message to repeat.

After the system completes the rename operation, the system prints the following message prompting you to mount the system volume:

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

Mount the system volume and type Y or any string beginning with Y, followed by a carriage return. If you type any other response the system prompts you to mount the system volume until you type Y. When you use /WAIT, make sure that PIP is on the system volume.

The following command line renames PRIAM.TXT to NESTOR.TXT. PRIAM.TXT is on an RK05 disk.

```
.RENAME/WAIT/NOLOG RK0:PRIAM.TXT NESTOR.TXT
Mount input volume in RK0:; Continue? Y
Mount system volume in RK0:; Continue? Y
```

The RESET command resets several background system tables and does a general clean-up of the background area.

```
RESET
```

The RESET command accepts no options or arguments.

It causes the system to purge all open input/output channels, initialize the user program memory area, and release any device handlers that were not explicitly made resident with the LOAD command. It also disables CTRL/O, clears locations 40–53, resets the ring buffers, and resets the KMON (keyboard monitor) stack pointer.

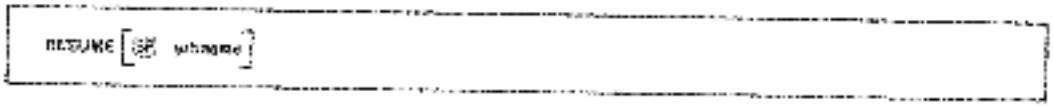
Use RESET before you execute a program if a device or the monitor needs reinitialization, or when you need to discard the results of previously issued GET commands. The RESET command has no effect on the foreground or system job.

The following example uses the RESET command before running a program.

```
.RESET
.R MYPROG
```

RESUME

The RESUME command continues execution of the foreground or system job from the point at which a SUSPEND command was issued.



If you have system job support enabled on your monitor, jobname represents the name of the foreground or system job you wish to resume. (The RESUME command accepts logical job names.) If you do not have system job support enabled on your monitor, do not include the name of the foreground job you wish to resume. When you issue the RESUME command, the foreground or system job enters any completion routines that were scheduled while the job was suspended. Note that RESUME is valid only with the FB and XM monitors.

The following command resumes execution of the foreground job that is currently suspended.

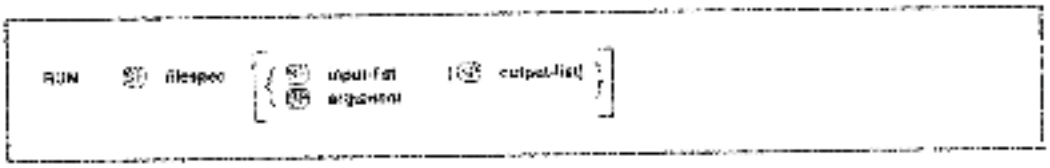
```
.RESUME
```

The next command resumes execution of the system job, QUEUE.SYS, that is currently suspended.

```
.RESUME QUEUE
```

You can also use the RESUME command to start a foreground job that you loaded with FRUN using /PAUSE. Likewise, you can use RESUME to start a system job that you loaded with SRUN using /PAUSE.

The RUN command loads a memory image file into memory and starts execution.



In the command syntax illustrated above, filespec represents the program to be executed. The system assumes a .SAV file type for the executable file, which can reside on any RT–11 block-replaceable device. The default device is DK:. When used to execute a virtual job, the RUN command automatically loads the device handler for the device you specify if it is not already resident. This eliminates the need to explicitly load a device handler when you run an overlaid program from a device other than the system device. The RUN command executes only those programs that have been linked to run as background jobs. (Use FRUN to execute foreground jobs under the FB or XM monitor.)

RUN is a combination of the GET and START commands. First it loads a memory image file from a storage device into memory. Then it begins execution at the program's transfer address.

You can use RUN to execute a privileged job under the XM monitor the same way you execute any other background job under the FB or SJ monitor. However, a virtual job under the XM monitor requires special preparation for execution. The RUN command creates a virtual memory partition for the job, creates a region 0 and window 0 definition block for the partition, and sets up the user mapping registers.

The following command executes MYPROG.SAV, which is stored on device DX1:.

```
.RUN DX1:MYPROG
```

You can also pass an argument in the RUN command to the program, or specify a list of input and output. This allows you to specify a line of input for a user program or for a system utility program (which accepts file specifications in the special syntax described in Chapter 1 of the *RT–11 System Utilities Manual*). The system automatically converts the input list and the output list you specify into a format that the Command String Interpreter (CSI) accepts. For example, to execute the directory program (DIR) and obtain a complete listing of the directory of DX1: on the printer, you can use the following command.

```
.RUN DIR DX1:*.* LP:/E
.
```

This command has the same effect as the following lines.

```
.RUN DIR
*LP:/E=DX1:*.*
*^Z
.
```

Note that when you use either an argument or an input list and output list with RUN, control returns to the monitor when the program completes.

The SAVE command writes memory areas in memory image format to the file and device that you specify.

```
SAVE dd: filespec [ /S: parameters]
```

In the command syntax shown above, filespec represents the file to be saved on a block-replaceable device. If you do not specify a file type, the system uses .SAV. The parameters represent memory locations to be saved.

Parameters are of the form:

address[-address(2)],address(3)[-address(n)]

where:

address — is an octal value representing a specific block of memory locations to be saved. If you specify more than one address, each address must be higher than the previous one.

RT–11 transfers memory in 256-word blocks, beginning on boundaries that are multiples of 256 (decimal). If the location(s) you specify make a block that is less than 256 words, the system saves additional words to make a 256-word block.

The system saves memory from location 0 to the highest memory address specified by the parameter list or to the program high limit (location 50 in the system communication area). Initially, the system gives the start address and the Job Status Word (JSW) the default value 0 and sets the stack to 1000. If you want to change these or any of the following addresses, you can use the Deposit command to alter them and the SAVE command to save the correct areas.

Area	Location
Start address	40
Stack	42
JSW	44
USR address	46
High address	50
Fill characters	56

If you change the values of the addresses, it is your responsibility to reset them to their default values. For more information concerning these addresses refer to the *RT–11 Programmer's Reference Manual*. Note that the SAVE command does not write the overlay segments of programs; it saves only the root segment. You cannot use the SAVE command for foreground or virtual jobs.

SAVE

The following command saves locations 10000 through 11777 and 14000 through 14777. It stores the contents of these locations in the file FILE1.SAV on device DK:.

```
.SAVE FILE1 10000-11000,14000-14100
```

The next example sets the reenter bit in the JSW and saves locations 1000 through 5777 in file FRAM.SAV on device SY:.

```
.D 44=2000
.SAVE SY:FRAM 1000-5777
```

The SET command changes device handler characteristics and certain system configuration parameters.

```
SET [/S] { physical-device-name: } [/C  condition[,...condition]]
         { item              }
```

In the command syntax illustrated above, physical-device-name represents the device handler whose characteristics you need to modify. See Table 3–1 in this manual for a list of the standard RT–11 permanent device names. The argument item represents a system parameter that you need to modify. The system items you can change include the default editor (SET EDIT), error handling (SET ERROR), program swapping upon exit (SET EXIT), IND and KMON handling of indirect command files and indirect control files (SET KMON), USR status (SET USR), and wildcard handling (SET WILD). Table 4–14 lists the devices and items you can modify, as well as the valid conditions for these devices and items.

If you set more than one condition for a device in a single SET command, separate the conditions with commas. With the exception of the SET TT, SET USR, and SET item commands, the SET command locates the file SY:device.SYS and permanently modifies it. The SET commands are valid for all three RT–11 monitors unless otherwise specified. They permanently modify the device handlers (except where noted); this means that the conditions remain set even across a reboot. For those SET commands that do not permanently modify the device handlers, the conditions return to the default setting after a reboot. To make these settings appear permanent, include the appropriate SET commands in your system's start-up indirect command file (see Section 4.4.3). The command you enter must be completely valid for the modification to take place. The SET command will modify only the device handler that corresponds to the currently booted monitor. For example, if you issue the SET command while running under the XM monitor, any device handlers modified will be of the form SYX.SYS.

NOTE

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.

The colon (:) after each device name is optional.

Figure 4–2: Format of a 12-Bit Binary Number (See SET CR: IMAGE command for accompanying text)

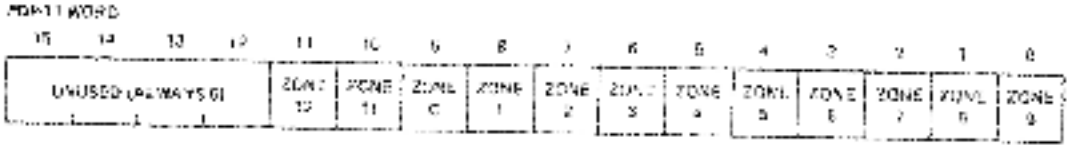


Table 4–14: SET Device Conditions and Modifications

Device or Item	Condition	Modification
CR:	CODE = n	Modifies the card reader handler to use either the DEC 026 or DEC 029 card codes. The argument n must be either 26 or 29. The default value is 29.
CR:	CRLF	Appends a carriage return/line feed combination to each card image. This is the normal mode.
CR:	NOCRLF	Transfers each card image without appending a carriage return/line feed combination. The default is CRLF.
CR:	HANG	Waits for you to make a correction if the reader is not ready at the start of a transfer. This is the normal mode.
CR:	NOHANG	Generates an immediate error if the device is not ready at the start of a transfer. The handler waits (regardless of how the condition is set) if the reader is not ready at some point during a transfer (that is, the input hopper is empty, but an end-of-file card has not been read). The default is HANG.
CR:	IMAGE	Causes each card column to be stored as a 12-bit binary number, one column per word. The CODE option has no effect in IMAGE mode. Figure 4–2 illustrates the format of the 12-bit binary number. This format allows the system to read binary card images. It is especially useful if you use a special encoding of punch combinations. Mark-sense cards can be read in this mode. The default is NOIMAGE.
CR:	NOIMAGE	Allows the normal translation (as specified by the CODE option) to take place. The system packs data one column per byte. It translates invalid punch combinations into the error character, ASCII backslash (\), which is octal 134. This is the normal mode.
CR:	TRIM	Removes trailing blanks from each card that the system reads. You should not use TRIM and NOCRLF together because card boundaries become difficult to read. TRIM is the normal mode.
CR:	NOTRIM	Transfers a full 80 characters per card. The default is TRIM.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
dd:	CSR = n	Modifies the device handler to use n as the Control and Status Register (CSR) address for the first controller. The valid range for n is 160000 to 177570 (octal). 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. This command is valid for the following handlers: MD: TU58 DL: RL01/02 DM: RK06/07 DU: RC25, RA80, RD51, RX50 DX: RX01 DY: RX02 LP: LS: RK: RK05 XL:
dd:	CSR2 = n	Modifies the device handler dd: to use n as the CSR address for the second controller. This option is valid only if you create the dd: dual controller handler through system generation). This command is valid for the following devices: DD: DX DY: DZ:
dd:	RETRY = n	Allows you to change the number of times a device handler attempts to recover from an error when the Error Logger is running. The value n must be an integer in the range 1 through 8. The default value for n is 8. The variable dd: represents the device mnemonic of any device that the Error Logger supports: DD: DL: DM: DU: DW: DX: DY: DZ: RK:

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
dd:	SUCCES	Allows you to choose to log successful I/O transfers as well as errors when the error logger is running. This is the default mode. The variable dd: represents the device mnemonic of any device that the Error Logger supports: DD: DL: DM: DU: DW: DX: DY: DZ: RK:
dd:	NOSUCCES	Allows you to choose not to log successful I/O transfers when the error logger is running. The default mode is SET dd: SUCCES. The variable dd: represents the device mnemonic of any device that the error logger supports: DD: DL: DM: DU: DW: DX: DY: DZ: RK:
dd:	VECTOR=n	Modifies the device handler to use n as the vector address for the first controller. The valid range for n is 100 to 474 (octal). This option enables you to set a special vector value in the device handler without having to modify and reassemble the handler source code. This command is valid for the following handlers: DD: TU58 DL: RL01/02 DM: RK06/07 DU: RC25, RA80, RD51, RX50 DX: RX01 DY: RX02 LP: LS: RK: RK05 XL:

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
dd:	VEC2 = n	Modifies the device handler dd: to use n as the vector for the second controller. This option is valid only if you create the dd: dual controller handler (through system generation). This command is valid for the following devices: DD: DL: DM: DU: DX: DY: RK:
DU:	CSR3 = n	Modifies the DU: device handler to use n as the CSR address for the third controller. This option is valid only if you create the third DU: controller handler (through system generation).
DU:	CSR4 = n	Modifies the DU: device handler to use n as the CSR address for the fourth controller. This option is valid only if you create the fourth DU: controller handler (through system generation).
DUn:	PART = x	Defines the partition of a disk on which device unit n resides. The variable x is an integer in the range 0–255, depending on the size of the disk device (each partition is 64K blocks). The default for x is 0. (See the *RT–11 Software Support Manual* for more information on MSCP disk partitioning.)
DUn:	PORT = x	Defines which port to access when device unit n is specified. The variable x is an integer in the range 0–3. (See the *RT–11 Software Support Manual* for more information on using multiple ports with MSCP devices.)
DUn:	UNIT = x	Defines which unit plug number to access when device unit n is specified. The variable x is an integer in the range 0–251.
DU:	VEC3 = n	Modifies the DU: device handler to use n as the vector for the third controller. This option is valid only if you create the third DU: controller handler (through system generation).
DU:	VEC4 = n	Modifies the DU: device handler to use n as the vector for the fourth controller. This option is valid only if you create the fourth DU: controller handler (through system generation).
DW	WCHECK	Verifies output to RD50 or RJ51 disks by reading data after writing it to the disk.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
DW	NOWCHECK	Does not verify output to RD50 and RD51 disks.
DW	WRITE	Write-enables RD50/RD51 drive unit 0.
DW	NOWRITE	Write-locks RD50/RD51 drive unit 0.
DXn:	WRITE	Write-enables RX01 drive unit n. The condition remains set across a reboot unless DXn is the system device when you issue the command. The default value for n is 0.
DXn	NOWRITE	Write-protects RX01 drive unit n. This condition remains set across a reboot unless DXn is the system device when you issue the command. The default value for n is 0.
DYn:	WRITE	Write-enables RX02 drive unit n. This condition remains set across a reboot unless DYn is the system device when you issue the command. The default value for n is 0.
DYn:	NOWRITE	Write-protects RX02 drive unit n. This condition remains set across a reboot unless DYn is the system device when you issue the command. The default value for n is 0.
EDIT	EDIT	Invokes the text editor EDIT with the keyboard monitor EDIT command. This is the normal mode. The system returns to this condition after a reboot.
EDIT	KED	Invokes the keypad editor (KED) with the keyboard monitor EDIT command. For more information on the keypad editor, see the *PDP-11 Keypad Editor User's Guide*. This condition is valid only for VT100-compatible terminals. The system returns to EDIT after a reboot.
EDIT	KEX	Invokes the virtual form of the keypad editor (KEX) with the keyboard monitor EDIT command. KEX runs only as a background job, and only under the XM monitor. Otherwise, you use KEX just as you would KED. See the *PDP-11 Keypad Editor User's Guide* for instructions on how to use KED. This condition is valid only for VT100-compatible terminals. The system returns to EDIT after a reboot.
EDIT	K52	Invokes the keypad editor (K52) with the keyboard monitor EDIT command. This condition is valid only if your terminal is a VT52. For more information on the keypad editor, see the *PDP-11 Keypad Editor User's Guide*. The system returns to EDIT after a reboot.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
EDIT	TECO	Invokes the text editor TECO with the keyboard monitor EDIT command. The default is EDIT. The system returns to that condition after a reboot.
EL:	LOG	Used when running the Error Logger under the SJ monitor. 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. The system returns to SET EL: NOLOG after a reboot.
EL:	NOLOG	Used when running the Error Logger under the SJ monitor. Turns off the Error Logger. This is the default condition.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
EL:	PURGE	Used when running the Error Logger under the SJ monitor. Discards the contents of the internal Error Logger buffer. This command is valid only if the Error Logger has been enabled with the SET /ELLOG command.
ERROR	ERROR	Causes indirect command files and keyboard monitor commands that perform multiple operations (such as EXECUTE, which combines assembling, linking, and running) to abort if errors or severe or fatal errors occur. These errors produce error messages that contain the severity codes -E-, -F-, and -U-. This setting causes indirect files and keyboard monitor commands to abort on MACRO assembly errors. An example of an error is an undefined symbol in an assembly. An example of a severe error is a device that is write-locked when the system attempts to write to it. If either condition occurs, the indirect command file or keyboard monitor command aborts the next time the monitor gets control of the system. This is the normal setting. The system returns to this condition after a reboot.
ERROR	FATAL	Causes indirect command files and keyboard monitor commands to abort only if severe or fatal errors occur. These errors produce error messages that contain the severity codes -E-, -F-, and -U-. See SET ERROR ERROR, which is the default setting. The system returns to that condition after a reboot.
ERROR	NONE	Allows indirect command files and keyboard monitor commands to continue to execute even though they contain significant errors. Most monitor fatal errors still cause the indirect command file or keyboard monitor command to abort. Fatal errors that always abort indirect command files contain the -U- characters in the error messages. See SET ERROR ERROR, which is the default setting. The system returns to that condition after a reboot.
ERROR	SEVERE	Causes indirect command files and keyboard monitor commands to abort only if severe or fatal errors occur. These errors produce error messages that contain the severity codes -F- and -U-. See SET ERROR ERROR, which is the default setting. The system returns to that condition after a reboot.
ERROR	WARNING	Causes indirect command files and keyboard monitor commands to abort if warnings, errors, or severe or fatal errors occur. These errors produce error messages that contain the severity codes -W-, -E-, -F-, and -U-. See SET ERROR ERROR, which is the default setting. The system returns to that condition after a reboot.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
EXIT	SWAP	When a program terminates, causes any portion of the program that resides in SWAP.SYS to be written back into memory. This is the default setting.
EXIT	NOSWAP	When a program terminates, prevents any portion of the program that resides in SWAP.SYS from being written back into memory. This may prevent you from being able to reenter a program; however, it allows considerably better performance when using slower media (such as TU58, RX01, and RX02). The default setting is SET EXIT SWAP.
KMON	IND	Causes IND to execute a file specified in the command @filespec as an IND control file. Causes KMON to execute a file specified in the command $@filespec as an indirect command file. The default setting is SET KMON NOIND.
KMON	NOIND	Causes KMON to execute a file specified in the command @filespec as an indirect command file. If you try to execute an indirect control file, an error occurs. This is the default setting.
LD	CLEAN	Verifies and corrects, if necessary, all current logical disk assignments by checking them against the files or volumes that are mounted. 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 keyboard commands SQUEEZE and BOOT automatically perform the SET LD CLEAN operation.
LDn:	WRITE	Used during disk subsetting; defines logical disk unit n as being write-enabled (read/write access allowed). The value n must be an integer in the range 0 through 7.
LDn:	NOWRITE	Used during disk subsetting; defines logical disk unit n as being write-locked (read-only access allowed). The value n must be an integer in the range 0 through 7.
LP:	CR	Sends carriage returns to the printer. To allow overstriking on the printer, use this condition for any FORTRAN program that uses formatted input and output. Use CR also for any LN01 or LP05 line printer to prevent loss of the last line in the buffer. LP:NOCR is the normal mode.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
LP:	NOCR	Prevents the system from sending carriage returns to the printer. This setting produces a significant increase in printing speed on LP11 printers, where the line printer controller causes a line feed to perform the functions of a carriage return. This is the default setting.
LP:	CSR = n	Modifies the line printer handler to use n as the Control and Status Register (CSR) address for the line printer controller. The value you supply must be an octal word address not less than 160000. This option enables you to set a special CSR value in the line printer handler itself, without having to modify and reassemble the handler source code. Use this option if you have installed the line printer controller at a nonstandard address.
LP:	CTRL	Passes all characters, including nonprinting control characters, to the printer. Use this condition to pass the bell character to the LA180 printing terminal. You can use this mode for LS11 line printers. (Other line printers print a space for a control character.) The default is NOCTRL.
LP:	NOCTRL	Ignores nonprinting control characters. This is the normal mode.
LP:	FORM	Declares that the line printer has hardware form feeds, causing the line printer handler to send form feeds to the controller. When you use this option, the line printer handler sends the form feed character to the printer each time the handler encounters a form feed. This is the default setting.
LP:	NOFORM	Causes the line printer handler to simulate hardware form feeds by sending one or more line feeds to the printer. When you use this setting, you must also use the LENGTH = n setting and position the paper at the top of a form (that is, at the page perforation) before you start to use the printer. Using the NOFORM condition is useful if you are using a preprinted form that has a nonstandard length. You must use this setting if your printer does not accommodate form feeds. FORM is the default setting.
LP:	FORM0	Issues a form feed before a request to print block 0. This is the normal mode.
LP:	NOFORM0	Turns off FORM0 mode, which is the default.
LP:	HANG	Waits for you to make a correction if the line printer is not ready or is not ready at some point during printing. If you expect output from the line printer and the system does not respond or appears to be idle, check to see if the line printer is powered on and ready to print. This is the normal mode.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
LP:	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.
LP:	LC	Allows the system to send lowercase characters to the printer. Use this condition if your printer has a lowercase character set. The default is NOLC.
LP:	NOLC	Translates characters in lowercase to uppercase before printing. This is the normal mode.
LP:	LENGTH=n	Causes the line printer to use n as the number of lines per page. The default number of lines per page is 66. Use this option with the NOFORM and SKIP=n settings.
LP:	SKIP=n	Causes the line printer handler to send a form feed to the printer when it comes within n lines of the bottom of a page. Use this setting to prevent the printer from printing over page perforations. The value you supply for n should be an integer from 0 to the maximum number of lines on the paper. If you set SKIP=0, the handler sends lines to the printer regardless of the position of the paper. If you have set SKIP to a value other than 0, set SKIP=0 to disable this condition. When you use this setting, you must also use the LENGTH=n setting. The default is SKIP=0.
LP:	TAB	Sends TAB characters to the line printer. The default is NOTAB.
LP:	NOTAB	Expands TAB characters by sending multiple spaces to the line printer. This is the normal mode.
LP:	VECTOR=n	Modifies the line printer handler to use n as the vector of the line printer controller. The value you supply for n must be an even octal address below 500. This option enables you to set a special vector value in the line printer handler itself, without having to modify the handler source code and reassemble. Use this option if you have installed the line printer controller at a nonstandard address.
LP:	WIDTH=n	Sets the line printer width to n, where n is a decimal integer between 30 and 255 inclusive. The system ignores any characters that print past column n. The default is 132.
LS:	CR	Sends carriage returns to the printer. To allow overstriking on the printer, use this condition for any FORTRAN program that uses formatted input and output. (Use CR also for any LS11 or LP05 line printer to prevent loss of the last line in the buffer.) This is the normal mode.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
LS:	NOCR	Prevents the system from sending carriage returns to the printer. This setting may produce a significant increase in printing speed on some line printers, where the printer controller causes a line feed to perform the functions of a carriage return. The default is Off.
LS:	CSR=n	Modifies the line printer handler to use n as the Control and Status Register (CSR) address for the printer controller. The value you supply for n must be an octal word address not less than 160000. This option enables you to set a special CSR value in the printer handler itself, without having to modify the handler source code and reassemble. Use this option if you have installed the printer controller at a nonstandard address.
LS:	CTRL	Passes all characters, including nonprinting control characters, to the printer. Use this condition to pass the bell character to the LA180 printing terminal. The default is NOCTRL.
LS:	NOCTRL	Ignores nonprinting control characters. This is the normal mode.
LS:	FORM	Declares that the line printer has hardware form feeds, causing the line printer handler to send form feeds to the controller. When you use this option, the line printer handler sends the form feed character to the printer each time the handler encounters a form feed. This is the default setting.
LS:	NOFORM	Causes the line printer handler to simulate hardware form feeds by sending one or more line feeds to the printer. When you use this setting, you must also use the LENGTH=n setting and position the paper at the top of a form (that is, at the page perforation) before you start to use the printer. Using the NOFORM condition is useful if you are using a preprinted form that has a nonstandard length. You must use this setting if your printer does not accommodate form feeds. FORM is the default setting.
LS:	FORM0	Issues a form feed before a request to print block 0. This is the normal mode.
LS:	NOFORM0	Turns off FORM0 mode. The default is FORM0.
LS:	HANG	Waits for you to make a correction if the line printer is not ready or becomes not ready during printing. If you expect output from the printer and the system does not respond or appears to be idle, check to see if the printer is powered on and ready to print. This is the normal mode.
LS:	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). The default setting is HANG.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
LS:	LC	Allows the system to send lowercase characters to the printer. Use this condition if your printer has a lowercase character set. This is the normal mode.
LS:	NOLC	Translates lowercase characters to uppercase before printing. The default is LC.
LS:	LENGTH=n	Causes the printer to use n as the number of lines per page. The default number of lines per page is 66. Use this option with the NOFORM and SKIP=n settings.
LS:	SKIP=n	Causes the line printer handler to send a form feed to the printer when it comes within n lines of the bottom of a page. Use this setting to prevent the printer from printing over page perforations. The value you supply for n should be an integer from 0 to the maximum number of lines on the paper. If you set SKIP=0, the handler sends lines to the printer regardless of the position of the paper. If you have set SKIP to a value other than 0, set SKIP=0 to disable this condition. When you use this setting, you must also use the LENGTH=n setting. The default is SKIP=0
LS:	SPEED=n	Sets the printer to run at baud n, where n can be any of the following baud rates: 50 150 1200 3600 75 300 1800 4800 110 600 2000 9600 134 600 2400 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.
LS:	TAB	Sends TAB characters to the printer. The default is NOTAB.
LS:	NOTAB	Expands TABs by sending multiple spaces to the printer. This is the normal mode.
LS:	VECTOR=n	Modifies the printer handler to use n as the vector of the line printer controller. The value you supply for n must be an even octal address below 500. This option enables you to set a special vector value in the line printer handler itself, without having to modify the handler source code and reassemble. Use this option if you have installed the printer controller at a nonstandard address.
LS:	WIDTH=n	Sets the printer to width n, where n is a decimal integer between 30 and 255 inclusive. The system ignores any characters that print past column n. The default is 132.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
MM:	DEFAULT=9	Returns to default settings for 9-track tape. The 9-track defaults are: DENSE=809 ODDPAR NODUMP
MM:	DENSE=[800 or 809 or 1600]	Sets density for the 9-track tape handler. Do not alter the density setting within a volume. A density setting of 1600 bits/in automatically sets parity to odd. The valid density settings for 9-track tape are: 800 bits/in 1600 bits/in

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
MM:	ODDPAR	Sets parity to odd for 9-track tape. DIGITAL recommends this setting.
MM:	NOODDPAR	Sets parity to even for 9-track tape. DIGITAL does not recommend this setting for normal operation, and provides it only for compatibility with other systems.
MT:	DEFAULT = 9	Returns to default settings for 9-track tape: DENSE = 800 ODDPAR
MT:	DENSE = 800 or 809	Sets density for 9-track tape. Settings 800 and 809 are the only valid settings for 9-track tape. Thus, the valid density setting is: 9-track: 800 or 809 = 800 bits/in **NOTE** These SET command options apply to all units of the magtape controller. Six-bit mode and core dump mode are described in the *RT–11 Software Support Manual*.
MT:	ODDPAR	Sets parity to odd for 9-track tape. DIGITAL recommends this setting.
MT:	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.
SL:	ASR	Allows the single-line editor to determine the type of terminal you are using, so SL can use the proper escape sequences. SL prints on the console the type of terminal 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 prints an error message.
SL:	LEARN	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 console 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. This command is valid for VT100-compatible terminals only.
SL:	NOLEARN	Unlocks the help display and allows it to scroll off the screen, so you can use the entire screen to display console input and output. This is the default setting. The SET SL: OFF command performs an automatic SET SL: NOLEARN command.
SL:	ON	Loads and enables the single-line editor.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
SL:	OFF	Unloads and disables the single-line editor.
SL:	SYSGEN	Converts the system generation characteristics of SL.SYS (under the SJ and FB monitors) or SLX.SYS (under the XM monitor) to match those of the current monitor without requiring you to reassemble SL.
SL:	TTYIN	Enables you to edit responses to prompts printed by the system utilities. When SET SL: TTYIN is enabled, the prompt prints 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.)
SL:	NOTTYIN	Disables your ability to edit responses to prompts printed by the system utilities. When SET SL: NOTTYIN is enabled, the prompt and your response appear on the same line. (This command prevents SL from intercepting and editing input requests from .TTYIN. SL still intercepts and edits input requests from .CSIGEN, .CSISPN, and .GTLIN.) This is the default mode.
SL:	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: ANSI instead of this command. This command supports the following terminals: VT52 (SET SL: VT52) VT62 (SET SL: VT62) VT100 (SET SL: VT100) VT101 (SET SL: VT101) VT102 (SET SL: VT102) The default setting is SET SL: VT100.
SL:	WIDTH = n	Allows you to set the width of the terminal. The variable n represents the maximum number of characters on a single line on the terminal. 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 = 80, and the prompt consists of only the keyboard monitor prompt (.), then the maximum number of characters you can type as input on one line is: 80 - 1 - 1 = 48 characters The default value for n is 79.

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
SP	FLAG=n	Sets the number of flag pages to generate whenever SPOOL begins printing a file. The default for n is 0. The largest value for n is 4.
SP	FORM0	Issues a form feed on the output device each time SPOOL encounters block 0 of a file to be printed. Useful if the output device handler does not support its own FORM0 option. The default mode is NOFORM0.
SP	NOFORM0	Turns off FORM0 mode. This is the default mode.
SP	KILL	Removes all currently spooled output from SPOOL's work file.
SP	NEXT	Stops sending output from the current file, discards the remaining spooled output for that file, and begins sending output from the next listing in SPOOL's work file
SP	WAIT	Suspends sending output from SPOOL's work file to the output device, but does not delete anything from the work file. SPOOL continues to accept input with SET SP WAIT in effect.
SP	NOWAIT	Resumes sending spooled output suspended by the command SET SP WAIT.
SP	WIDE	Causes SPOOL to generate 132-column flag pages.
SP	NOWIDE	Causes SPOOL to generate 80-column flag pages.
TT:*	CONSOL=n	Directs the system to use the terminal whose logical unit number you specify as the console terminal. The terminal whose logical unit number you specify must not be currently attached by the foreground or any system job. To use this setting, you must have a multiterminal configuration. The system returns to this default after a reboot. You cannot use this setting for a remote line.
TT:*	CRLF	Issues a carriage return/line feed combination on the console terminal whenever you attempt to print past the right margin. You can change the margin with the WIDTH command. This is the normal mode. This setting is invalid with a non-multiterminal SJ monitor.
TT:*	NOCRLF	Takes no special action at the right margin. This setting is invalid with a non-multiterminal SJ monitor. The default; the system returns to this condition after a reboot.
TT:*	FP	Treats CTRL/B and CTRL/P (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. This is the normal mode. This setting is not valid for the SJ monitor. The system returns to this condition after a reboot.

*SET TERM can be substituted for SET TT:

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
TT:*	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. This setting is not valid for the SJ monitor. The default is FB. The system returns to that condition after a reboot.
TT:*	FORM	Indicates that the console terminal is capable of executing hardware form feeds. This setting is invalid with a non-multiterminal SJ monitor.
TT:*	NOFORM	Simulates form feeds by generating eight line feeds. This setting is not valid for the non-multiterminal SJ monitor. This is the normal mode. The system returns to this condition after a reboot.
TT:*	HOLD	Enables the hold screen mode of operation for the VT50, VT52, and VT61 terminals. The command has no effect on any other terminals, but it can cause a left square bracket ([) to print. This setting is valid for all monitors. NOHOLD is the default setting. The system returns to that condition after a reboot.
TT:*	NOHOLD	Disables the hold screen mode of operation for the VT50 terminal. The command has no effect on any other terminal, but it can cause a backslash (\) to print. This setting is valid for all monitors. The default is NOHOLD. The system returns to that condition after a reboot.
TT:*	PAGE	Treats CTRL/S and CTRL/Q characters as terminal output hold and unhold flags and does not transmit them to your program. You must use this setting if you are using a VT100 terminal. This setting is not valid for the non-multiterminal SJ monitor. This is the normal mode. The system returns to this condition after a reboot.
TT:*	NOPAGE	Causes CTRL/S and CTRL/Q to have no special meaning. This setting is not valid for the non-multiterminal SJ monitor. The default is PAGE. The system returns to that condition after a reboot.
TT:*	QUIET	Prevents the system from echoing lines from indirect files. The default is NOQUIET. The system returns to that condition after a reboot.
TT:*	NOQUIET	Echoes lines from indirect files. This is the default mode. The system returns to this condition after a reboot.

*SET TERM can be substituted for SET TT:

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
TT:*	SCOPE	Echoes DELETE or RUBOUT characters as backspace-space-backspace. Use this mode if your console terminal is a VT50, VT05, VT52, VT55, VT61, VT100, or if G1 ON is in effect. The default, the system returns to this condition after a reboot. Note that you delete TAB characters by typing a single RUBOUT or DELETE, even though the cursor does not move back the correct number of spaces. This is a restriction in SCOPE modes.
TT:*	NOSCOPE	Echoes DELETE or RUBOUT characters by enclosing the deleted characters in backslashes. SCOPE is the default.
TT:*	TAB	Indicates that the console terminal is capable of executing hardware tabs. This setting is not valid for the non-multiterminal SJ monitor. The default is NOTAB. The system returns to that condition after a reboot.
TT:*	NOTAB	Simulates tab stops every eight positions. Many terminals supplied by DIGITAL have hardware tabs. This setting is not valid for the non-multiterminal SJ monitor. This is the normal mode. The system returns to this condition after a reboot.
TT:*	WIDTH – n	Sets the terminal width to n, where n is an integer between 30 and 255. The system initially sets the width to 80. This setting is not valid for the non-multiterminal SJ monitor. (See SET TT CRLF). The system returns to 80 after a reboot.
USR	SWAP	Allows the background job to place the User Service Routine in a swapping state. This setting is not valid for the XM monitor. This is the normal mode for FB and SJ monitors. The system returns to this condition after a reboot.
USR	NOSWAP	Prevents the background job from placing the User Service Routine in a swapping state. This setting is not valid for the XM monitor. The default is SWAP for FB and SJ monitors. The system returns to that condition after a reboot.
VM	BASE – nnnnnn	Allows you to select the location in memory where block 0 of a virtual disk will begin (the base address). 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 1002500 (octal), specify 100256 for nnnnnn. The default value for nnnnnn is 1600 (octal) under the SJ and FB monitors, and 10000 (octal) under the XM monitor. (The address 160000 is the division between 18- and 22-bit addresses.)

*SET TERM can be substituted for SET TT:

(Continued on next page)

Table 4–14: SET Device Conditions and Modifications (Cont.)

Device or Item	Condition	Modification
WILD	EXPLICIT	Causes the system to recognize file specifications exactly as you type them. If you omit a file name or a file type in a file specification the system does not automatically replace the missing item with an asterisk (*). Wildcards are described in Section 4.2 of this manual. The default is IMPLICIT. The system returns to that condition after a reboot.
WILD	IMPLICIT	Causes the system to interpret missing fields in file specifications as asterisks (*). Wildcards are described in Section 4.2 of this manual. Table 4–2 shows how the system interprets commands that have missing fields. This is the normal mode. The system returns to this condition after a reboot.
XC	SPEED = n	Sets the communication port to run at baud n, where n can be any of the following baud rates: 50 150 1200 3600 75 200 1800 4800 110 300 2000 9600 134 600 2400 19200 You must specify a value for n in this command. If you do not use this command, the communication port runs at 1200 baud.

The following examples illustrate the SET command. This command allows the system to send lowercase characters to the printer:

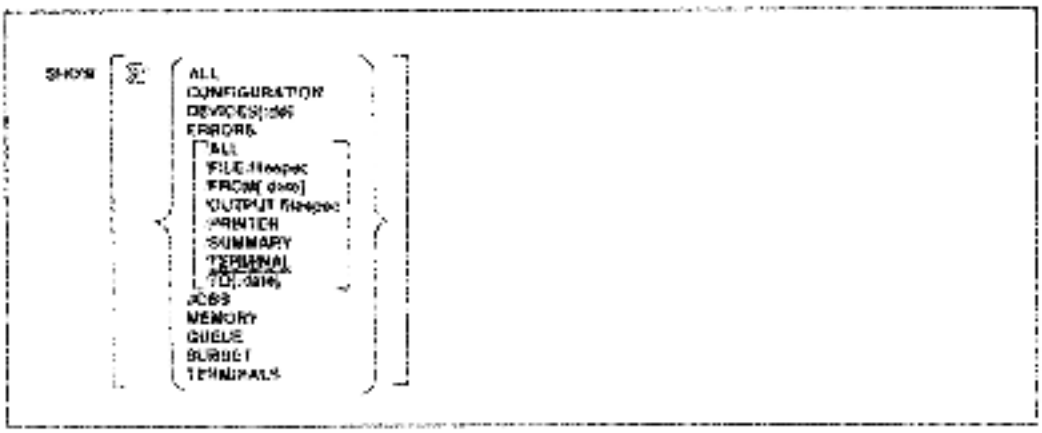
```
.SET LP LC
```

The next command sets the system wildcard default to implicit.

```
.SET WILD IMPLICIT
```

As a result of this command the system inserts an asterisk (*) in place of a missing file name or file type in a file specification. See Table 4–2 for a list of these commands.

The SHOW command prints information about your RT–11 system on the console terminal.



The information includes 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, terminal characteristics for terminals currently active on a multiterminal system, logical disk subsetting, and device handler status. If you are running the Error Logger, QUEUE, or SPOOL, the SHOW command can provide information on errors and the update status of files waiting to be sent to an output device.

If you specify SHOW without an option, SHOW displays your system's device assignments. The devices the system lists are those known by the RT–11 monitor currently running. This list reflects any additions or deletions you have made with the INSTALL and REMOVE commands. The listing also includes additional information about particular devices. The informational messages and their meanings are:

Message	Indicated Condition
(RESORC) or =RESORC	The device or unit is assigned to the background job RESORC (for FB and XM monitors only).
(F) or =F	The device or unit is assigned to the foreground job (for FB and XM monitors only and monitors without system job support).
(jobname) or =jobname	The device or unit is assigned to the system or foreground job (for FB and XM monitors that have system job support), where jobname represents the name of the system or foreground job.

(Loaded)	The handler for the device has been loaded into memory with the LOAD command.
(Resident)	The handler for the device is included in the resident monitor.
=logical-device-name(1), logical-device-name(2)... ,logical-device-name(n)	The device or unit has been assigned the indicated logical device names with the ASSIGN command.
xx free slots	The last line tells the number of unassigned, or free, device slots.

The following example was created under an FB monitor that has system job support. It shows the status of all devices known to the system:

```
.SHOW
TT   (Resident)
RK   (Resident)
  RK1:   = SY, DK, OBJ, SRC, BIN
  RK2:   = LST, MAP
MQ   (Resident)
DL   (Loaded)
DM
DX   (Loaded)
  DX0:   (MYPROG)
LP:  (Loaded-QUEUE)
MT
5 free slots
```

The listing shows first that TT, MQ, and RK are resident in memory. The other device handlers known to the system are MQ, DL, DM, DX, LP, and MT. There are five free slots in the table. RK0: has the logical names SY, DK, OBJ, SRC, and BIN. RK1: has the logical names LST and MAP. The DX handler is loaded and device DX0: belongs to the foreground job, MYPROG. The LP: handler is loaded and belongs to the system job, QUEUE.

The options for the SHOW command follow.

ALL This option is a combination of CONFIGURATION, DEVICES, device assignments (SHOW command with no option), JOBS, TERMINALS, MEMORY, and SUBSET in that order.

CONFIGURATION This option 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.

First, the listing always shows the version number and update level of the currently running monitor.

Next, information about the monitor is displayed. The first line indicates the device from which the system was bootstrapped, and the next line indicates whether or not 22-bit addressing is on if you are running the XM monitor, followed by whether any foreground jobs are loaded. Then the listing shows whether the user service routine (USR) is set to SWAP or NOSWAP, whether EXIT is set to SWAP or NOSWAP, whether KMON is set to IND or NOIND, whether TT is set QUIET or NOQUIET, the severity level to which ERROR is set, whether SL is set ON or OFF, and the default editor. The indirect file nesting prints out as a decimal number, followed by whether global .SCCA support is enabled or disabled.

Next, the listing displays the system hardware configuration. It lists the processor type, which can be one of the following:

LSI 11	PDP 11/15,20	PDP 11/23 PLUS
MICRO/PDP–11	PDP 11/24	PDP 11/35,40
PC325/PC350	PDP 11/34	PDP 11/44
PDT 130/150	SBC 11/21	PDP 11/45,50,55
PDP 11/04	SBC 11/21 PLUS	PDP 11/60
PDP 11/05,10	PDP 11/23	PDP 11/70

Then, the total amount of memory your system contains is displayed; for example:

56KB of memory

A separate line prints out for each of the following items that is present on your system:

FP11 Hardware Floating Point Unit
Commercial Instruction Set (CIS)
Extended Instruction Set (EIS)
Floating Instruction Set (FIS)
KT11 Memory Management Unit
Parity Memory
Cache Memory
60 Cycle System Clock

If you have graphics hardware (VT11 or VS60), another line is printed out to indicate it. The clock frequency (50 or 60 cycles) prints next, followed by a line for the KW11–P programmable clock, if there is one on your system.

Finally, the listing either shows that there are no special features in effect, or it lists the appropriate features from the following list:

Device I/O time-out support
Error logging support
Multi-terminal support
Memory parity support
SJ timer support
System job support

The following example was created on a PDP 11/23 processor:

```
>SHOW CONFIGURATION

RT-11FB    V05.xx
Booted from DL0:F11FB

USR is set SWAP
EXIT is set SWAP
XMON is set IND
TT is set NOQUIET
ERROR is set ERROR
SL is set OFF
EDIT is set EDIT
XMON nesting depth is 0
Global >3000 flag is disabled

PDP 11/23
1024KB of memory
Extended Instruction Set (EIS)
KT11 Memory Management Unit
Parity Memory
60 Cycle System Clock
```

DEVICES[dd] This option displays the RT-11 device handlers and their status, CSR addresses, and vectors. You can obtain this information for a specific device by including the optional argument dd. The variable dd represents the two-letter permanent device mnemonic.

The messages for handler status are as follows:

Installed
Not installed
-Not installed (handler special features do not match those of the monitor)
nnnnnn (load address of handler)
Resident

The following example illustrates SHOW DEVICES.

```
>SHOW DEVICES

DEVICE     STATUS              CSR      VECTOR(S)

DY         122620              177170   264
DD         Installed           176700   300 304
DL         Installed           174400   160
DX         Not installed       177170   264
LS         Installed           176500   300 304
LP         Installed           177514   200
MS         Installed           172522   224 300
DU         Installed           172150   154
MU         Installed           000000   000
LD         Installed           000000   000
DM         Installed           177640   210
VM         Installed           177572   000
RD         Resident            177400   220
SL         Not installed       000000   000
MT         Installed           172520   224
MM         Not installed       172440   774
```

Because of its special format, the TT handler is never listed.

ERRORS The SHOW ERRORS command is valid only if you have error logging enabled on your monitor. For a complete description of the Error Logger and directions on how to start it, see Chapter 16 of the *RT–11 System Utilities Manual*, Error Logging. 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.

The SHOW ERROR command invokes ERROUT, one of the programs in the error logging package. ERROUT runs as a background job under the FB and XM monitors, and as the only job under the SJ 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 errors that occurred. For complete descriptions of the reports ERROUT creates, see Chapter 16 of the *RT–11 System Utilities Manual*, Error Logging.

ERRORS ⏎	Prints a full report on each I/O transfer that has occurred in addition to each I/O, memory parity, and cache memory error that has occurred.
ERRORS/ALL	Same as SHOW ERRORS ⏎
ERRORS/FILE:filespec	Prints a full I/O transfer and error report from the file you specify. The file you specify must be of the same format that the error logger uses for its statistical compilations.
ERRORS/FROM:date	Prints a full I/O transfer and error report for errors that occurred starting from the date you specify. Enter the date as dd-mmm-yy, where: dd is a two-digit day mmm is the first three characters of a month name yy is the last two digits of a year
ERRORS/TO:date	Prints a full I/O transfer and error report for errors that occurred up to the date you specify.

ERRORS/OUTPUT:filespec	Enters the I/O transfer and error report in the output file you specify. This is useful if you want to save the error logging reports.
ERRORS/PRINTER	Prints the I/O transfer and error report at the line printer.
ERRORS/SUMMARY	Prints a summary error report at the terminal. The summary error report lists only the errors that occurred, not the successful I/O transfers.
ERRORS/TERMINAL	Prints the I/O transfer and error report at the terminal. /TERMINAL is the default setting.

JOBS This option displays data about the jobs that are currently loaded. This option also tells the following:

- The job name and number (if you have not enabled system job support on your monitor, the foreground job name appears as FORE, and its priority is 1)
- The console the job owns (with a non-multiterminal monitor, this space is blank)
- The priority level of the job
- The job's running state (running, suspended, or done but not unloaded)
- The low and high memory limits of the job
- The start address of the job's impure area

The example that follows displays data about currently running jobs:

```
.SHOW JOBS

JOB    NAME    CONSOLE   LOW-FL   STATE    LOW      HIGH     IMPURE
 1     QUEUE     0         0      SUSPEND  115274   130200   115254
 0     RESUNC    0         0      RUN      000000   126110   102344
```

MEMORY The SHOW MEMORY command lists the organization of physical memory. The listing displays such information as where jobs are loaded, where devices handlers are loaded, where in memory KMON and the USR will reside, and the number of words of memory each occupies. Memory addresses are displayed in octal.

If you are running under the XM monitor, the SHOW MEMORY listing is divided into two sections, the first for extended memory and the second for kernal memory.

The following example displays the organization of physical memory when running under the SJ monitor.

```
.SHOW MEMORY

Address   Module    Words

160000    IOPAGE     4096.
157400    RK          120.
127274    RMON       6170.
126112    DY          313.
001000    ..BG..    21787.
```

The next example shows the organization of physical memory when running under the XM monitor.

```
.SHOW MEMORY

     Extended Memory

Address     Module      Words

01000000    VM        983216.
00160000    ......    102400.

     Kernel Memory

Address     Module      Words

160000      IOPAGE      4096.
157350      RK           140.
128144      RMON        6970.
122612      DY           350.
111610      USR         2905.
001000      ..BG..     16220.
```

QUEUE Use the SHOW QUEUE command to get 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*). The QUEUE listing shows the output device, job name, input files, job status, and number of copies for each job that is queued. The SPOOL listing shows whether the SPOOL 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 sample command line that follows lists the current contents of the queue for QUEUE.

```
.SHOW QUEUE
DEVICE      JOB       STATUS      COPIES      FILES

LP0:        LARY        P           1         PASS3 .LST
                                    2         PASS4 .LST
                                    2         PASS5 .LST
LP0:        HUGO        Q           3         MEMMAP.DOC
MT1:        SAM         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 This option displays the subsetting of physical disks into logical disks. The logical disk unit is displayed with the file name to which it is assigned and the size of the logical disk in decimal blocks.

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[4000,3
LD2: is DL1:DISK.SRC[1200,3
LD1: is DL1:WORK.DSK[600,3
```

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, the system prints the message *LD handler unavailable*. If no logical disk units have been mounted (by using the MOUNT command), the system prints *No LD units mounted*.

TERMINALS This option 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 prints:

```
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 prints a table of the existing terminals and lists the following information:

Unit number:	0–15
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 or DZ

Width: Width in characters, up to 255

SET options in effect:

TAB
CRLF
FORM
SCOPE

Line speed: Baud rate if DZ; 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           RESURC   G-Console DL    132   No   Yes    No    No    N/A
1           F        Local     DL     80   Yes  No     No    Yes   N/A
```

SQUEEZE

The SQUEEZE command consolidates all unused blocks into a single area on the device you specify and consolidates the directory entries on the device.

In the command syntax illustrated above, device represents the random-access volume to be compressed. To perform a squeeze operation, the system moves all the files to the beginning of the device you specify, producing a single unused area after the group of files. The squeeze operation does not change the bootstrap blocks of a device. The system requests confirmation before it performs the squeeze operation. You must type Y followed by a carriage return to execute the command.

The squeeze operation does not move files with .BAD file types. This feature prevents you from reusing bad blocks that occur on a disk. During a squeeze operation, files with a .BAD file type will be renamed FILE.BAD. The system inserts files before and after .BAD files until the space between the last file it moved and the .BAD file is smaller than the next file to be moved. Note that you should use the SQUEEZE command when you get a directory full error, even if there is still space remaining on the volume.

If you perform a squeeze operation on the system device, the system automatically reboots the running monitor when the compressing operation completes. This reboot takes place in order to prevent system crashes that might occur when the monitor file or handler files are moved. The system volume cannot be squeezed if a foreground or system job is loaded.

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.

The options for the SQUEEZE command follow.

/OUTPUT:device Use this option to transfer all the files from the input device to the output device in compressed format, an operation that leaves the input device unchanged. The output device must be an initialized random-access volume. (Use the INITIALIZE command to do this.) Note that the system does not request confirmation before this operation proceeds. If the output device is not initialized, the system prints an error message and does not execute the command. Note that /OUTPUT does not copy boot blocks; you must use the COPY/BOOT command to make the output volume bootable.

The following example transfers all the files from RK0: to RK1: in compressed format, leaving RK0: unchanged.

```
.SQUEEZE/OUTPUT:RK1: RK0:
```

/QUERY This option causes the system to request confirmation before it executes a squeeze operation. You must respond by typing a Y or any string beginning with Y, followed by a carriage return, for execution to proceed. This is the default operation. /QUERY is invalid with the /OUTPUT option.

/NOQUERY Use this option to suppress the confirmation message that prints before a squeeze operation executes. The following command compresses all the files on device DY1: and does not query.

```
.SQUEEZE/NOQUERY DY1:
```

/WAIT When you use /WAIT, the system initiates the SQUEEZE operation, but then pauses and waits for you to mount the volume you want to squeeze. This option is especially useful if you have a single-disk system.

When the system pauses, it prints *Mount input volume in <device>; Continue?*. Mount the volume and type Y or any string beginning with Y, followed by a carriage return, to continue the squeeze operation. Type N or any string beginning with N, or two CRTL/Cs, to abort the squeeze operation and return control to the keyboard monitor. Any other response causes the message to repeat.

When the squeeze operation is complete the system prompts you to remount the system volume. Mount the system volume and type Y or any string beginning with Y, followed by a carriage return. If you type any other response the system prompts you to mount the system volume until you type Y. The system then prints the keyboard monitor prompt. When you use the /WAIT option, make sure DUP is on the system volume.

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
```

The system may repeat the mount input volume, mount output volume cycle several times to complete the SQUEEZE operation.

SRUN

The SRUN command initiates system jobs.

In the command syntax illustrated above, filespec represents the program to be executed. Because this command runs a system job, it is valid only for FB and XM monitors that have system job support, a special feature enabled through the system generation process.

You can run up to six system jobs simultaneously, in addition to the foreground and background jobs. If you attempt to run a system job that is already active, an error message prints on the terminal.

Note that when you issue the SRUN command, the monitor assumes a .REL file type. The default device is SY:.

In an XM monitor, you can use the SRUN command to run a virtual .SAV image program. You must type the file type explicitly.

The options that you can use with SRUN follow.

/BUFFER:n Use this option to reserve space in memory over the actual program size. The argument n represents the number of octal words of memory to allocate. You must use this option to run any FORTRAN program as a system job. If you use this option for a virtual job linked with the /V option (or /XM), the system ignores /BUFFER because the system provides a buffer in extended memory.

/LEVEL:n Use this option to assign an execution priority level to the job, where n can be 1, 2, 3, 4, 5, or 6. If you attempt to assign the same priority level to two system jobs, an error message prints at the terminal. If omitted, the priority level defaults to the highest level that is unassigned.

/NAME:logical-jobname Use this option to assign a logical job name to a program. This is the name that programmed requests and SYSLIB calls use to reference a system job. If you attempt to assign the same logical job name to two system jobs, an error message prints at the terminal. If you do not specify a logical job name, the system assumes the file name of the program.

/PAUSE Use this option to help you debug a program. When you type the carriage return at the end of the command string, the system prints the load address of your program and waits. By means of ODT, you can examine or modify the program before starting execution (see Chapter 20 of the *RT-11 System Utilities Manual*). You must use the RESUME command to restart the system job.

The following command loads the program MFUNCT.SYS, prints the load address, and waits for a RESUME command to begin execution.

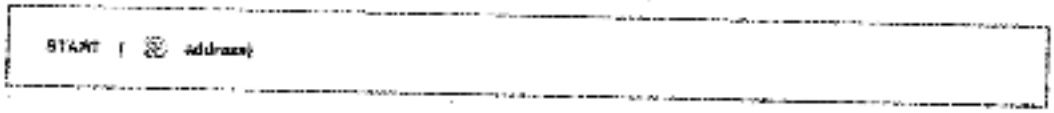
```
.SRUN MFUNCT/PAUSE
Loaded at 125555
.RESUME MFUNCT
```

/TERMINAL:n Use this option to change the console of the system job. Your system must have multiterminal support, a special feature available only through system generation, before you can use it. The argument n represents a terminal logical unit number. By assigning a different terminal to interact with the system job, you eliminate the need for system, foreground, and background jobs to share the console terminal.

Note that the original console terminal still interacts with the background job and with the keyboard monitor, unless you use the SET TT: CONSOL command to change this.

START

The START command initiates execution of the program currently in memory (loaded with the GET command) at the address you specify.



In the command syntax shown above, address is an even octal number representing any 16-bit address. If you omit the address or if you specify 0, the system uses the starting address that is in location 40. If the address you specify does not exist or is invalid for any reason, a trap to location 4 occurs and the monitor prints an error message. Note that this command is valid for background jobs only, and not for extended-memory virtual jobs.

The following example loads MYPROG.SAV into memory and begins execution.

```
.GET MYPROG
.START
```

The next example 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 SY:PROG
.GET ODT
.START 1222
 ODT V05.00
*
```

The SUSPEND command temporarily stops execution of the foreground or system job.

```
SUSPEND [ job [job...]]
```

If you have system job support enabled on your monitor, specify the name of the system or foreground job you wish to suspend. If you do not have system job support, then do not include an argument with the SUSPEND command. The SUSPEND command is not valid for the SJ monitor.

The system permits foreground input and output that are already in progress to finish; however, it issues no new input or output requests and enters no completion routines (see the *RT-11 Programmer's Reference Manual* for a detailed explanation of completion routines). You can continue execution of the job by typing the RESUME command.

The following command suspends execution of the foreground job that is currently running on a system that does not have system job support.

```
.SUSPEND
```

The next command suspends execution of the system job, QUEUE, that is currently running on a system that does have system job support.

```
.SUSPEND QUEUE
```

TIME

Use the TIME command to set the time of day or to display the current time of day.

```
TIME [ hh mm ss]
```

In the command syntax shown above, *hh* represents the hour (from 0 to 23), *mm* represents the minutes (from 0 to 59), and *ss* represents the seconds (from 0 to 59). The system keeps time on a 24-hour clock.

To enter the time of day, specify the time in the format described above. You should do this as soon as you bootstrap the system. The following example enters the time, 11:15:00 A.M.

```
.TIME 11:15
```

As this example shows, if you omit one of the arguments the system assumes 0.

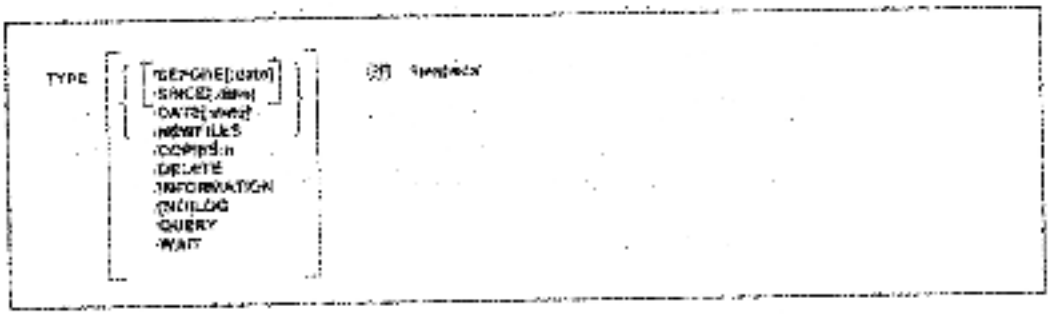
To display the current time of day, type the TIME command without an argument, as this example shows.

```
.TIME
11:15:01
```

When you install the standard RT–11 monitors, the clock rate is preset to 60 cycles. Consult the *RT–11 System Generation Guide* for information on setting the clock to a 50-cycle rate.

The FB and XM monitors automatically reset the time each day at midnight when a TIME command is used, or if a .GTIM programmed request is issued (The TIME command issues a .GTIM programmed request.) The SJ monitor resets the time under these conditions only if you select timer support during the system generation process.

The TYPE command prints the contents of one or more files on the terminal.



In the command syntax illustrated above, filespecs represents the file or files to be typed. You can explicitly specify up to six files as input to the TYPE command. The system prints the files in the order in which you specify them in the command line. You can also use wildcards in the file specification. In this case, the system prints the files in the order in which they occur in the directory of the device you specify. If you specify more than one file, separate the files by comman. If you omit the file type for a file specification, the system assumes .LST. You can specify the entire command on one line, or you can rely on the system to prompt you for information. The TYPE command prompt is *Files?*.

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

[dd]:mmm][:yy]

where:

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

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), the system uses the values from the current system date. For example, if you specify only the year :82 and the current system date is May 4, 1983, the system uses the date 4:MAY:82. 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.)

The TYPE command options and examples follow.

/BEFORE[:date] This option prints on the terminal all files on a specified volume created before a specified date. The following command prints only those .MAC files on DK: created before March 24, 1983.

```
.TYPE/BEFORE:24-MAR-83 DK:*.MAC
```

/COPIES:n Use this option to list more than one copy of the file. The meaningful range of values for the decimal argument n is from 2 to 32 (1 is the default). The following command, for example, prints three copies of the file REPORT.LST on the terminal.

```
.TYPE/COPIES:3 REPORT
```

/DATE[:date] Use this option to print 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, 1983:

```
.TYPE/DATE:20-MAR-83 DK:*.MAC
```

/DELETE Use this option to delete a file after it is typed on the terminal. This option must appear following the command in the command line. The TYPE/DELETE operation does not ask you for confirmation before it executes. You must use /QUERY for this function.

The following example prints a BASIC program on the terminal, then deletes it from DY1:.

```
.TYPE/DELETE DY1:PROG1.243
```

/INFORMATION Use this option to change the severity level of the error message that prints when not all of the input files you specified are found. If you do not use /INFORMATION, the system prints an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, the system prints 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 printed. However, since the system is unable to find DL0:FILE2.TXT, the system prints a message to inform you.

```
.TYPE/INFORMATION DL0:[FILE1,FILE2,FILE3].TXT
?PIP-I-File not found DL0:FILE2.TXT
```

/LOG This option prints on the terminal the names of the files that were printed by the current command. Normally, the system prints a log only if there is a wildcard in the file specification. If you specify /QUERY, the query messages replace the log, unless you specifically type /LOG/QUERY in the command line.

The following example shows a TYPE command and the resulting log.

```
.TYPE/LOG OUTFIL.LST
 Files copied:
DK:OUTFIL.LST    to TT:
```

/NOLOG This option prevents a list of the printed files from printing on the terminal. You can use this option to suppress the log if you use a wildcard in the file specification.

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

```
.TYPE/NEWFILES *.LST
 Files copied:
DK:REPORT.LST    to TT:
```

/QUERY If you use this option, the system requests confirmation before it performs the operation. /QUERY is particularly useful on operations that involve wildcards, when you may not be sure which files the system selected for an operation. Note that if you specify /QUERY in a TYPE command line that also contains a wildcard in the file specification, the confirmation messages printed on the terminal replace the log messages that would normally appear.

You must respond to a query message by typing Y or any string beginning with Y, followed by a carriage return, to initiate execution of a particular operation. The system interprets any other response to mean NO and does not perform the specific operation.

```
.TYPE/QUERY/LOG *.LST
 Files copied/deleted:
DK:OUTFIL.LST    to TT:? NO
DK:REPORT.LST    to TT:? Y
```

/SINCE[:date] This option prints on the terminal all files on a specified volume created on or after a specified date. The following command prints only those .MAC files on DK: created on or after April 21, 1983.

```
.TYPE/SINCE:21-APR-83 *.MAC
```

/WAIT This option is useful if you have a single-disk system. When you use this option, the system initiates the TYPE operation, but then pauses and waits for you to mount the volume that contains the volumes you want to print.

When the system pauses, it prints *Mount input volume in <device>; Continue?*. Mount the input volume and type Y or any string beginning with Y, followed by a carriage return, to continue the operation. Type N or any string beginning with N, or two CTRL/Cs, to abort the the operation and return control to the keyboard monitor. Any other response causes the message to repeat.

After the system completes the operation, the system prints the following message prompting you to mount the system volume:

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

Mount the system volume and type Y or any string beginning with Y, followed by a carriage return. If you type any other response the system prompts you to mount the system volume until you type Y. When you use /WAIT, make sure that PIP is on the system volume.

The following sample prints AJAX.DOC from an RL02 disk:

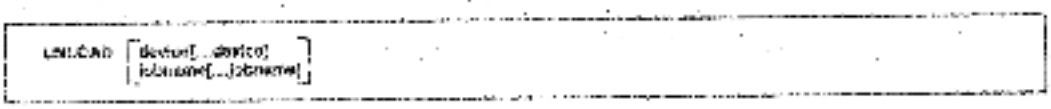
```
.TYPE/WAIT DL0:AJAX.DOC
Mount input volume in DL0:; Continue? Y
```

After the system has printed 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 a Y followed by a carriage return, you terminate the TYPE operation.

The UNLOAD command removes previously loaded handlers from memory, thus freeing the memory space they occupied. It also removes terminated foreground or system jobs.



In the command syntax shown above, device represents the device handler and jobname represents the job to be unloaded. You can specify both device handlers and job names on the same command line. The colon that follows the device handler is optional with SJ, FB, and XM monitors and monitors that do not have system job support. If your system has system job support, it is recommended that you type the colon to unload a handler. If you do not type a colon, the system checks the table of system jobs for a job with that name before it checks the device table. Therefore, if you have a system job with the same name as a device handler, you must include the colon to remove the handler.

UNLOAD clears ownership for all units of the device type you specify. A request to unload the system device handler clears ownership for any assigned units for that device, but the handler itself remains resident. After you issue the UNLOAD command, the system returns any memory it frees to a free memory list. The background job eventually reclaims free memory. Note that if you interrupt an operation that involves magtape, you must unload and then load (with the LOAD command) the appropriate device handler (MM, MT, or MS).

The system does not accept an UNLOAD command while a foreground job is running if the foreground job owns any units of that device. This is because a handler that the foreground job needs might become nonresident. You can unload a device while a foreground job is running if none of its units belong to the foreground job.

A special function of this command is to remove a terminated foreground or system job and reclaim memory, since the system does not automatically return the space occupied by the foreground or system job to the free memory list.

The following command unloads the foreground job and frees the memory it occupied. This command is valid only if the foreground job is not running.

```
.UNLOAD F
```

The following command unloads the system job QUEUE.

```
.UNLOAD QUEUE
```

The following command clears ownership of all units of RK:. If RK: is the system device, the RK handler itself remains resident.

```
.UNLOAD RK:
```

The next command releases the line printer and RL02 handlers and frees the area they previously held.

```
.UNLOAD LP:,RL:
```

The UNPROTECT command removes a file's protected status so that you can delete the file.

In the command syntax illustrated above, filespecs represents the file or files whose protected status you want to remove. 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.

You can explicitly specify up to six files. If you specify more than one file, separate the files with commas. You can also use wildcards in the file specifications. You can enter the UNPROTECT command as one line, or you can rely on the system to prompt you for information. The UNPROTECT command prompt is *Files?*.

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

[dd][:mmm][:yy]

where:

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

The default value for the date is the current system date. If you omit any of the date values (dd, mmm, or yy), the system uses the values from the current system date. For example, if you specify only the year :82 and the current system date is May 4, 1983, the system uses the date 4:MAY:82. 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.)

The following sections describe options you can use with the UNPROTECT command and include command examples.

/BEFORE[:date] Use this option to remove 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, 1983.

```
.UNPROTECT/BEFORE:20-MAR-83 *.MAC
 Files unprotected:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

/DATE[:date] Use this option to remove 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, 1983.

```
.UNPROTECT/DATE:20-MAR-83 *.MAC
 Files unprotected:
DK:A.MAC
DK:B.MAC
DK:C.MAC
```

/EXCLUDE This option 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 DY0: except .SAV files.

```
.UNPROTECT/EXCLUDE DY0:*.SAV
?PIP-W-No .SYS action
 Files unprotected:
DX0:ABC.OLD
DX0:PAT.OLD
DX0:COMB.
DX0:MERGE.OLD
```

/INFORMATION Use this option to change the severity level of the error message that prints when not all of the input files you specified are found. If you do not use /INFORMATION, the system prints an error message when it is unable to find an input file, and execution halts after the command is processed. When you use /INFORMATION, the system prints an informational message to tell you which files it cannot find, but execution continues.

In the following example, the system removes protection from input files FILE1.TXT and FILE3.TXT. However, since the system is unable to find DL0:FILE2.TXT, the system prints a message to inform you.

```
.UNPROTECT/INFORMATION DL0:(FILE1,FILE2,FILE3).TXT
?PIP-I-File not found DL0:FILE2.TXT
```

/LOG This option 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, the system does not ask you for confirmation before execution proceeds. Use both /LOG and /QUERY to invoke logging and querying.

/NOLOG This option prevents a list of files from which protection is being removed from appearing on the terminal.

/NEWFILES Use this option to remove protection from only the files that have the current system date. The following example removes protection from the files created today.

```
.UNPROTECT/NEWFILES DY1:*.BAK
 Files unprotected:
DY1:MERGE.BAK ? Y
```

/QUERY Use this option to request a confirmation message from the system before it removes protection from 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.

You must respond to a query message by typing Y or any string beginning with Y, followed by a carriage return, to initiate execution of a particular operation. The system interprets any other response as NO; it does not perform the operation.

The following example shows querying. Protection is removed from only the file DY1:AAF.MAC:

```
.UNPROTECT/QUERY DY1:*.*
 Files unprotected:
DY1:ABC.MAC   ? N
DY1:AAF.MAC   ? Y
DY1:MERGE.FOR ? N
```

/SETDATE[:date] This option causes the system 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, the system places zeros in the directory entry date position. Normally, the system 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 DY0:*.FOR
 Files unprotected:
DY0:ABC.FOR
DY0:AAF.FOR
DY0:MERGE.FOR
```

/SINCE[:date] Use this option to remove 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 DY0: created on or after April 21, 1983:

```
.UNPROTECT/SINCE:21-APR-83 DY0:*.MAC
 Files unprotected:
DY0:A.MAC
DY0:B.MAC
DY0:C.MAC
```

/SYSTEM Use this option if you need to remove protection from system (.SYS) files and you use wildcards in the file type. If you omit this option, the system files are excluded from the unprotect operation and a message is printed on the terminal to remind you of this. This example removes protection from all files on DY0: with the file name MM, including .SYS files.

```
.UNPROTECT/SYSTEM DY0:MM.*
 Files unprotected.
DY0:MM.MAC
DY0:MM.OBJ
DY0:MM.SAV
DY0:MM.SYS
```

/WAIT This option is useful if you have a single-disk system. When you use this option, the system initiates the UNPROTECT operation but then pauses for you to mount the volume that contains the files whose protection status you want to change. When the system pauses, it prints *Mount input volume in <device>; Continue?*, where <device> represents the device into which you mount the volume. Mount the volume and type Y or any string beginning with Y, followed by a carriage return. Type N or any string beginning with N, or two CTRL/Cs, to abort the operation and return control to the keyboard monitor. Any other response causes the message to be repeated.

When the operation completes the system prints the *Continue?* message again. Mount the system volume and type Y or any string beginning with Y, followed by a carriage return. If you type any other response the system prompts you to mount the system volume until you type Y. The system then prints the keyboard monitor prompt. Make sure PIP is on your system volume when you use the /WAIT option.

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
```

4.7 Concise Command Language (CCL)

Concise Command Language (CCL) allows you to run a program and pass it a command string on a single command line.

When you type a CCL command, the keyboard monitor translates the command into a RUN SY: command followed by the program name you specify and one or more lines of file specifications and options for that program. When the operation completes, control returns to the keyboard monitor and it prompts you for another command.

The syntaxes for CCL commands are:

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

.PROGNAM out-filespec[/option]=in-filespec[/option]

where:

PROGNAM — represents the RT–11 utility program you want to run. These programs are described in the *RT–11 System Utilities Manual*.

in-filespec — represents the device, file names, and file types of the input files. Implicit wildcards are not allowed in file specifications; you must use the wildcard symbols * and %.

out-filespec — represents the device, file names, and file types of the output files. Implicit wildcards are not allowed in file specifications; you must use the wildcard symbols * and %. In the first syntax line shown above, out-filespec is optional.

[/option] — represents the single-character options for each utility program, as described in the *RT–11 System Utilities Manual*.

The following example copies all files on DY0: with the file type .MAC created on or after January 12, 1983 to DY1:.

```
.PIP DY0:*.MAC/I:12-JAN-83 DY1:*.*
```

The next example achieves the same results.

```
.PIP DY1:*.*=DY0:*.MAC/I:12-JAN-83
```

The next example calls KED to inspect the file JMS.MAC.

```
.KED JMS.MAC
```

Chapter 5
Indirect Control File Processor (IND)

This chapter tells you how to create indirect control files for execution by the Indirect Control File Processor (IND).

An indirect control file, or control file, consists of one or more lines of keyboard commands and special commands, called IND directives, that control system execution. You can use control files to execute keyboard commands, access files, and perform logical tests to control the flow of execution.

This chapter begins by describing how to create an indirect control file and how to structure its command lines. The second section describes how to execute the control files you create. Next follows a section of tables containing brief descriptions of all the IND directives and special characters, and a section describing the use of symbols in control files. The chapter concludes with detailed descriptions of how to use each IND directive.

5.1 Creating an Indirect Control File

An indirect control file consists of one or more lines of directives or keyboard commands to be processed by IND. Each line of an indirect control file can contain up to three elements, illustrated in Figure 5–1: a label, IND directives and keyboard commands, and a comment. Labels are used to mark specific locations in a program. IND directives (or simply directives) and keyboard commands control the execution of your program and perform specific operations. Comments are used to document the program. These elements are described in the following sections.

Figure 5–1: Indirect Control File Line Elements

.SUBONE:	.IFNDF SYM .GOSUB SUBTWO	;GET DEFINITION OF SYM
Label	IND directive	Comment

In general, follow these rules of syntax when you create indirect control files:

1. Separate directives from their arguments and from keyboard commands by at least one space or tab, unless otherwise indicated in this manual.

However, do not place spaces or tabs between arithmetic operators, such as plus signs (+) and minus signs (–), and their arguments. For example, the following command line is correct:

```
.SETN NUM (2+3)*2
```

The next line is incorrect:

```
.SETN NUM (2 + 3) * 4
```

2. Using tabs and spaces allows you to format the control files for readability. You can format control files by inserting spaces and horizontal tabs in a command line in the following locations:

   - At the start of the command line
   - Immediately following the colon (:) of a label
   - Directly before the semicolon (;) or period-semicolon (.;) of a comment
   - At the end of a command line

3. IND accepts up to 132 characters in a command line.

4. You can include up to 80 characters (before IND processing) in .ASK, .ASKN, and .ASKS prompts.

5. Keyboard monitor commands, concise command language (CCL) commands, and commands that run system utility programs must be complete on one line. For commands that query or prompt, you must either use /NOQUERY or the equivalent utility option, or enter the responses to the prompts at the terminal. The following example shows an incorrect method of running a utility program from within a control file; the command is not complete on one line.

```
RUN MACRO
PROG=PRUGA
^Z
```

5.1.1 Labels

A label assigns a name to a line in the control file so that the line can be referenced. Labels are from one to six characters long, and must be prefixed with a period (.) and suffixed with a colon(:). A label may contain only alphanumeric or dollar sign ($) characters. For example, in the following line, .START is a label.

```
.START:    .ASKS ANS DO YOU HAVE A LINEPRINTER?
```

Place labels at the beginning of a line. You may use a label on a line that has directives or keyboard commands, on a line with a comment, or on a line by

itself. You can place only one label on a line, but you need not use a label on every line.

See Section 5.5.1 for more information on using labels.

5.1.2 IND Directives and Keyboard Commands

Control file lines may also contain IND directives, keyboard commands, and CCL commands. Section 4.6 of this manual describes keyboard commands. Section 4.7 describes CCL.

Directives and keyboard commands can be used together on the same line or on separate lines. Some directives can also be used on the same line with other directives.

When processing a control file, IND reads the control file and interprets each command line either as a command to be passed directly to the keyboard monitor (KMON) or as a request for action by the indirect control file processor itself. IND directives, which are processed by the indirect control file processor, are preceded by a period. Keyboard commands have no preceding characters when used in a control file.

5.1.2.1 IND Directives — Directives can be used on a line with a label or a comment, or alone on a line. Directives must be placed after a label but before a comment. The following line shows that the directive follows the label and precedes the comment:

```
.VIDEO:   .SET VT                    .VIDEO TERMINAL AVAILABLE
```

Label IND directive Comment

IND directives allow you to:

- Define labels
- Define and assign values to three types of symbols: logical, numeric, and string
- Create and access data files
- Control the logical flow within a control file
- Perform logical tests
- Enable or disable operating modes
- Increment or decrement a numeric symbol
- Perform time-based operations, if you have timer support

IND directives are described in detail in Section 5.5.

5.1.2.2 Keyboard Commands – Keyboard commands may appear with labels but not on lines with comments. When IND encounters a keyboard command, IND passes the remainder of the command line to the keyboard monitor. Since the keyboard monitor would be unable to interpret the comment, an invalid command error results.

Keyboard commands used within an indirect control file must be complete on one line. Therefore, if you use keyboard commands that query (such as the DELETE command with wildcards) or prompt you (such as the LINK/PROMPT command), you must either specify the /NOQUERY option or enter the response to the query at the terminal when the command is executed. You can, however, process commands with more than one line by using the .ENABLE DATA or .DATA directive to create and execute an indirect file from within the control file.

The control file lines in the next two examples create and execute an indirect command file that runs PIP, unprotects the file DY:FILE.TST, and copies the file to DY1:. The first example uses the .ENABLE DATA directive to create the control file:

```
.ASKS DEV WHICH DEVICE CONTAINS FILE.TST?
.OPEN COPY.TMP
.ENABLE DATA
RUN PIP
'DEV':FILE.TST/Z
DY1:FILE.TST='DEV':FILE.TST/Z/Y
^C
.DISABLE DATA
.CLOSE
@COPY.TMP
```

The second example uses the .DATA directive to create and execute the same indirect file:

```
.ASKS DEV WHICH DEVICE CONTAINS FILE.TST?
.OPEN COPY.TMP
.DATA RUN PIP
.DATA 'DEV':FILE.TST/Z
.DATA DY1:FILE.TST='DEV':FILE.TST/Z/Y
.DATA ^C
.CLOSE
@COPY.TMP
```

See Section 5.2.4 for information on executing indirect files from control files. See Sections 5.5.9 and 5.5.14 for information on using the .DATA and .ENABLE DATA directives.

You can also issue CCL commands in control files to run the utility programs described in the *RT–11 System Utilities Manual*. The following example executes a CCL command if the RL02 device handler is loaded:

```
.ROUTN:  .IFLOA DL PIP RL:/SYSPRG/SAV=DL:PROG.SAV
```

The following command line copies a diskette in image mode if a certain condition is true. The .IFT directive tests the condition represented by the symbol LOGSYM for a true or false value.

```
.IFT LOGSYM DLS DY1:*=DY0:/I/Y
```

As is true for keyboard commands, options that query or prompt are not allowed unless you specify the option that prevents querying (as does /Y in the above example), or unless you enter the response to the query at the terminal when the control file is executed.

Keyboard monitor commands are described in detail in Section 4.6 of this manual, and the corresponding utility programs are described in the *RT–11 System Utilities Manual*.

NOTE

The following keyboard commands should not be used in control files, as they will produce unpredictable results:

Base	R
Deposit	REENTER
Examine	SAVE
GET	START

5.1.3 Comments

You can document a control file by including comments. Comments provide you with information but perform no operations.

You can include either external or internal comments in a control file. During execution, IND prints external comments at the console. To define an external comment, begin the comment with a semicolon (;). Internal comments are for documenting a control file internally and are never printed at the console. To define an internal comment, begin the comment with a period and semicolon (.;).

You can place an external comment on a line with a label and with directives that do not branch to another location. An external comment on a line with a branching directive is never printed, because IND branches before the comment is processed. You can place an internal comment on a line with a label or with most directives. When used on a line with a label, or with directives, the comment should be the last field on the line. However, you can not place any comment on a line with a keyboard command, since the keyboard monitor will interpret the comment as an invalid command.

Comments can include up to 80 characters including the period, semicolon, carriage return, and line feed characters. IND ignores any characters that exceed the 80-character limit.

The following line shows an internal command on a line with a label and directives. The second half of the comment appears on a line by itself.

```
.DEVICE:  .IFLOA DY0: .GOSUB COPY        .IDY0: IS LOADED, GO TO
                                         .COPY SUBROUTINE
```

5.2 Executing Indirect Control Files

You can execute indirect control files either from keyboard monitor level or from within another indirect control file. Section 5.2.3 discusses executing control files from within other control files. You can also execute indirect command files from control files, as described in Section 5.2.4.

To execute an indirect control file from keyboard monitor level, you must call IND as follows.

```
.R IND
*
```

The Command String Interpreter (CSI) prints an asterisk at the left margin of the terminal and waits for you to enter a command string. If you enter only a carriage return, IND prints its current version number and the CSI prompts you again for a command string.

You can type CTRL/C to halt IND and return to the keyboard monitor when IND is waiting for input from the console terminal. You must type two CTRL/Cs to abort IND at any other time.

The syntax of the command string is:

ctrl-filespec[/options][parameters]

where:

ctrl-filespec	represents the control file you want to execute. The default file type is .COM.
options	is one or more of the options listed in Table 5-1. The brackets surrounding the options are not part of the command syntax. The options you specify are stored in the local string symbol COMMAN with the entire command line, and in local string symbol P0. See Section 5.2.2 for more information on COMMAN and P0.
parameters	is one or more values (up to nine) that you want to pass to the control file. The parameters you specify are stored in local string symbols P1–P9. The brackets, which are not part of the command syntax, indicate that the parameters are optional. Separate the parameters you specify with a space. See Section 5.2.2 for more information on passing parameters.

You can also call IND and execute a control file by issuing the following command:

```
.RUN IND ctrl-filespec/options
```

or simply:

```
.IND ctrl-filespec/options
```

However, you can not pass parameters when you use this syntax.

When SET KMON IND is in effect, you can use the following syntax to execute a control file, in response to the keyboard monitor prompt (.):

@ctrl-filespec/options[parameters]

You can run IND from the console by specifying the console device (TT:) as the control file specification. IND prints an asterisk (*) to prompt you for a single command line to execute. Since IND processes one command line at a time when run from the console, labels have no meaning and branch instructions produce no results. This function is especially useful for testing the results of individual command lines.

In the following example, IND executes one command line and prompts for input.

```
.RUN IND TT:
*.ASKS NAM WHAT IS YOUR NAME?
* WHAT IS YOUR NAME? [S1 JOHN]
```

5.2.1 IND Options

IND options (See Table 5–1) allow you to change the way IND processes and displays a control file. The following sections describe the options you can use in an IND command string.

Table 5–1: IND Options

Option	Section	Function
/D	5.2.1.1	Deletes the control file when IND has finished processing that file.
/N	5.2.1.2	Directs IND to ignore all keyboard commands in the control file.
/Q	5.2.1.3	Suppresses the display of keyboard commands and their results.
/T	5.2.1.4	Displays each command line that has been processed.

5.2.1.1 Delete Control File Option (/D) – Deletes the control file when IND has finished processing that file. The processing of a control file is complete only when IND executes the .EXIT directive or reaches the end of the control file.

Therefore, if you use the /D option and IND encounters a .STOP directive or a slash (/) character, the file is not deleted. The .DISABLE DELETE directive in a control file overrides the /D option.

5.2.1.2 Suppress Keyboard Commands Option (/N) -- When you use the /N option, IND processes all the directives in a control file, but does not execute any of the keyboard commands. This function is useful if you wish to test the logical flow of your control file without executing any of the keyboard commands within.

If /N is used when calling a nested control file, keyboard commands are ignored until the previous control file is reentered, an .ENABLE DCL or .ENABLE MCR directive is found in the control file, or processing at all levels is complete.

5.2.1.3 Suppress Console Display Option (/Q) -- The /Q option suppresses the display of keyboard commands and their results as IND executes the keyboard commands. The /Q option remains in effect even when control passes to a nested control file. A .DISABLE QUIET directive in the control file overrides the /Q option.

5.2.1.4 Command Tracing Option (/T) -- When you use the /T option, IND prints on the console all directives or keyboard commands as they are processed. This option is useful for testing and debugging control files.

When you enable tracing, IND prints an exclamation mark (!) on the console before each line that begins with a directive. IND does not print any characters before comments and keyboard commands as they are processed.

If control branches to another location in the control file, IND does not print the command lines that have not been processed.

The /T option has the same effect as the .ENABLE TRACE directive in a control file and can be overridden from within the control file by the .DISABLE TRACE directive.

5.2.2 Passing Parameters

Passing parameters allows you to define the contents of certain symbols when IND begins processing a file. You can pass as many as nine parameters when you execute a control file. When you execute a control file, either directly from the terminal or by calling another control file from within a control file, IND stores the command line, including parameters, in the local string symbol COMMAN. IND then stores the parameters separately in local string symbols P1 through P9.

Passing parameters allows you to predetermine the value of these symbols when you begin execution of a control file. Parameter values can be global symbols, numeric values, or character strings.

When the following command line is executed, IND stores in COMMAN the string DX:COACH A B D.

```
$ IND
@DX:COACH A B D
```

IND breaks the command string into elements separated by spaces, which IND stores in the local string symbols P0 through P9. P0 contains the file specification and any options. P1 through P8 contain the parameters you specify. P1 contains the first space-delimited parameter, P2 the second, and so on. P9 contains any remaining parameters. Any parameter not present results in the equivalent symbol being set to null. The number of symbols set, excluding the symbols that contain null values, appears in the numeric symbol <STRLEN>.

When IND processes the command line @DX:COACH A B D, it produces the following symbols:

Symbol	Contents
COMMAN	DX:COACH A B D
P0	DX:COACH
P1	A
P2	B
P3	D
P4	null
P5	null
P6	null
P7	null
P8	null
P9	null
<STRLEN>	4

5.2.3 Nested Indirect Control Files

You can call indirect control files from within an indirect control file. This process is called *nesting control files*; the control file that is called from the first control file is referred to as nested. You can nest up to three control files, for a total of four levels of control files (including the first file).

Use a command line with the following syntax to call a control file from within a control file:

@ctrl-filespec[/options][parameters]

where:

ctrl-filespec — represents the control file to which you want to branch. The default file type is .COM.

options — is one or more of the options listed in Table 5–1. The brackets shown in the syntax illustration are not part of the command syntax. The options you specify are stored in the local string symbol COMMAN with the entire command line, and in local string symbol P0. See Section 5.2.2 for more information on these symbols.

parameters — is one or more parameters (up to nine) that you want to pass to the control file. The surrounding brackets in the syntax illustration, which indicate that the parameters are optional, are not part of the command syntax. The parameters you specify are stored in local string symbols P0–P9. See Section 5.2.2 for more information on passing parameters.

You cannot include either internal or external comments on this command line.

The .ENABLE GLOBAL directive allows you to define symbols as global to all nested indirect control file levels. Refer to Section 5.4.1 for information on defining global symbols. If you do not use this directive, each time IND enters a deeper level it masks all symbols defined by the previous level out of the symbol table, so that only symbols defined in the current level are available. These symbols are recognized only within the level of the control file in which the symbols are defined. When control returns to a previous level, the symbols defined in that level become available again and the symbols from the lower levels are lost.

5.2.4 Executing Indirect Command Files from Control Files

The method for calling an indirect command file from within an indirect control file is similar to calling a control file from a control file (see Section 5.2.3). You can invoke an indirect command file from a control file by placing a dollar sign, at sign ($@) sequence before the name of the indirect command file you wish to invoke. When you pass control to an indirect command file, the keyboard monitor processes and executes the file. Control then returns to the control file from which the indirect command file was called.

The command to invoke an indirect command file is:

$@filespec

where:

filespec — represents the indirect command file you wish to invoke. The default file type is .COM.

For example, the following command line invokes the indirect command file DYOUT.COM:

```
$@DYOUT
```

After the keyboard monitor has finished executing the indirect command file, IND resumes processing of the control file from which the command file was called.

5.3 Directive Summary

This section includes four tables of abbreviated information. The IND directives described in this chapter are listed in Table 5–2 by category. Use these directives in your control files to direct execution. Table 5–3 lists the operating modes you can use with the .ENABLE and .DISABLE directives. A detailed description of each directive and operating mode is presented in alphabetical order in Section 5.5. Table 5–4 lists some special IND characters, and Table 5–5 lists the arithmetic, logical, and relational operators you can use to form numeric expressions. Refer to Section 5.4.3.2 for more information on numeric expressions.

Table 5–2: IND Directive Summary

Directive	Section	Function
Label Definition		
label:	5.5.1	Assigns a name, represented by label, to a line in the control file so that the line can be referenced.
Symbol Definition		
.ASK	5.5.3	Prints a prompt, and uses the response to define or redefine a logical symbol and assign the symbol a logical (true or false) value.
.ASKN	5.5.4	Prints a prompt, and uses the response to define or redefine a numeric symbol and assign it a numeric value.
.ASKS	5.5.5	Prints a prompt, and uses the response to define or redefine a string symbol and assign it a string value.
.DUMP	5.5.13	Displays local, global, and special symbol definitions.
.ERASE	5.5.16	Deletes local or global symbols from the symbol tables.
.PARSE	5.5.24	Breaks a string into substrings.
.SETD/SETO	5.5.26	Redefines a numeric symbol to decimal (.SETD) or octal (.SETO) radix.
.SETL	5.5.29	Defines or redefines a logical symbol and assigns it a logical value.
.SETN	5.5.30	Defines or redefines a numeric symbol and assigns it a numeric value.
.SETS	5.5.31	Defines or redefines a string symbol and assigns it a string value.
.SETT/SETF	5.5.32	Defines or redefines a logical symbol or redefines bits within a numeric symbol, and assigns the symbol or bits a true or false value.
.TEST	5.5.34	Tests attributes of a symbol or string and stores the results in special symbols.

(Continued on next page)

Table 5–2: IND Directive Summary (Cont.)

Directive	Section	Function
.TESTDEVICE	5.5.35	Tests a specified device and stores the device attributes in the special symbol <$XSTR>.
.TESTFILE	5.5.36	Determines if a file exists and stores the results in the special symbols <FILSPC> and <FILEXR>.
.VOL	5.5.37	Assigns a volume ID to a string symbol.
File Access		
.CHAIN	5.5.7	Closes the current control file, opens another control file, and resumes execution.
.CLOSE	5.5.8	Closes an output data file.
.DATA	5.5.9	Specifies a single line of data to be sent to an output data file.
.OPEN	5.5.23.1	Creates an output data file. If the file you specify with .OPEN already exists, .OPEN creates a new file and will delete the existing file if you subsequently use the .CLOSE directive. Use the .OPEN directive only when you wish to write to a file.
.OPENA	5.5.23.2	Opens an existing file and appends subsequent data to it. If the file you specify does not exist, .OPENA creates a new file. Use this directive only when you wish to write to a file.
.OPENR	5.5.23.3	Opens an existing file for use with the .READ directive. Use this directive only when you wish to read from a file.
.PURGE	5.5.25	Discards or closes an output file without making any changes to the file.
.READ	5.5.26	Reads the next record from a file into a string variable. The file must have been previously opened with .OPENR.
Logical Control		
.BEGIN	5.5.6	Marks the beginning of a begin-end block.
.END	5.5.15	Marks the end of a begin-end block.
.EXIT	5.5.17	Terminates processing of either the current control file or a begin-end block, and returns control to the previous level; can also assign a value to the numeric symbol <EXSTAT> (see the <$XSTAT> special symbol description in Table 5–6).

(Continued on next page)

Table 5–2: IND Directive Summary (Cont.)

Directive	Section	Function
.GOSUB	5.5.18	Branches to a subroutine within the control file.
.GOTO	5.5.19	Branches to another location in the control file.
.ONERR	5.5.22	On detecting an error, branches to another location in the control file.
.RETURN	5.5.27	Returns control from a subroutine to the line immediately following that subroutine's call.
.STOP	5.5.33	Terminates control file processing.
Logical Tests		
.IF	5.5.20.1	Determines whether a symbol satisfies one of several possible conditions.
.IFDF/.IFNDF	5.5.20.2	Determines whether a symbol is defined or not defined.
.IFENABLED/ .IFDISABLED	5.5.20.3	Determines whether an operating mode is enabled or disabled. See Section 5.5.14 for descriptions of the operating modes.
.IFLOA/.IFNLOA	5.5.20.4	Determines whether a device handler has been loaded or has not been loaded.
.IFT/.IFF	5.5.20.5	Determines whether a logical symbol is true or false, or tests specific bits in a numeric symbol.
Execution Control		
.DELAY	5.5.11	Delays control file processing for a specified period of time.
Enable/Disable Operating Modes		
.DISABLE	5.5.12	Disables the operating modes. See Table 5–3 and Sections 5.5.12 and 5.5.14 for more information on the operating modes.
.ENABLE	5.5.14	Enables the operating modes. See Table 5–3 and Sections 5.5.12 and 5.5.14 for more information on the operating modes.
Increment/Decrement Numeric Symbols		
.DEC	5.5.10	Subtracts one from the value of a numeric symbol.
.INC	5.5.21	Adds one to the value of a numeric symbol.

Table 5–3: Operating Modes

These operating modes are arguments for the .ENABLE and .DISABLE directives. The entry in the scope column refers to whether the operating mode automatically returns to its default setting or remains at its current setting when control passes to a nested control file. Local operating modes return to their default settings; global operating modes keep their current settings. See Sections 5.5.12 and 5.5.14 for more information on operating modes.

Operating Mode	Default	Scope	Function
ABORT	Enabled	Global	When ABORT is enabled, a control file can be aborted by typing a double CTRL/C.
CONTROL-Z	Enabled	Global	When CONTROL-Z is enabled, typing a CTRL/Z in response to an .ASK, .ASKN or .ASKS prompt aborts the control file.
DATA	Disabled	Local	When DATA is enabled, IND sends to an output file all lines that follow the .ENABLE DATA directive until a .DISABLE DATA or .CLOSE directive is encountered.
DCL	Enabled	Local	When DCL is disabled, IND suppresses execution of keyboard commands in a control file.
DELETE	Disabled	Local	When DELETE is enabled, control files are deleted after execution of the file has completed.
ESCAPE	Disabled	Global	When ESCAPE is enabled, IND recognizes the escape character as a valid response to an .ASK, .ASKS, or .ASKN directive.
GLOBAL	Disabled	Global	When GLOBAL is enabled, symbol names that begin with a dollar sign ($) are recognized as global symbols; that is, these symbols are recognized throughout all levels of control files.
LOWERCASE	Enabled	Global	When LOWERCASE is enabled, characters typed in response to an .ASKS directive are stored in the string symbol without automatic lowercase to uppercase conversion.
MCR	Enabled	Local	When MCR is disabled, IND suppresses execution of keyboard commands in a control file.
OCTAL	Enabled	Global	When OCTAL is enabled, the default radix of responses to .ASKN directives and of numeric symbol definitions is octal.
PREFIX	Enabled	Global	When PREFIX is disabled, IND suppresses printing of the asterisk (*) before all prompts that result from .ASK, .ASKN, and .ASKS directives, and the semicolon (;) in front of comments.

(Continued on next page)

Table 5–3: Operating Modes (Cont.)

Operating Mode	Default	Scope	Function
QUIET	Disabled	Local	When QUIET is enabled, IND does not display keyboard command lines.
SUBSTITUTION	Enabled	Global	When SUBSTITUTION is enabled, IND replaces symbols with their assigned values.
SUFFIX	Enabled	Global	When SUFFIX is disabled, IND suppresses printing of the question mark and [Y/N] notation at the end of an .ASK prompt, and suppresses range, default, timeout and execution type notations for all ASK directive prompts.
TIMEOUT	Disabled	Global	When TIMEOUT is enabled, IND recognizes the timeout parameter for ASK directives if your monitor includes timer support.
TRACE	Disabled	Local	When TRACE is enabled, IND displays the command line processed.
TYPEAHEAD	Enabled	Global	When TYPEAHEAD is enabled, characters typed before an .ASK, .ASKN or .ASKS prompt are included in the prompt response.

Table 5–4 Special IND Characters

This table lists a set of characters which have special meaning during control file execution.

Character	Function
.	Used as a prefix character for all IND directives and labels, and as a suffix character for all integers that have a decimal radix.
#	Used as a prefix character to define a response to an .ASKN directive prompt as octal. Also used as a prefix character for file numbers specified with the .CLOSE, .DATA, .OPEN, .OPENA, .OPENR, and .PURGE directives.
@	Used before a nested control file specification to pass control to that file.
$	Used as a prefix character to define symbols as global.
$@	Used before an indirect command file specification to pass control to that file.
/	Used as an end-of-file character. This character has the same effect as the .STOP directive. See Section 5.5.2 for more details.
+	Concatenates string symbols (see Section 5.4.4).
;	Indicates the start of an external comment. During execution, IND prints external comments at the console. These comments can have up to 132 characters. If you use the period-semicolon (.;) to begin a comment, the comment is an internal comment, which IND does not print.

Table 5–5: Arithmetic, Logical, and Relational Operators

This table lists the arithmetic, logical, and relational operators you can use to form numeric expressions.

Character	Function
+	Add
–	Subtract
*	Multiply
/	Divide
|	Logical inclusive OR
&	Logical AND
^	Logical NOT
EQ or =	Equal to
NE or <>	Not equal to
GE or >=	Greater than or equal to
LE or <=	Less than or equal to
GT or >	Greater than
LT or <	Less than

5.4 Symbols

You can use symbols in your control files as variables that represent logical, numeric, or string values. By testing or comparing these symbols, you can control the flow of execution. You can also substitute symbols for keyboard commands, data records or data files, or comments to be displayed on the console.

All symbols are from one to six characters in length. Symbols can include alphanumeric characters and dollar signs ($).

Symbols can be one of three types: logical, numeric, and string. Any of these symbols can be local or global.

You use directives to define a symbol's type and value. You can redefine a symbol's value throughout a control file, but you cannot redefine its type except within a begin-end block. See Section 5.5.6 for more details on begin-end blocks.

5.4.1 Local and Global Symbols

Logical, numeric, and string symbols can be defined as either local or global. Local symbols are recognized only within the begin-end block or control file in which the local symbols are defined. Global symbols are recognized throughout all levels of nested control files.

You can define a symbol as global by including the .ENABLE GLOBAL directive in your command file. After the .ENABLE GLOBAL directive, symbols that begin with a dollar sign ($) are defined as global. Local symbols are those that are defined previous to the .ENABLE GLOBAL directive, after a dollar sign ($).

Once a symbol is defined as global or local you cannot change its scope. Therefore, if a symbol that begins with a dollar sign ($) is defined before the .ENABLE GLOBAL directive, the symbol is defined as local and remains local even after a later .ENABLE GLOBAL directive. After an .ENABLE GLOBAL directive, if another symbol with the same name is defined, there will be two symbols with the same name: one local symbol and one global symbol.

5.4.2 Logical Symbols

Logical symbols represent true or false values. You can define or redefine a logical symbol by using the .ASK, .SETL, .SETT, or .SETF directive. See Section 5.5.

5.4.3 Numeric Symbols

A numeric symbol represents an integer. You can use numeric symbols to represent integer arguments in command lines, and you can combine them with other numeric symbols and constants to form arithmetic expressions. The valid range for an integer represented by a numeric symbol is 0 to 65535 (decimal) or 177777 (octal).

5.4.3.1 Defining the Radix of a Numeric Symbol — When a numeric symbol is defined using the .ASKN or .SETN directive, the default radix (octal or decimal) of the symbol definition is determined by the status of the OCTAL operating mode. When OCTAL is enabled, the symbol definition is stored as an octal number; when OCTAL is disabled, the symbol definition is stored as a decimal number.

When you use .ASKN, you can control the default radix by specifying decimal or octal values for range, default, and timeout indicator values. Indicator values you specify with a decimal point (.) are interpreted as decimal. When decimal indicator values are specified, or when octal mode is disabled, the response to the prompt will be stored in decimal. IND indicates

this by printing a (D) decimal indicator when it processes the .ASKN prompt. If an octal response is given by placing a number sign (#) before the number, the response is stored as its decimal equivalent. When octal mode is enabled and you specify no decimal indicator values, the response will be stored as an octal value; IND prints an (O) octal indicator instead. If a decimal response is given by placing a decimal point (.) after the number, the response is stored as its octal equivalent. See Section 5.5.4 for more information on the .ASKN directive.

When you use the .SETN directive, you can override the octal default radix by specifying decimal values in the numeric expression that defines the symbol. Values you specify with a decimal point (.) are interpreted as decimal. If octal mode is enabled and you specify no decimal values, the expression is evaluated and stored as an octal value. When you use all decimal values in the expression, or when octal mode is disabled, the expression is evaluated and stored as a decimal value. When you use both octal and decimal values and octal mode is enabled, all octal values are converted to decimal before the expression is evaluated and the symbol is stored as a decimal value. See Section 5.5.30 for more information on the .SETN directive.

Once a numeric symbol is defined as an octal or decimal value, you can change the symbol's radix by using the .SETD and .SETO directives. See Section 5.5.28 for more information on these two directives.

5.4.3.2 Numeric Expressions – You can form numeric expressions by using operators to combine numeric symbols and constants.

Operator	Function
+	Addition
-	Subtraction
/	Division
*	Multiplication
!	Logical OR
&	Logical AND
EQ or =	Equal to
NE or <>	Not equal to
GE or >=	Greater than or equal to
LE or <=	Less than or equal to
GT or >	Greater than
LT or <	Less than

When you divide numeric symbols, IND always truncates the dividend to yield an integer.

Numeric expressions are evaluated from left to right, unless you use parentheses to form subexpressions which IND evaluates first. Do not put blanks or tabs between operators and numeric symbols. The following directive lines assign numeric symbol N3 the value 24 (octal):

```
.SETN N1 2
.SETN N2 3
.SETN N3 N1+N2*4
```

In the next example, IND assigns the symbol N3 the value 16 (octal):

```
.SETN N1 2
.SETN N2 3
.SETN N3 N1+(N2*8)
```

5.4.4 String Symbols

You can use a string symbol to represent a string of up to 132 ASCII characters. You assign string symbols with the .ASKS and .SETS directives.

When you assign a character string to a string symbol, enclose the character string with quotes, as in the following example, which assigns a string to the string symbol PROMPT:

```
.SETS PROMPT "DO YOU HAVE ANY PROTECTED FILES"
```

IND permits you to break string symbols into substrings. You can use substrings with only the .SETS and .IF directives. In the following example, string symbol ALPHA2 is assigned four characters from the string represented by BETA2:

```
.SETS BETA2 "DEVICE DX1: DK2: DK1:"
.SETS ALPHA2 BETA2[8.:11.]
```

In this example, [8.:11.] indicates that characters 8 to 11 (decimal) of BETA2 are to make up ALPHA2. ALPHA2 consequently contains the characters DX1:. The square brackets are part of the command line; you must use them when you specify the range for a substring. If you use a decimal point with either number in the range specification (that is, the numbers that appear between the brackets), IND interprets both numbers as decimal.

String symbols can also represent other string symbols. In the next example, the string symbol NAME is assigned the contents of a previously defined string symbol, VOLID.

```
.SETS NAME VOLID
```

You can concatenate string symbols with other string symbols, substrings, and character strings by using the plus sign (+). The .SETS directive description (Section 5.5.31) provides more details on concatenating string symbols.

5.4.5 Special Symbols

In addition to the symbols that you create, IND has its own special symbols, listed in Table 5-6. IND sets these symbols according to specific system characteristics and responses to queries presented during command file execution; you cannot set them directly. You can, however, test these symbols to determine system characteristics. IND special symbols are enclosed in angle brackets so you can distinguish them from symbols you create.

Table 5–6: IND Special Symbols

Symbol	Value
Logical Symbols	
<ALPHAN>	Set to true if last string entered in response to an .ASKS directive contains only alphanumeric characters. The .TEST and .TESTFILE directives can set <ALPHAN> to true or false. An empty string sets <ALPHAN> to true, and any embedded blank sets <ALPHAN> to false. If lowercase mode is disabled, a lowercase response to a query sets <ALPHAN> to false.
<ALTMOD> or <ESCAPE>	Set to true if the last query was answered with a single escape character. Otherwise set to false.
<DEFAULT>	Set to true if response to last query was a default response (that is, a carriage return was entered).
<EOF>	Set to true after the .READ directive encounters end-of-file. Otherwise set to false.
<FALSE>	Permanently set to false; can be used to specify a default response for an .ASK directive.
<MAPPED>	Set to true if IND is running under the XM monitor; false if otherwise.
<OCTAL>	Set to true if the last numeric symbol tested with the .TEST directive contained an octal value; false if the last numeric symbol contained a decimal value. Also set to true if a numeric symbol defined using the .SETN or .ASKN directive was assigned an octal value; false if assigned a decimal value.
<RAD50>	Set to true if last string entered in response to an .ASKS directive, or tested with a .TEST or .TESTFILE directive, contains only Radix–50 characters. The period and dollar sign are valid Radix–50 characters, but blank characters and lowercase alphabetic characters are not. An empty string sets <RAD50> to true.
<TIMOUT>	Set to true if the last response to an ASK directive exceeded the timeout count; false if otherwise.
<TRUE>	Permanently set to true; can be used to specify a default response for an .ASK directive.
Numeric Symbols	
<ERROR>	Permanently assigned the value 2 (octal) to represent an error.
<EXSTAT>	Assigned the exit status value of 0, 1, 2, or 4 according to the contents of the user error byte as a result of the last keyboard command executed. The values 0, 1, 2, and 4 indicate: 0 Warning 1 Success 2 Error 4 Severe error This special numeric symbol is modified at the completion of each keyboard command. The .EXIT directive can also modify <EXSTAT>. The value is returned from the task that has completed

(Continued on next page)

Table 5–6: IND Special Symbols (Cont.)

Symbol	Value
	if you have used the .EXIT directive to specify an exit status value. Otherwise, the value is returned from KMON.
<FILERR>	Assigned a numeric status code indicating whether a file operation was successful. The codes returned, along with their descriptions, are: 1 Success 372 No room to fetch handler 366 End-of-file detected 365 Data overrun 359 Device full 346 No such file 340 File accessed for write 340 Device read error 337 Device write error 333 No file accessed on channel 327 File exceeds space allocated 325 Bad record type (non-ASCII) 324 File accessed for read 313 File already open 312 Bad file name 244 Invalid device or unit
<SEVERE>	Permanently assigned the value 4 (octal) to represent a severe error.
<SPACE>	Assigned an octal number to represent the amount of space available, in bytes, in the symbol table. IND uses a minimum of 10 bytes (octal) for each symbol.
<STRLEN>	Assigned the length, in octal, of the string entered in response to the last .ASKS directive or the string tested by the last .TEST directive.
<SUCCES>	Permanently assigned the value 1 (octal) to represent success.
<SYMTYP>	Assigned a numeric code to represent the type of symbol tested by a .TEST directive. The codes represent the following: 0 Logical symbol 2 Numeric symbol 4 String symbol
<SYSTEM>	Assigned an octal number to represent the operating system on which IND is running. The octal numbers represent one of the following: 0 RSX–11D 1 RSX–11M 2 RSX–11S 3 IAS 4 RSTS 5 VAX/VMS 6 RSX–11M–PLUS 7 RT–11 (SJ monitor) 10 RT–11 (FB monitor) If <MAPPED> is true, the XM monitor is running.
<SYUNIT>	Assigned the unit number of the user's system device (SY:).
<WARNIN>	Permanently assigned the value 0 (octal) to represent a warning.

(Continued on next page)

Table 5–6: IND Special Symbols (Cont.)

Symbol	Value
String Symbols	
<DATE>	Assigned the current date; format is dd-mmm-yy. The length of the value of <DATE> is predetermined (nine characters). If date is defined with less than nine characters, the value is padded with blank characters in the symbol table. If no date has been assigned, <DATE> contains nine blank characters.
<EXSTRI>	Assigned the physical device name, device size, and attributes of a device tested with the .TESTDEVICE directive. The following mnemonics are found in <EXSTRI>:
	NSD No such device LDD Device handler loaded UNL Device handler not loaded ONL On line OFL Off line UNK Unknown MTD Mounted NMT Not mounted
	For word positions, see .TESTDEVICE description, Section 5.5.85.
<FILSPC>	Assigned the full file specification (device, file name, and file type) of the file tested with the .TESTFILE directive, opened with the .OPEN, .OPENA, or .OPENR directive, or opened when a control file is called to begin execution.
<MONNAM>	Assigned the name of the currently running monitor. The length of the value of this special string symbol is predetermined (six characters). Therefore, if <MONNAM> is defined with less than six characters, the value is padded with blank characters in the symbol table.
<SYDISK>	Assigned the device mnemonic of the user's system device.
<TIME>	Assigned the current time; format is hh:mm:ss.

5.4.8 Symbol Value Substitution

Symbol value substitution is a means of replacing a symbol you use in a command line with that symbol's contents. When you use the .ENABLE SUBSTITUTION directive and enclose a symbol in apostrophes, IND replaces that symbol with the value assigned to it. This process is known as substitution. You can enable substitution in any line of a control file. Using substitution, you have a quick way to print character strings and prompts at the console, and you can manipulate symbol values neatly and efficiently in your control files.

If IND encounters an apostrophe when substitution is enabled, IND treats the subsequent text, up to a second apostrophe, as a symbol name. IND then substitutes the value of the symbol in the command line in place of the symbol. You can also use substitution within the comments that IND prints at the console.

The following example illustrates substitution. The lines below appear in a control file.

```
.ENABLE SUBSTITUTION
.ASKS DEV ENTER INPUT DEVICE
ASSIGN 'DEV' INP
```

When IND processes the above lines, it prints the following at the console:

```
* ENTER INPUT DEVICE [ST: DY
.ASSIGN DY INP
```

In the example above, DY is the response to the displayed prompt. This reply assigned the string value DY to string symbol DEV. Then when IND read ASSIGN 'DEV' INP, it substituted for 'DEV' the value assigned to DEV; that is, DY. If substitution mode was not enabled, IND would simply have passed the line to KMON as it appeared in the control file (that is, ASSIGN 'DEV' INP).

If substitution mode is enabled, an apostrophe signals the beginning of a string symbol. Thus, to include an apostrophe as text within a command line, rather than as the start of a symbol, you must replace the single apostrophe with two consecutive apostrophes ('). For example, if substitution mode is enabled, IND displays the control file line:

```
!THE SYMBOL'S VALUE
```

as:

```
!THE SYMBOL'S VALUE
```

5.5 IND Directives

The sections that follow give detailed information on the IND directives and also provide examples.

NOTE

In the sections that follow, unless specified otherwise, square brackets ([ ]) that enclose directive arguments indicate that the arguments are optional. They are not part of the directive syntax.

5.5.1 Define a Label (.label:)

The .label: directive assigns a name to a location in your control file so that the location can be easily referenced. The .label: directive has the following syntax:

```
.label:
```

where:

label represents the name you want to assign to a location in your control file

Labels must be from one to six characters, prefixed with a period (.) and followed by a colon (:). Labels must always appear at the beginning of a line.

You can use labels as reference points in your control files. A label is known only within the level of the indirect control file in which the label is defined.

5.5.1.1 Label Processing — When your control file instructs IND to branch to a label, IND first determines whether or not the label is a direct access label (See Section 5.5.1.2). If not, IND searches for the label from the current position in the file to the end of the file. If the label is not found, IND searches from the beginning of the file toward the current position. When IND finds the label, processing continues on that line. If IND cannot find the label it prints an error message.

5.5.1.2 Direct Access Labels — Direct access labels are labels IND can branch to quickly. You can define a direct access label by placing a label on a line by itself. When processing a control file, IND recognizes these labels as direct access labels and records the label and its location in an internal table. When a direct access label is referenced, IND checks the direct access table and jumps directly to the proper location without having to search the file. IND then continues processing at the statement directly below the direct access label.

You can define up to 20 direct access labels within an indirect control file. If you define more than 20, the newly defined labels replace the already defined labels in order, beginning with the first direct access label defined. Direct access labels that are replaced are thereafter treated as nondirect access labels.

In the following example, 100 is a direct access label, while 200 is not:

```
.100;
          .THIS IS THE START OF A SUBROUTINE
                    .
                    .
                    .
          .RETURN
.200      .ASK A DO YOU WANT TO CONTINUE
          .IFT A .GOSUB 100
```

5.5.2 Define Logical End of File (/)

The logical end-of-file directive, the slash character (/), terminates file processing. You cannot assign an exit status value when you use the slash character. The slash character performs the same function as the .STOP directive (see Section 6.5.33).

When IND encounters the slash, it prints the following message at the console:

```
@ <EOF>
```

IND ignores any characters that follow the slash on the same line. You can use this directive at any location in the control file to quickly terminate file processing.

The following example uses the end-of-file directive:

```
          .ASK CONT DO YOU WISH TO CONTINUE
          .IFT CONT .GOTO 100
          /
.100:
```

In the last example, execution halts if the logical symbol CONT is defined as false.

5.5.3 .ASK Directive

The .ASK directive sets the value of a specified logical symbol to true or false. The .ASK directive prints a question at the console, waits for a yes or no response, and sets a specified logical symbol to a value of true for yes or false for no. If the symbol has not already been defined, IND makes an entry in the symbol table. If the symbol has been defined, IND resets its value according to the response. IND prints an error message and exits if the symbol was previously defined as a numeric or string symbol.

5.5.3.1 Syntax – The .ASK directive has the following syntax:

.ASK [def:time] logsym prompt

where:

def : time — represents the default response that is used when only a carriage return is entered as a response, or when a specified time interval elapses and no response is given. Specify the default response by using a logical or special symbol (such as <TRUE> or <FALSE>) that is assigned a true or false value. If no default response is specified, the default response is no.

time — represents the timeout count. If the timeout count is exceeded and no response is given, IND uses the default response and the special symbols <DEFAULT> and <TIMOUT> are set to true. A timeout count can be used only if you have enabled timeout mode by means of the .ENABLE TIMEOUT directive. Also, your configuration must include a system clock and your monitor must include timer support. If you specify a timeout count in a control file, and any of these conditions is not met, the timeout indicator is not displayed in the resulting prompt and the timeout count is ignored.

The timeout count syntax is nnu. The variable nn represents the number of time units to count before the timeout occurs, and u represents one of the following time units:

T	Ticks
S	Seconds
M	Minutes
H	Hours

IND interprets nn as an octal number unless you use a decimal point (.) following the number to denote decimal. If you specify an invalid timeout parameter, an error occurs.

logsym represents a logical symbol to be set to true or false.

prompt represents the question to be printed at the console. The prompt you specify can include up to 80 characters.

The brackets surrounding the optional parameters def and time are part of the syntax; you must include them if you specify a value for either parameter. Although both parameters are optional, they are position-dependent within the brackets. If you specify time without specifying a default, you must delimit the position of the default parameter with a colon (:).

The following command line specifies a timeout count but no default response. If no response is given within 15 (decimal) seconds, IND assigns the value false (for no) to the logical symbol DONE.

```
>ASK [:15.S] DONE ARE YOU FINISHED
```

5.5.3.2 Question Display -- When IND processes an .ASK directive in a command line, IND prints an asterisk followed by a space (*), the question you specified (prompt), and a question mark (?), followed by response information, taken from the optional parameters, in brackets. For example, when IND processes the command line shown above, IND prints:

```
* ARE YOU FINISHED? [Y/N D:N T:15.S]
```

The Y/N indicates that a yes or no response is required. The notation D:N indicates that the default response is no. The notation T:15.S indicates that the default response will be used if no response is given within 15 (decimal) seconds.

5.5.3.3 Responses -- IND interprets any string that begins with a Y to mean yes, and sets the specified logical symbol to a value of true. IND interprets any string that begins with an N to mean no, and the logical symbol is set to a value of false. A response that begins with any other character causes IND to reprint the question.

If only a carriage return is typed in response to the question, IND uses the default response indicated within the brackets. This response also sets <DEFAUL> to true.

If the response [ESC] is typed while escape recognition is enabled (with the .ENABLE ESCAPE command), the special symbol <ESCAPE> is set to true and the symbol is set to true if it is undefined. However, if the symbol has been previously defined its value remains unchanged. If the response [ESC] is typed while escape recognition is disabled (by the .DISABLE ESCAPE directive), <ESCAPE> is set to false and IND prints an error message.

The response CTRL/Z causes IND to print the following message, then terminate processing:

^Z[EOF]

5.5.4 .ASKN Directive

The .ASKN directive sets the value of a specified numeric symbol to a numeric value. The .ASKN directive prints a question or prompt at the console, waits for a numeric response, and sets the specified numeric symbol to the value of the response. If the symbol has not already been defined, IND makes an entry in the symbol table. If the symbol has been defined, IND resets its value according to the response and the default radix mode enabled (octal or decimal). IND prints an error message and exits if the symbol was previously defined as a logical or string symbol.

5.5.4.1 Syntax – The .ASKN directive has the following syntax:

.ASKN [low:high:def:time] numsym prompt

where:

low:high — represents the inclusive numerical range within which the response must fall. The default range is 0 through 177777 (octal), or 0 through 65535 (decimal). If you specify values for low and high, they must fall within the default range. You can specify these values as numbers or as numeric expressions.

def — represents the default response that is used when only a carriage return is entered as a response, or when a specified time interval elapses and no response is given. You can specify the default response either as a number or as a numeric expression. If no default response is specified, the default response is assigned the value of the low limit of the range (either the assigned range or the default range if none is assigned).

time — represents the timeout count. If the timeout count is exceeded and no response is given, IND uses the default response and the special symbols <DEFAULT> and <TIMOUT> are set to true. A timeout count can be used

only if you have enabled timeout support by means of the .ENABLE TIMEOUT directive. Also, your configuration must include a system clock and your monitor must include timer support. If you specify a timeout count in a control file, and any of these conditions is not met, the timeout indicator is not displayed in the resulting prompt and the timeout count is ignored.

The timeout count syntax is nnnu. The variable nn represents the number of time units to count before the timeout occurs, and u represents one of the following time units:

T	Ticks
S	Seconds
M	Minutes
H	Hours

If you specify an invalid timeout count, an error occurs.

numsym represents a numeric symbol to be assigned the value of the response

prompt represents a string of characters to be printed at the console. If the prompt is a question, you must include the question mark as part of prompt. The prompt you specify can include up to 80 characters.

The brackets surrounding the optional parameters low, high, def and time are part of the syntax; you must include them if you specify a value for any of these parameters. Although all four parameters are optional, they are position-dependent within the brackets. You must use a colon to delimit the position of any parameter you exclude if you want to specify a parameter that follows it within the brackets.

The following command line specifies high value for the range and a timeout count, but no low limit or default response. If no response is given within 15 seconds, IND assigns the value 0 (the default value of the low limit) to the numeric symbol NUM.

```
-ASKN 0:7::15 NUM # OF LINEPRINTERS IN CONFIGURATION?
```

5.5.4.2 Determining the Radix — The radix is determined by the .ENABLE/DISABLE OCTAL directive. The radix affects how range, default, and timeout indicators are displayed in a resulting prompt, and the radix of the response.

If the default radix, octal, is in effect IND considers the indicators you specify to be octal. However, if decimal mode is in effect (the .DISABLE OCTAL directive has been issued), IND interprets the indicators you specify as decimal.

You can override octal mode by placing a decimal point (.) after any of the values you specify within the brackets. IND considers values that you

specify with a decimal point (.) to be decimal values. Any values within the same set of brackets specified without a decimal point are interpreted as octal, but IND converts them to their decimal equivalents before printing the resulting prompt. When decimal mode is in effect, all values specified within the brackets are considered decimal; using a decimal point (.) has no effect.

For example, in the following control file octal mode is enabled. When IND processes the .ASKN directive, IND interprets the default and timeout values as decimal numbers. However, IND converts the range values from octal to decimal.

```
.ENABLE OCTAL
.ASKN D:10:3.:20.$? ERR NO. OF ERROR CODES TO USE
```

Therefore, the valid range for the response is 0–8 (decimal), the default response is 3 (decimal), and the timeout count is 20 (decimal) seconds.

When you use numeric symbols or expressions to specify the range or default response, the radix of the numeric symbols determines the radix of the range and default values.

5.5.4.3 Question Display – When IND processes an .ASKN directive in a command line, IND prints an asterisk followed by a space (*) and the prompt you specified, followed by response indicators taken from the optional values you specify, in brackets. For example, when IND processes the command line shown above, IND prints:

```
* NO. OF ERROR CODES TO USE (0:R:0.-8., D:D., T:20.S)
```

Since decimal values were specified within the brackets in the original command line, all values within the resulting prompt are shown as decimal. (The decimal points (.) following the values for the range, default response, and timeout count indicate that these are decimal numbers.) The notation R:0.–8. indicates that the value must be a number in the range 1 to 8 (decimal) inclusive; the notation D:3. indicates that the default response is 3; and the notation T:20.S indicates that the default response will be used if no response is given within 20 (decimal) seconds.

The D indicates that for this example the default radix for the response is decimal, and that response will be always stored as a decimal number.

If octal values had been specified in the original command line, IND would print the O (octal) indicator instead, meaning the default radix of the response is octal and the response will be stored as an octal value.

5.5.4.4 Responses – The response to an .ASKN directive must be an octal or decimal number within the range specified by the prompt. The O or D radix indicator tells you the radix in which the response is stored.

If the radix indicator within the prompt brackets is O, IND assumes the response is an octal value. You can specify a decimal value by typing a decimal point (.) after the response. IND stores the response as its octal equivalent.

If the radix indicator is D, IND assumes the response is a decimal value. You can specify an octal value by typing a number sign (#) before your response. IND stores the response as its decimal equivalent.

If only a carriage return is typed in response to the question, IND uses the default response indicated within the brackets. This response also sets <DEFAULT> to true.

If the response ESC is typed while escape recognition is enabled (with the .ENABLE ESCAPE command), the special symbol <ESCAPE> is set to true and the numeric symbol is set to 0 if the symbol has not yet been defined. If the symbol was previously defined, its value remains unchanged. However, if escape recognition is disabled, <ESCAPE> is set to false and IND prints an error message and reprompts for a valid response.

The response CTRL/Z causes IND to print the following message, then terminate processing:

%EOF

5.5.5 .ASKS Directive

The .ASKS directive sets the value of a specified string symbol to an ASCII string. The .ASKS directive prints a question or prompt at the console, waits for an ASCII string response, and sets the specified string symbol to the value of the response. If the symbol has not already been defined, IND makes an entry in the symbol table. If the symbol has been defined, IND resets its value according to the response. IND prints an error message and exits if the symbol was previously defined as a logical or numeric symbol.

5.5.5.1 Syntax – The .ASKS directive has the following syntax:

.ASKS [low:high:"def":time] strsym prompt

where:

low:high	represents the inclusive number of characters permitted in the response string. The default range is 0 through 204 (octal), or 0 through 132 (decimal). If you specify values for low and high, they must fall within the default range. You can specify these values as numbers or as numeric expressions.
"def"	represents the default response that is used when only a carriage return is entered as a response, or when a specified time interval elapses and no response is given. You

can specify the default response either as a string, string symbol, or string expression. The quotation marks are part of the directive syntax if you specify a string.

time — represents the timeout count. If the timeout count is exceeded and no response is given, IND uses the default response and the special symbols <DEFAUL> and <TIMOUT> are set to true. A timeout count can be used only if you have enabled timeout support by means of the .ENABLE TIMEOUT directive. Also, your configuration must include a system clock and your monitor must include timer support. If you specify a timeout count in a control file, and any of these conditions is not met, the timeout indicator is not displayed in the resulting prompt and the timeout count is ignored. If these conditions are all met and you specify a timeout count but no default response, an error results and IND will not execute the command line.

The timeout count syntax is nnu. The variable nn represents the number of time units to count before the timeout occurs, and u represents one of the following time units:

T	Ticks
S	Seconds
M	Minutes
H	Hours

If you specify an invalid timeout count, an error occurs.

strsym — represents a string symbol to be assigned the value of the response

prompt — represents a string of characters to be printed at the console. If the prompt is a question, you must include the question mark as part of prompt. The prompt you specify can include up to 80 characters.

The brackets surrounding the optional parameters low, high, def, and time are part of the syntax; you must include them if you specify a value for any of these parameters. Although all four parameters are optional, they are position-dependent within the brackets. You must use a colon to delimit the position of any parameter you exclude if you want to specify a parameter that follows it within the brackets.

The following command line specifies a low value for the range, a default response, and a timeout count. Since no high limit is specified, the default high limit 204 (octal) is assumed. If no response is given within 15 seconds, IND assigns the value DY0 to the string symbol DEV.

```
.ASKS DEV,"DY0"[,155] DEV DEVICE TO USE FOR DEFAULT?
```

5.5.5.2 Determining the Radix of Range and Timeout Values — The radix of the range and timeout values is determined by the .ENABLE/DISABLE OCTAL directive. The radix of these values determines how they are displayed in the resulting prompt.

If the default radix, octal, is in effect IND considers the numbers you specify to be octal. However, if decimal mode is in effect (the .DISABLE OCTAL directive has been issued), IND interprets the values you specify as decimal.

You can override octal mode by placing a decimal point (.) after any of the values you specify within the brackets. IND considers values that you specify with a decimal point (.) to be decimal values. Any values within the same set of brackets specified without a decimal point are interpreted as octal, and IND converts them to their decimal equivalents. When decimal mode is in effect, all values specified within the brackets are considered decimal; using a decimal point (.) has no effect.

For example, in the following control file octal mode is enabled. When IND processes the .ASKS directive, IND interprets the timeout value as a decimal number. However, IND converts the range values from octal to decimal.

```
.ENABLE OCTAL
.ASKS LOG:D:"RT11A":20,S:1 LOG TO TYPE YOUR VOLUME ID
```

Therefore, the response must contain from 0 (decimal) to 8 (decimal) ASCII characters, the default response is RT11A, and the timeout count is 20 (decimal) seconds.

When you use numeric symbols or expressions to specify the range, the radix of the numeric symbols determines the radix of the range.

5.5.5.3 Question Display — When IND processes an .ASKS directive in a command line, IND prints an asterisk followed by a space (*) and the prompt you specified, followed by response information taken from the optional parameters, in brackets. For example, when IND processes the command line shown above, IND prints:

```
* TYPE YOUR VOLUME ID (S R:0.-8, D:"RT11A" T:20.S)
```

The S indicates that the response must be an ASCII string. The notation R:0.-8 indicates that the response string can be from 0 to 8 (decimal) characters long, inclusive. The notation D:"EBCDIC" indicates that the default volume ID is "EBCDIC". (If no default response was specified, the D: is not displayed.) The notation T:20.S indicates that the default response will automatically be used if no response is given within 20 (decimal) seconds. Note that IND indicates decimal values by printing a decimal point (.), and indicates octal values by excluding the decimal point.

5.5.5.4 Responses — The response to an .ASKS directive must be an ASCII string whose length is within the range specified by the prompt.

If only a carriage return is typed in response to the question, IND uses the default response indicated within the brackets. This response also sets <DEFAULT> to true. If no default response has been specified, the symbol is set to null.

If the response ESC is typed while escape recognition is enabled (with the .ENABLE ESCAPE command), the special symbol <ESCAPE> is set to true and the symbol is defined as null (if not previously defined). If the symbol was previously defined, the definition remains unchanged. However, if escape recognition is disabled, <ESCAPE> is set to false and IND prints an error message and reprompts for a valid response.

The response CTRL/Z causes IND to print the following message, then terminate processing:

```
$EOF
```

5.5.6 Begin Block (.BEGIN)

The .BEGIN and .END directives permit you to structure the control file in blocks. Modular, block-structured control files are easy to debug and maintain. More importantly, begin-end blocks isolate local symbol definitions and thus conserve symbol table space. When you define a symbol, IND creates an entry in an internal symbol table.

The symbol table entries retain their definitions throughout the control file execution if defined locally, or throughout all levels of control files if defined globally. Local symbols defined within a block, however, are defined only within that block; they are erased from the symbol table when IND encounters an .END directive. Thus, if a symbol is defined as a logical, numeric, or string symbol outside of a begin-end block, you can redefine the symbol to another type within the begin-end block. However, when you exit from the begin-end block, the redefined symbol is erased and the symbol returns to its previous type.

The .BEGIN directive marks the beginning of a begin-end block. All local symbols following the directive are local to the block instead of to the entire control file. The .ERASE LOCAL directive erases all local symbols within the block.

Begin-end blocks can be nested up to a maximum depth of 127, but IND usually exhausts stack space before this limit can be reached.

The .BEGIN directive has the following syntax:

```
.BEGIN
```

Anything that follows a .BEGIN directive on the same line is ignored.

The block must be terminated by an .END directive. Each .BEGIN directive must have a corresponding .END directive.

5.5.7 Chain to Another File (.CHAIN)

The .CHAIN directive closes the current indirect control file, disregards all current local symbols, and continues processing by using commands from another indirect control file. The .CHAIN directive does not close data files or change the control file level.

The .CHAIN directive has the following syntax:

.CHAIN *filespec* [/options]

where:

filespec represents the indirect control file to which control is to be passed

/options represents one of the options described in Section 5.2.1

In the following example, IND passes control to the file DK:OUTPUT.COM:

```
.CHAIN OUTPUT
```

5.5.8 Close File (.CLOSE)

The .CLOSE directive closes the file opened by the .OPEN, .OPENA, and .OPENR directives. You must close any open files before passing control from IND to the keyboard monitor. The .CLOSE directive disables data mode. When you use the .CLOSE directive, make sure you use a file number with the file specification.

The .CLOSE directive has the following syntax:

.CLOSE [*#n*] [*filespec*]

where:

n represents an optional file number from 0 to 8 (the default file number is 0). If substitution is enabled, you can substitute a symbol for *n* by enclosing the symbol in apostrophes.

filespec represents the name of the file you are closing

Using a file specification with .CLOSE causes no action but can make your control file more readable.

If you use the .CLOSE directive after .OPENR, .CLOSE has the same effect as the .PURGE directive.

5.5.9 Send Data to File (.DATA)

The .DATA directive writes a record to a file previously opened by an .OPEN or .OPENA directive.

The .DATA directive has the following syntax:

.DATA [*#n*] *text-string*

where:

n	represents an optional file number from 0 to 3 (the default file number is 0). If substitution is enabled, you can substitute a symbol for n by enclosing the symbol in apostrophes.
text-string	represents text to be sent to the output file. (If substitution is enabled, text-string can be a string symbol in apostrophes.) You can send blank lines as text to an output file, as well as characters.

The .DATA command line cannot exceed 132 characters.

In the following example, IND sends the string THIS IS DATA to the output file TEMP.DAT (file number 0):

```
.OPEN TEMP
.DATA THIS IS DATA
.CLOSE
```

In the next example, IND sends the output file COMMAN.DAT (file number 1):

```
.OPEN #1 COMMAN
.DATA #1 .DISABLE DATA
.CLOSE #1
```

The .DATA directive is also useful for creating indirect files that execute commands that use more than one line. The following example creates and executes an indirect command file that runs PIP, unprotects the file DY:FILE.TST, and copies the file to DY1:

```
.OPEN COPY.INP
.DATA RUN PIP
.DATA DY:FILE.TST/Z
.DATA DY:FILE.TST=DY1:FILE.TST/W/Y
.DATA ^C
.CLOSE
$@COPY.INP
```

5.5.10 Decrement Numeric Symbol (.DEC)

The .DEC directive decrements a numeric symbol by one.

The .DEC directive has the following syntax:

.DEC numsym

where:

numsym	represents the numeric symbol to be decremented.

IND prints an error message and terminates processing if you use a logical or string symbol with .DEC.

5.5.11 Delay Execution (.DELAY)

The .DELAY directive delays control file processing for a specified period of time. The directive is valid only if your monitor includes timer support and your configuration includes a clock.

The .DELAY directive has the following general syntax:

.DELAY nnu

where:

nn represents the number of time units for which you wish to delay execution

u represents one of the following time units:

T	Ticks
S	Seconds
M	Minutes
H	Hours

IND interprets the number you specify for nn as octal, unless you use a decimal point to denote decimal.

When .DELAY suspends execution, IND prints the following message at the console:

```
Delaying...
```

When execution resumes, IND prints the following at the console:

```
...Continuing
```

In the following example, .DELAY delays execution for 8 seconds (10 octal):

```
.DELAY 10S
```

In the following example, .DELAY delays execution for 25 (decimal) seconds.

```
.DELAY 25.S
```

5.5.12 Disable Option (.DISABLE)

The .DISABLE directive disbles a specified operating mode. See the .ENABLE directive for details on enabling operating modes.

The .DISABLE directive has the following syntax:

.DISABLE op-mode[,op-mode,op-mode...]

where:

op-mode represents one or more of the following operating modes:

ABORT	GLOBAL	SUBSTITUTION
CONTROL-Z	LOWERCASE	SUFFIX
DATA	MCR	TIMEOUT
DCL	OCTAL	TRACE
DELETE	PREFIX	TYPEAHEAD
ESCAPE	QUIET	

Each operating mode is independent of the others; all can be disabled at the same time. You can disable more than one operating mode with one .DISABLE directive by separating the operating modes with commas. The DATA operating mode is an exception; the .DISABLE DATA directive must appear on a line by itself. If you disable more than one operating mode on one line and an error occurs, none of the operating modes are disabled.

In the following example, the SUBSTITUTION and GLOBAL operating modes are disabled with one directive.

```
.DISABLE SUBSTITUTION,GLOBAL
```

Operating modes can be local or global in scope. Local operating modes automatically return to their default settings when you enter a nested control file. You must explicitly enable or disable local operating modes when you enter a nested file. Conversely, when you return from a nested control file to the previous level file, the operating mode returns to its previous setting as specified for that file. However, global operating modes remain enabled or disabled throughout all levels of control files until you explicitly change the setting. Table 5–7 in Section 5.5.14 lists the scope of each operating mode.

The .DISABLE DATA directive requires special treatment. Like other .DISABLE directives, you can place the .DISABLE DATA directive in any column of your control file. Therefore, you can format the control file by using spaces and tabs before the .DISABLE DATA directive on a command line. However, .DISABLE DATA must be the first and only command on the line, and you may not use labels on the same command line with a .DISABLE DATA directive.

When IND is processing a control file and you type CTRL/O, IND suppresses terminal output until it encounters a .DISABLE QUIET directive.

5.5.13 Display Symbol Table (.DUMP)

The .DUMP directive displays the contents of the local, global, or special symbol table, or displays the contents of all symbol tables. See Section 5.4 for a description of local, global, and IND special symbols.

The .DUMP directive has the following syntax:

```
.DUMP [symboltable]
```

where:

symboltable	represents the symbol table whose contents you want to display: LOCAL, GLOBAL, or SPECIAL. If you do not specify a symbol table, IND prints the contents of all three.

IND first indicates what type of symbols will be displayed. When displaying the contents of all symbol tables, IND lists the special symbols first, followed by the global symbols and lastly the local symbols. Each line of the symbol table display is formatted as follows:

SYMBOL(TYPE): VALUE

where:

SYMBOL	represents the symbol name
TYPE	represents the type of symbol:

L	Logical symbol
O	Octal numeric symbol
D	Decimal numeric symbol
S	String symbol

VALUE	represents the symbol's value: T or F for a logical symbol, a number for a numeric symbol, or an ASCII string in quotation marks for a string symbol.

Local symbols are displayed in reverse order of definition; the last local symbol defined is listed first. If you use the .DUMP LOCAL directive and the local symbol table is empty, IND prints:

```
*****There are no local symbols***
```

Global symbols are listed in order of definition; the first global symbol defined is listed first. If you use the .DUMP GLOBAL directive and the global symbol table is empty, IND prints:

```
****There are no global symbols****
```

IND may also print one or both of these messages when displaying all three symbol tables (the .DUMP directive with no argument). Note, however, that the special symbol table is never empty because IND special symbols are permanent symbols.

The following example displays the contents of all three symbol tables.

```
.dump
```

```
Special symbols:

MAPPED(L): F
ALTMOD(L): F
ESCAPE(L): F
DEFAUL(L): F
RADIX50(L): F
ALPHAN(L): F
EOF    (L): F
FALSE  (L): F
TIMOUT(L): F
TRUE   (L): T
OCTAL  (L): T
SPACE  (D): 7680
SYMTYP(D): 0
FILERR(D): 0
STRLEN(D): 1
SYUNIT(D): 0
EXSTAT(D): 1
SUCCES(D): 1
WARNIN(D): 0
ERROR  (D): 2
SEVERE(D): 4
SYSTEM(D): 7
MONNAM(S): "RTLISJ"
SYDISK(S): "DL"
DATE   (S): "20-MAR-90"
TIME   (S): "00:06:14"
FILSPC(S): "TT:"
EXSTRI(S): ""

Global symbols:

                    ****There are no global symbols****

Local symbols:

NAME   (S): "JON"
UNITS  (D): 4
BUSIND(L): T
P8     (S): ""
P7     (S): ""
P6     (S): ""
P5     (S): ""
P4     (S): ""
P3     (S): ""
P2     (S): ""
P1     (S): ""
P0     (S): "TT:"
COMMAN(S): "TT:"
```

5.5.14 Enable Option (.ENABLE)

You can use the .ENABLE directive to invoke one of the operating modes listed in Table 5–7.

The .ENABLE directive has the following syntax:

```
.ENABLE op-mode[,op-mode,op-mode...]
```

where:

op-mode represents one or more of the operating modes.

Each mode is independent of the others; all can be active simultaneously. You can enable more than one operating mode with one .ENABLE directive by separating the operating modes with commas. The DATA operating mode is an exception; the .ENABLE DATA directive must appear on a line by itself. If you enable more than one operating mode on one line and an error occurs, none of the operating modes are enabled.

In the following example, the LOWERCASE and TIMEOUT operating modes are enabled with one directive.

```
.ENABLE LOWERCASE,TIMEOUT
```

Operating modes can be local or global in scope. Local operating modes automatically return to their default settings when you enter a nested control file. You must explicitly enable or disable local operating modes when you enter a nested file. Conversely, when you return from a nested control file to the previous level file, the operating modes return to their previous settings as specified for that file. However, global operating modes remain enabled or disabled throughout all levels of control files until you explicitly change the settings.

Table 5–7 lists the operating mode default settings and the scope of each mode.

Table 5–7: IND Operating Modes

Mode	Default	Scope
ABORT	Enabled	Global
CONTROL-Z	Enabled	Global
DATA	Disabled	Local
DCL	Enabled	Local
DELETE	Disabled	Local
ESCAPE	Disabled	Global
GLOBAL	Disabled	Global
LOWERCASE	Enabled	Global
MCR	Enabled	Local
OCTAL	Enabled	Global
PREFIX	Enabled	Global
QUIET	Disabled	Local
SUBSTITUTION	Enabled	Global
SUFFIX	Enabled	Global
TIMEOUT	Disabled	Global
TRACE	Disabled	Local
TYPEAHEAD	Enabled	Global

Descriptions and examples of each of the operating modes follow.

- **Abort Mode (ABORT)**

  The IND directives .ENABLE ABORT and .DISABLE ABORT enable and disable abort mode. In abort mode, you can abort a control file by typing a double CTRL/C. If you disable abort mode (.DISABLE ABORT),

IND ignores CTRL/C characters typed at the terminal. Abort mode remains disabled until the currently executing control file (including control files nested within it) exits or until you issue the .ENABLE ABORT directive. The default setting is .ENABLE ABORT.

.DISABLE ABORT is available only if you have included global .SCCA support in your monitor through the system generation process. If you issue the .DISABLE ABORT directive and your monitor does not include global .SCCA support, IND returns the value 0 (for warning) in the special symbol <EXSTAT> when the directive is processed.

In the following example, double CTRL/C aborts are disabled during an INITIALIZE operation.

```
.ASKS DEV What disk do you want to initialize?
.IF DEV EQ "" .GOTO 10
.!Disable abort so user cannot abort INITIALIZE
.!with double CTRL/C
.DISABLE ABORT
!@INITIALIZE/BADBLOCKS/RECOVERY 'DEV':
.ENABLE ABORT
.10:
```

Abort mode also enables or disables the ability to abort a control file by typing CTRL/Z in response to an .ASK, .ASKN, or .ASKS directive.

- **Control-Z Mode (CONTROL-Z)**

The IND directives .ENABLE CONTROL-Z and .DISABLE CONTROL-Z enable and disable CONTROL-Z mode. When CTRL/Z is enabled, a <CTRL/Z> typed in response to an .ASK, ASKN, or .ASKS prompt aborts the control file. If you disable CTRL/Z (.DISABLE CONTROL-Z), IND ignores <CTRL/Z> characters typed at the terminal. CTRL/Z remains disabled until the currently executing control file (including control files nested within it) exits or until you issue the .ENABLE CONTROL-Z directive. The default setting is .ENABLE CONTROL-Z.

- **Data mode (DATA [#*n*])**

In data mode, IND sends to an output file lines that follow the directive line .ENABLE DATA. (To send a single line of text to a file, see the .DATA directive description.) When you use .ENABLE DATA, blank lines are ignored; this allows you to format your control file.

In the .ENABLE DATA [#*n*] directive, *n* represents an optional file number in the range 0–3. (The default is 0.) If substitution is enabled, you can substitute a symbol for the value *n* by enclosing the symbol in apostrophes.

When the control file contains:

```
.OPEN SECFIL.DAT
.ENABLE DATA #'NUM'
    .
    .
    .
.DISABLE DATA #'NUM'
```

IND writes the lines that fall between the .ENABLE and .DISABLE directives to the file SECFIL.DAT.

NOTE

If you have enabled data mode for one file and wish to send data to a second file, you must disable data mode for the first file before you enable it for the second file. If you fail to disable data mode for the first file, the data you direct to the second file is sent to the first file.

Data mode is also useful for creating indirect files that execute commands that use more than one line. The next example creates an indirect file that runs PIP, unprotects the file DY:FILE.TST, and copies the file to DY1:.

```
.OPEN COPY.TMP
.ENABLE DATA
RUN PIP
DY:FILE.TST/Z
DY:FILE.TST=DY1:FILE.TST/M/Y
^C
.DISABLE DATA
.CLOSE
$@COPY.TMP
```

- **DCL command mode (DCL)**

In DCL mode, IND passes lines it does not recognize to the keyboard monitor to be executed. When the control file contains:

```
.ASKS DEV WHICH DEVICE WILL YOU USE FOR THE LOG FILE? DY0:
ASSIGN 'DEV' LOG
```

and DCL mode is enabled, the keyboard command ASSIGN DY0: LOG is executed. If DCL mode is disabled, or if the /N option was used, this command is ignored.

- **Delete mode (DELETE)**

When delete mode is enabled in a control file, the control file is deleted when IND is through processing it. Processing is complete when IND executes the .EXIT directive or reaches the end of the control file.

The .ENABLE DELETE directive has the same effect as the /D option.

- **Escape recognition mode (ESCAPE)**

Escape recognition permits the escape character to be a valid response to an .ASK, .ASKS, or .ASKN directive. A question answered with a single escape character sets the special logical symbol <ESCAPE> to true. The escape character, followed by a carriage return, must be used only as an immediate terminator to the question: if one or more characters precede or follow the escape, IND will print the following error message:

```
?IND-E Invalid Answer or Terminator
```

IND will then repeat the query. If you type ESC in response to an .ASK directive, the specified logical symbol will be set to true if the symbol has not previously been defined; otherwise, it remains unchanged.

When the control file contains:

```
!IF YOU WANT A LIST OF OPTIONS, TYPE <ESC><RET>
.ENABLE ESCAPE
.ASKS A ENTER OPTION
.IFT <ESCAPE> .GOTO LIST
        .
        .
        .
.LIST: !OPTIONS ARE: A (ADD), S (SUBTRACT), D (DIVIDE)
```

and you type the ESC key, followed by a carriage return, in response to ENTER OPTION, the corresponding lines displayed at the terminal are:

```
!IF YOU WANT A LIST OF OPTIONS, TYPE <ESC><RET>
> ENTER OPTION [S]: <ESC><RET>
!OPTIONS ARE: A (ADD), S (SUBTRACT), D (DIVIDE)
```

- **Global symbol mode (GLOBAL)**

In global symbol mode, symbol names that begin with a dollar sign ($) are defined as global to all levels of control files; once such a symbol has been defined, all levels recognize it. Symbols that do not begin with a dollar sign are local to the level that defines them.

The file LORRAN.COM contains the following lines:

```
.ENABLE SUBSTITUTION
!'$X'
```

When the control file contains:

```
.ENABLE GLOBAL
.SETS $X "TEST"
@LORRAN.COM
```

IND prints at the console:

```
.TEST
```

- **Lowercase mode (LOWERCASE)**

When lowercase mode is enabled, IND stores responses to .ASKS directive prompts, and strings that define symbols with the .SETS directive, in the string symbol as the characters are typed (uppercase characters are stored in uppercase, lowercase characters are stored in lowercase). When lowercase mode is disabled, IND stores all characters as uppercase characters, regardless of whether they were typed as uppercase or lowercase characters.

Character case is significant when comparing strings; the .IF and .TEST directives discriminate between lowercase and uppercase characters, regardless of whether lowercase mode is enabled or disabled. Also, if lowercase mode is disabled and the response to a query is in lowercase, the special logical symbol <ALPHAN> will be set to false.

When the control file contains:

```
.ENABLE SUBSTITUTION,LOWERCASE
.ASKS A DEFINE STRING SYMBOL A
!'A'
```

IND prints at the console:

```
* DEFINE STRING SYMBOL, 4 (B): SQRT Subroutine
!SQRT Subroutine
```

- **MCR command mode (MCR)**

  In MCR mode, IND passes lines it does not recognize to the keyboard monitor to be executed. When the control file contains:

  ```
  .ASKS DEV DEVICE TO USE FOR LOG FILE? DY0:
  ASSIGN 'DEV' LOG
  ```

  and MCR mode is enabled, the keyboard command ASSIGN DY0: LOG is executed. If MCR mode is disabled, or if the /N option was used, this command is ignored.

- **Octal mode (OCTAL)**

  In octal mode, numeric symbols and .ASKN directive responses are interpreted as octal rather than decimal. For example, if octal mode is enabled and the control file contains the line:

  ```
  .ASKN VECTR ENTER VECTOR ADDRESS OF FIRST CONTROLLER
  ```

  IND prints the following and interprets the response to be an octal number:

  ```
  > ENTER VECTOR ADDRESS OF FIRST CONTROLLER [0]:
  ```

  The .ENABLE OCTAL directive can be overridden by specifying decimal numbers in the range specification, or by issuing the .DISABLE OCTAL directive.

- **Prefix mode (PREFIX)**

  In prefix mode, IND prints an asterisk and a space in front of all prompts resulting from .ASK, .ASKN, and .ASKS directives, and prints semicolons (;) in front of all external comment lines. For example, suppose a control file contains the following lines:

  ```
  .ENABLE PREFIX
  .ASK CONT DO YOU WANT TO CONTINUE
  .IFF CONT .GOTO SUB2              ;DO NOT CONTINUE
  !CONTINUE
  ```

  When IND processes these lines, IND prints the following on the console:

  ```
  * DO YOU WANT TO CONTINUE? [Y/N D:N]: <RET>
  !CONTINUE
  ```

  If prefix mode had been disabled, IND would print the same lines but without the asterisk-space combination and semicolon.

- **Quiet mode (QUIET)**

  In quiet mode, IND does not display keyboard command lines. The command lines are executed normally, and if they return a message or display, the message or display is printed on the console.

When the control file contains:

```
.ASK QUIET DO YOU WANT COMMAND LINES SUPPRESSED
.IFT QUIET .ENABLE QUIET
.IFF QUIET .DISABLE QUIET
ASSIGN DX DOT
```

and the response is affirmative, IND processes the ASSIGN command but does not display it on the console.

- **Substitution mode (SUBSTITUTION)**

  In substitution mode, IND replaces a symbol with its assigned value. The symbol must be enclosed by apostrophes. For example, if the string symbol A has been assigned the string value THIS IS A TEST, then every occurrence of 'A' will be replaced by THIS IS A TEST. When substitution mode is enabled, IND performs substitutions on each line before scanning the line for directives and keyboard commands.

  When the control file contains:

```
.ENABLE SUBSTITUTION
.ASKS FIL SPECIFY SOURCE FILE
MACRO 'FIL'
```

  IND prints at the console:

```
* SPECIFY SOURCE FILE [SY:]SOURCE
/MACRO SOURCE
```

- **Suffix mode (SUFFIX)**

  In suffix mode, IND prints a question mark after all .ASK prompts, and response specifications after all prompts that result from .ASK, .ASKN, and .ASKS directives. For example, suppose a control file contains the following line:

```
.ASKN [1:2,1:] INIT NO. OF DISKS TO INIT?
```

  If suffix mode is enabled, IND prints the following on the console:

```
* NO. OF DISKS TO INIT? (0 TO 1:1-0 DEF)
```

  If suffix mode is disabled, IND prints only the following:

```
* NO. OF DISKS TO INIT?
```

  Even when suffix mode is disabled, the response specifications are still used to check the validity of the response.

- **Timeout mode (TIMEOUT)**

  In timeout mode, IND recognizes timeout counts used with ASK directives, if the monitor includes timer support and the configuration includes a system clock. (If the monitor does not include timer support or your configuration lacks a system clock, IND returns the value 0, for warning, in the special symbol <EXSTAT> when the directive is processed). If timeout mode is disabled, timeout counts are ignored.

  If the control file contains the following line and timeout mode is enabled, IND waits 15 (decimal) ticks for a response before using the specified default response VT100:

```
.ASKS [,,"VT100",15,] TERM CONSOLE TYPE BEING USED?
```

If timeout mode is disabled or the monitor does not include timer support, you must enter a response or carriage return to proceed.

- **Trace mode (TRACE)**

  In trace mode, IND prints on the console each command line in a control file as the command line is processed. IND prints an exclamation mark (!) before each command line containing directives. No leading characters are placed before lines containing only keyboard monitor commands or comments.

  When trace mode is enabled, the effect is the same as using the /T option in the CSI command string.

- **Type-Ahead Mode (TYPEAHEAD)**

  The IND directives .ENABLE TYPEAHEAD and .DISABLE TYPEAHEAD enable and disable type-ahead mode. When type-ahead is enabled, IND uses characters you type before an .ASK, .ASKN, or .ASKS prompt in the response. If you disable type-ahead (.DISABLE TYPEAHEAD), IND discards characters that have been typed before processing .ASK, .ASKN, and .ASKS directives. Consequently, if you respond to a prompt before IND prints the prompt, your response is discarded. The default setting is .ENABLE TYPEAHEAD.

  In the following example, type-ahead is disabled until the directory operation is complete. Therefore, any characters typed before the .ASKS prompt is printed are discarded.

```
.DISABLE TYPEAHEAD
DIR DL:
.!Since typeahead is disabled, an answer to the following
.!question will not be accepted until the directory
.!listing has finished.
.ENABLE TYPEAHEAD
.ASKS FILE Next file or DL do you want printed?
PRINT /FILE:*
```

5.5.15 End Block (.END)

The .END directive marks the end of the begin-end block.

The .END directive has the following syntax:

```
.END
```

Anything that follows an .END directive on the same line is ignored. If IND encounters more .END directives than .BEGIN directives, IND prints an error message. (See Section 5.5.6 for more information about begin-end blocks.)

5.5.16 Delete Symbols (.ERASE)

The .ERASE directive deletes local or global symbol definitions from the symbol table. When you define a symbol, either locally or globally, IND creates an entry in a symbol table. The .ERASE directive erases either all entries in that table or specific entries.

IND permits you to redefine global and local symbols after you have used the .ERASE directive.

The .ERASE directive has the following syntax:

```
.ERASE LOCAL [symbol]
.ERASE GLOBAL [symbol]
```

where:

symbol	represents the symbol you want to erase from the specified symbol table. If you do not specify a symbol, all symbols from that symbol table are erased.

For example, the following directive erases the symbol DEV from the global symbol table:

```
.ERASE GLOBAL DEV
```

The following directive erases all global symbols from the global symbol table:

```
.ERASE GLOBAL
```

When you use .ERASE LOCAL without a symbol name, the IND internal local symbols P0–P9 and COMMAN are erased as well as the local symbols that you have defined. See Section 5.2.2 for more information on these internal local symbols.

The .DUMP directive enables you to see which symbols each symbol table contains. See Section 5.5.13 for more information on the .DUMP directive.

An .ERASE LOCAL directive outside of a begin-end block erases all local symbols. An .ERASE LOCAL directive within a begin-end block erases only those local symbols defined in that block. An .ERASE GLOBAL, either outside of or within a begin-end block, erases all global symbols.

5.5.17 Exit Current Control File (.EXIT)

The .EXIT directive terminates processing of the current control file or begin-end block and returns control to the previous level control file or begin-end block. If the directive is encountered in the first control file, IND exits and passes control to the keyboard monitor. The .EXIT directive also allows you to specify a value for the special symbol <EXSTAT>.

The .EXIT directive has the following syntax:

.EXIT [value]

where:

value — is an optional numeric expression or the value of a special symbol that is assigned to the special symbol <EXSTAT>

For example, the following line appears in indirect control file FILE1:

```
@FILE2
```

The file FILE2.COM contains the following line:

```
.EXIT
```

When IND encounters the .EXIT directive in FILE2, control returns to FILE1.COM.

If the .EXIT directive in FILE2.COM includes a numeric expression as in the following example, IND evaluates the expression and then assigns the value to <EXSTAT>.

```
.EXIT 1+2
```

When IND reaches the end of a control file, the effect is the same as executing an .EXIT directive.

5.5.18 Call a Subroutine (.GOSUB)

The .GOSUB directive saves the current location in a control file and then branches to another location, identified by a label. The label identifies an entry point to a subroutine. IND can branch to any subroutine in the current control file, regardless of begin-end blocks. The maximum nesting depth for subroutine calls is eight.

The .GOSUB directive has the following syntax:

.GOSUB label

where:

label represents the subroutine entry point

The label used with .GOSUB must not include the leading period and trailing colon.

To return from the subroutine to the calling location, use the .RETURN directive. Refer to Section 5.5.27 for more information on the .RETURN directive.

The following directive transfers control to the subroutine labeled .EVAL:

```
.GOSUB .EVAL
```

5.5.19 Branch to a Label (.GOTO)

The .GOTO directive causes a branch from one line in a control file to another line, identified by a label. All lines between the .GOTO directive and the specified label are ignored. Branches can go forward or backward in the file.

If a .GOTO directive appears in a begin-end block, its target must be in that same block. The .GOTO directive cannot branch to a nested begin-end block, but can branch to another location in the control file that appears after a nested begin-end block.

When IND encounters a .GOTO directive within a begin-end block, it scans the current begin-end block from top to bottom for the label within that block. Since the label scan starts at the .BEGIN directive and continues to the .END directive, labels that have multiple definitions are permitted within a block. IND finds the first definition of the label and branches to that location.

The .GOTO directive has the following syntax:

GOTO label

where:

label represents the name of the target label

The label used with .GOTO must not include the leading period and trailing colon.

The following example transfers control to the entry point labeled .100:

```
.GOTO 100
        .
        .
        .
.100:
```

5.5.20 Logical Tests

IND has a set of directives that perform logical tests. If the test results in a true value, IND processes the remainder of the command line.

Logical tests can be combined into a compound logical test by using the .AND and .OR directives.

5.5.20.1 Test If Symbol Meets Specified Condition (.IF) – The .IF directive compares a numeric or string symbol with another symbol of the same type to determine if one of several possible conditions is true. If the comparison yields a true value, IND executes the remainder of the command line.

When comparing string values, IND compares the value of each string's ASCII value. Because of this, IND can establish less-than, greater-than, or equal-to relationships between string values; differences between uppercase and lowercase characters are important.

The .IF directive has the following syntax:

.IF symbol operator expr action

where:

symbol	represents a numeric or string symbol
operator	represents one of the following relational operators:
expr	represents an expression of the same symbol type
action	represents how processing continues if the test results in a true value

EQ or =	Equal to
NE or <>	Not equal to
GE or >=	Greater than or equal to
LE or <=	Less than or equal to
GT or >	Greater than
LT or <	Less than

In the following example, IND compares two string values.

```
.SET$ X "A"
.SET$ Y "a"
.IF X LT Y .GOTO 200
```

In this example, the ASCII value of string symbol X is less than the ASCII value of string symbol Y, which yields the less-than condition. Thus, control passes to the line containing the label .200:.

In the following example, IND compares two numeric values. In this example, if N1 is less than or equal to N2, IND will increment N1.

```
.SETN N1 2
.SETN N2 7
.IF N1 <= N2 .INC N1
```

In the following example, IND compares the value S1 with the value of S2 concatenated with the first character of S3:

```
.SETS S1 "AB5"
.SETS S2 "AA"
.SETS S3 "BBBB"
.IF S1 >= S2+$SUBSTR(S3,1,1) .INC N
```

In this example, the .IF directive yields a true result and IND increments N.

5.5.20.2 Test if Symbol is Defined or Not Defined (.IFDF/.IFNDF) – The .IFDF and .IFNDF directives test whether a logical, numeric or string symbol has been defined (.IFDF) or not defined (.IFNDF). If the test is true, IND processes the remainder of the command line. These directives do not evaluate symbols.

The .IFDF and .IFNDF directives have the following syntax:

.IFDF symbol action

.IFNDF symbol action

where:

symbol	represents a 1- to 6-character symbol
action	represents how processing continues depending on the test results

The following example illustrates the .IFDF and .IFNDF directives:

```
.IFDF  A .GOTO 100
.IFNDF A .ASK A DO YOU WANT TO SET TIME
```

In this example, if symbol A is defined, control branches to .100:; otherwise IND prints a prompt at the console.

5.5.20.3 Test if Operating Mode is Enabled or Disabled (.IFENABLED/.IFDISABLED) – The .IFENABLED and .IFDISABLED directives test whether a specific operating mode is enabled (.IFENABLED) or disabled (.IFDISABLED). If the test is true, IND processes the rest of the directive line.

The .IFENABLED and .IFDISABLED directives have the following syntax:

.IFENABLED op-mode action

.IFDISABLED op-mode action

where:

op-mode	represents the operating mode you want to test. See Section 5.5.14 for more information on operating modes.
action	represents how processing continues depending on the test results

In the following example, control branches to COM if DCL mode is enabled:

```
.IFENABLED DCL .GOTO COM
```

5.5.20.4 Test If Device Is Loaded (.IFLOA/.IFNLOA) — The .IFLOA and .IFNLOA directives test whether a specific device handler is loaded (.IFLOA) or not loaded (.IFNLOA). If the test is true, IND processes the rest of the directive line.

The .IFLOA and .IFNLOA directives have the following syntax:

.IFLOA dev action

.IFNLOA dev action

where:

dev	represents the device handler you want to test. You can use a character string, or string symbol in single quotes (when substitution mode is enabled), to specify the device handler.
action	represents how processing continues depending on the test results

In the following example, if DY1 is loaded, control branches to RT67:

```
.IFLOA DY1 .GOTO RT67
```

5.5.20.5 Test If Symbol Is True or False (.IFT/.IFF) — The .IFT and .IFF directives test whether a logical symbol is true or false, or test whether specific bits in a numeric symbol are set to 1 or 0.

The .IFT and .IFF directives have the following syntax:

.IFT logsym action
.IFT [mask] numsym action

.IFF logsym action
.IFF [mask] numsym action

where:

logsym	represents a logical symbol you want to test for a true or false value
[mask]	represents a numeric symbol or expression, in the range 0–177777(octal) or 0–65535 (decimal), that determines which bits to test for a 1 or 0 value. You must include the surrounding brackets when you use this argument.
numsym	represents a numeric symbol whose bits you want to test for a 1 or 0 value
action	represents how processing continues depending on the test results

.IFT Directive

When you use the .IFT directive with a logical symbol, IND tests the symbol. If the symbol's value is true, IND processes the remainder of the command line. Otherwise, the symbol's value is false, and the next command line is processed instead.

When you use the .IFT directive with [mask], IND checks to see which bits in the mask are set to 1 (for true), and tests the corresponding bits in the numeric symbol. If any of these are also set to 1, IND processes the remainder of the command line. Otherwise, the next command line is processed instead.

When the following sample command lines are processed, IND branches to line 100.

```
.SET LOGA
.IFT LOGA .GOTO 100
```

IND also branches control to line 100 when the next two command lines are processed.

```
.SET [8] NUMB
.IFT [7] NUMB .GOTO 100
```

.IFF Directive

When you use the .IFF directive with a logical symbol, IND tests the symbol. If the symbol's value is false, IND processes the remainder of the command line. Otherwise the next command line is processed.

When you use the .IFF directive with [mask], IND checks to see which bits in the mask are set to 1, and tests the corresponding bits in the numeric symbol. If any of these are set to 0, IND processes the remainder of the command line. Otherwise, the next command line is processed instead.

When the following sample command lines are processed, IND branches control to line 100.

```
.SETF LOGA
.IFF LOGA .GOTO 100
```

IND also branches to line 100 when the next two command lines are processed.

```
.SETF [5] NUMB
.IFF [7] NUMB .GOTO 100
```

5.5.20.6 Compound Tests -- You can combine logical tests with the .AND and .OR directives.

IND also lets you perform more than one logical test on the same line. If you do put more than one logical test on the same line, IND assumes an .AND directive between the tests so you can omit it.

IND processes .AND directives before .OR directives. When IND processes the first line in the following example, the effect is the same as the second line:

```
.IFT A .OR .IFT B .AND .IFT C .GOTO D
.IFT A .OR (.IFT B .AND .IFT C) .GOTO D
```

In the following example, control will pass to HELP if logical symbol A is true and logical symbol B is false:

```
.IFT A .AND .IFF B  .GOTO HELP
```

In the following example, IND assumes an .AND directive between the two tests .IFT A and .IFF B:

```
.IFT A .IFF B  .GOTO HELP
```

In the following example, control will branch to the label HELP if A is true or if B is false:

```
.IFT A .OR .IFF B .GOTO HELP
```

5.5.21 Increment Numeric Symbol (.INC)

The .INC directive adds one to a numeric symbol.

The .INC directive has the following syntax:

.INC numsym

where:

numsym represents the numeric symbol being incremented

In the following example, IND increments numeric symbol UNITS by one:

```
.INC UNITS
```

If you use the .INC directive to increment a logical or string symbol, IND prints an error message and exits from processing.

5.5.22 Branch on Error (.ONERR)

The .ONERR directive causes IND to continue processing at another location in a control file when IND detects any of the errors listed in Table 5–8. Table 5–8 lists the actual error messages generated by the errors; refer to the *RT–11 System Message Manual* for more detail on the causes of these errors.

The .ONERR directive has the following syntax:

```
.ONERR label
```

where:

label — represents a label in a control file marking the location at which you want to continue processing. If you do not specify a label, the .ONERR directive is disabled.

Table 5–8: Errors Intercepted by .ONERR Directive

?IND-F-Bad range or default specification
?IND-F-Data file error
?IND-F-Data file open
?IND-F-Deletions special symbol
?IND-F-Error reading from terminal
?IND-F-File already open
?IND-F-File not open
?IND-F-File read error
?IND-F-Invalid file number
?IND-F-Invalid keyword
?IND-F-Invalid nesting
?IND-F-Label not at beginning of line
?IND-F-Null control string (to .PARSE directive)
?IND-F-Redefining symbol to different type (symbol)
?IND-F-RETURN without .GOSUB
?IND-F-String substitution error
?IND-F-Subroutine nesting too deep
?IND-F-Swap error
?IND-F-Symbol type error (symbol)
?IND-F-Undefined label (,label)
?IND-F-Undefined symbol (symbol)

Usually when IND detects any of these errors, IND prints an error message and stops executing the control file. When you use .ONERR, IND branches control to the label you specify and continues execution instead.

You must place the .ONERR directive before the location in the control file where IND detects an error. You may use .ONERR directives anywhere in your control file, but each time IND detects an error control passes to the label specified in the most recently processed .ONERR directive. A .ONERR directive is only effective within the begin-end block or the control file in which it is defined.

Once issued, an .ONERR directive remains in effect until it is redefined (when IND finds another .ONERR directive in the control file) or until the .ONERR directive is disabled. Whenever IND detects one of the errors listed in Table 5–8, the current .ONERR directive is processed and then disabled. You must then define another .ONERR directive to continue error processing.

5.5.23 Opening Data Files

IND has the following three directives for opening auxiliary data files:

```
.OPEN
.OPENA
.OPENR
```

The sections that follow show how to use these directives.

When you use these file opening directives, observe the following guidelines:

1. You can have up to four files open at any time. When you use any of these directives, specify a file number, from 0 to 3, with the file specification. The first file you open is number 0, the second is number 1, and so on.
2. Before exiting from IND or passing execution to the keyboard monitor, you must close any open files.

5.5.23.1 Open File (.OPEN) – The .OPEN directive opens a file for output. Use the .OPEN directive only when you wish to send data to a file. Before you open a file, make sure that the output file you specify is not protected.

If you use the .OPEN directive and specify a file that already exists, IND deletes the original file when you subsequently use the .CLOSE directive.

The .OPEN directive has the following syntax:

```
.OPEN [#n] filespec
```

where:

n	represents a file number. (The default is #0.) You can substitute a numeric symbol for n by enclosing the symbol in apostrophes.

filespec — represents the file to be opened. (The default file type is .DAT.)

In the following example, IND opens SECOUT.DAT for output:

```
.OPEN #0 SECOUT
```

5.5.23.2 Open File for Append (.OPENA) — The .OPENA directive opens a file and appends all subsequent data to the file. Use this directive only when you wish to send output to a file. If you use this directive with a file that does not already exist, this directive has the same effect as the .OPEN directive. Before you open a file, make sure it is not protected.

The .OPENA directive has the following syntax:

.OPENA [#n] filespec

where:

n — represents a file number. (The default is #0.) You can substitute a numeric symbol for n by enclosing the symbol in apostrophes.

filespec — represents the file to be opened. (The default file type is .DAT.)

In the following example, IND opens SECOUT.DAT and appends subsequent data to it:

```
.OPENA #0 SECOUT
```

5.5.23.3 Open File for Read (.OPENR) — The .OPENR directive opens a file so that you can read it. Use this directive only when you wish to read from a file.

The .OPENR directive has the following syntax:

.OPENR [#n] filespec

where:

n — represents a file number. (The default is #0.) You can substitute a numeric symbol for n by enclosing the symbol in apostrophes.

filespec — represents the file to be opened

5.5.24 Parse a String (.PARSE)

The .PARSE directive divides a character string into substrings. IND then assigns the substrings to string symbols that you specify.

The .PARSE directive has the following syntax:

```
.PARSE string "ctrl-string" sym1 sym2 ...
```

Where:

string	represents the character string you wish to parse. You can use a string symbol to represent the string you wish to parse, or you can specify a character string. If you specify a character string, make sure the string begins and ends with quotes (").
ctrl-string	represents the control string, which specifies the substring delimiters. (Make sure you include the quotes.) Do not separate substring delimiters.
sym1 sym2 ...	represent the substring symbols into which you wish to store the substrings. Separate string symbols with spaces.

IND parses the character string into substrings as specified by the control string. The following example illustrates a command line that uses the .PARSE directive:

```
.PARSE "DY1:LNKLIB.OBJ" ":." DEV FILE TYPE
```

In the sample line above, DY1:LNKLIB.OBJ is the character string to be parsed. The control string ":." contains a colon and a period as the substring delimiters. The colon serves as a terminator for the first substring, and the period serves as a terminator for the second substring. If the number of substring symbols exceeds the number of characters in the control string, the last character of the control string will be repeated as a substring delimiter. The last substring symbol receives the remaining part of the character string.

If there are more substring symbols than substrings, IND sets the additional substring symbols to null. The special symbol <STRLEN> contains the number of strings processed, including explicit null symbols.

In the sample command line above, .PARSE "DY1:LNKLIB.OBJ" ":." DEV FILE TYPE, DEV, FILE, and TYPE are the symbols into which the substrings are to be stored. In this example, the .PARSE directive stores all the characters up to a colon (that is, DY1) in the first symbol, DEV. All characters up to the period (that is, LNKLIB) are stored in the second symbol, FILE. The remaining characters (that is, OBJ) are stored in the last symbol, TYPE.

In the following example, the symbol to be parsed, MACFIL, contains the string COPY FILE1.MAC,FILE2.MAC,,FILE3.MAC:

```
.PARSE MACFIL "_,." CDM A1 A2 A3 A4 A5
```

When IND processes the command line above, it produces the following symbols:

Symbol	Contents
COM	COPY
A1	FILE1.MAC
A2	FILE2.MAC
A3	null
A4	FILE3.MAC
A5	null
<STRLEN>	6

5.5.25 Purge File (.PURGE)

The .PURGE directive discards or closes a specified output file and frees its file number, without taking any other action. If you use .PURGE with a file that was previously opened with .OPEN, IND discards that file. If you use .PURGE with a file previously opened with .OPENA, IND makes no changes to that file. If you use .PURGE after .OPENR, .PURGE has the same effect as .CLOSE.

The .PURGE directive has the following syntax:

.PURGE [#n]

where:

n represents a file number from 0 to 3. (The default is 0.)

5.5.26 Read a Record (.READ)

The .READ directive reads an ASCII record from a file previously opened with the .OPENR directive. The record is stored in the specified string symbol. An ASCII record is a string of characters delimited by line terminators.

The .READ directive has the following syntax:

.READ [#n] strsym

where:

n	represents a file number from 0 to 3. (The default is 0.)
strsym	represents the string symbol which is assigned the characters in the record

Since the string variable cannot exceed 132 characters, file records cannot exceed 132 characters, which includes the carriage return and line feed characters at the end of the record.

The <EOF> symbol is set only when the .READ directive encounters end-of-file. If end-of-file has occurred, both <EOF> and <FILERR> will be set to indicate end-of-file. When processing is complete, close the file with the .CLOSE or .PURGE directive. The .READ directive ignores null characters.

5.5.27 Return from a Subroutine (.RETURN)

The .RETURN directive appears at the end of a subroutine and returns control to the most recently saved position in the indirect control file.

The .RETURN directive has the following syntax:

```
.RETURN
```

5.5.28 Set Numeric Symbol to Decimal or Octal (.SETD/.SETO)

The .SETD and .SETO directives change the radix of a numeric symbol to decimal (.SETD) or octal (.SETO). These directives do not alter the value of a symbol, only its radix.

The .SETD and .SETO directives have the following syntax:

```
.SETD numsym
.SETO numsym
```

where:

numsym	represents the numeric symbol whose radix is being changed

In the following example, the value of the numeric symbol UNITS is set to 10 (decimal), then changed to 12 (octal).

```
.SETN UNITS 10.
.SETO UNITS
```

5.5.29 Set Symbol to Logical Value (.SETL)

The .SETL directive sets or clears the bits of a logical symbol depending on the value of a logical expression. If the symbol has not been defined, IND makes an entry in the symbol table and sets the symbol to the value (set or cleared) of the logical expression. If the symbol has already been defined, IND resets the symbol accordingly. If the logical symbol was previously defined as a numeric or string symbol, IND prints an error message and exits from processing.

The .SETL directive has the following syntax:

```
.SETL logsym logexp
```

where:

logsym	represents the logical symbol to be set or cleared
logexp	represents a logical expression that can include logical symbols and numeric values joined by the logical operators & (logical AND), ! (logical OR), and ^ (logical NOT). No imbedded blanks or tabs are permitted. IND evaluates from

left to right unless parentheses are used to form subexpressions, which are evaluated first. If any value in an expression is specified as decimal, IND assumes that all values in the expression are decimal; otherwise, all values are octal.

When you use the .SETL directive, the logical symbol you specify is set to the value represented by the logical expression.

In the following example, the control file contains these lines:

```
.SETL MONITR $INTSW!FBSW!XMSW
```

If any of the three logical symbols (SW, FB, or XM) is set to true, the logical symbol MONITR is set to true. If none of the three is set to true, MONITR is set to false.

5.5.30 Set Symbol to Numeric Value (.SETN)

The .SETN directive defines or changes the numeric value of a specified symbol. If the symbol has not been defined, IND makes an entry in the symbol table and sets the symbol to the numeric value specified. If the symbol has already been defined, IND resets the symbol accordingly. If the numeric symbol was previously defined as a logical or string symbol, IND prints an error message and exits from processing.

The .SETN directive has the following syntax:

```
.SETN numsym numexp
```

where:

numsym	represents a numeric symbol
numexp	represents a numeric expression

When specifying a numeric value to assign to a symbol, you may combine numeric symbols and constants to form a numeric expression. If numeric expressions are used, no imbedded blanks or tabs are permitted. IND evaluates from left to right unless parentheses are used to form subexpressions, which are evaluated first. If none of the values in the expression includes a decimal point, the radix of the symbol is determined by the default radix in effect (as determined by the .ENABLE's in the expression and octal mode is enabled, all values specified without decimal points are assumed to be octal, and are converted to decimal before the arithmetic operation is performed. All values specified with decimal points are treated and stored as decimal values.

In the following example, IND assigns the integer 27 (octal) to the numeric symbol NUMBER:

```
.SETN NUMBER 27
```

In the following example, IND assigns to the numeric symbol A1 the value of symbol A2 minus five, multiplied by three. All numbers are interpreted as octal.

```
.SETN A1 3*(A2-5)
```

5.5.31 Set Symbol to String Value (.SETS)

The .SETS directive defines or changes the value of a specified string symbol. If the symbol has not been defined, IND makes an entry in the symbol table and sets the symbol to the specified string value. If the symbol has been defined, IND resets the symbol accordingly. If the symbol has been defined previously as logical or numeric, IND prints an error message.

The .SETS directive has the following syntax:

```
.SETS strsym strexp
```

where:

strsym	represents a string symbol
strexp	represents any string expression (You can concatenate string symbols and substrings to form a valid string expression.)

IND assigns to the specified symbol the string value represented by the string expression strexp. If a string constant is used in strexp, the constant must be enclosed by quotes.

You can combine a string symbol, constant, or substring with another string symbol or substring by using the plus sign (+) to form a string expression.

In the following example, IND assigns to the symbol A the string value ABCDEF:

```
.SETS A "ABCDEF"
```

In the following example, IND assigns to the symbol X the value of symbol STR2 (which contains ZZZ) plus ABC, resulting in ZZZABC:

```
.SETS X STR2+"ABC"
```

In the next example, IND assigns the symbol X the string value of STR2 (which contains ZZZ) plus the last three characters of string A (which contains ABCDEF), resulting in ZZZDEF:

```
.SETS X STR2+A[4:6]
```

In the next example, IND assigns to the string symbol MYFILE the string value of the system device plus the string MYFILE.TXT. If the system device is RK:, string symbol MYFILE is assigned the string value RK:MYFILE.TXT.

```
.SETS MYFILE "SYS$SYS0+":MYFILE.TXT"
```

5.5.32 Set Symbol to True or False (.SETT/.SETF)

The .SETT and .SETF directives assign a value of true or false to a logical symbol. If the logical symbol has not been defined, IND makes an entry in the symbol table and sets the symbol to the value specified. If the symbol has already been defined, IND resets the symbol accordingly.

You can also use the .SETT and .SETF directives to redefine a numeric symbol by setting and clearing bits. You can use these directives with a numeric symbol only to reset bits in a previously defined symbol.

If the logical or numeric symbol you specify was previously defined as another type of symbol, IND prints an error message and processing halts.

The .SETT and .SETF directives have the following syntax:

```
.SETT logsym
.SETT [mask] numsym

.SETF logsym
.SETF [mask] numsym
```

where:

logsym	represents a logical symbol you want to define or redefine
[mask]	represents a numeric symbol or expression, in the range 0-177777(octal) or 0-65535 (decimal), that determines which bits are to be set or cleared in a numeric symbol. You must include the surrounding brackets when you use this argument.
numsym	represents a numeric symbol to be redefined by setting or clearing bits as dictated by the mask.

5.5.32.1 .SETT Directive — When you use the .SETT directive with a logical symbol, IND assigns the value of true.

When you use the .SETT directive with [mask], for every bit in the mask that is set to 1 the corresponding bit in the numeric symbol is also set to 1. For every bit in the mask that is set to 0, the corresponding bit in the symbol remains unchanged.

In the following example, IND sets the logical symbol X to true:

```
.SETT X
```

As a result, X can be used in the following sample command line:

```
.IFT X .GOTO 300
```

In the next example, IND sets bits 0, 1 and 2 of the previously defined numeric symbol NUM to 1. The rest of the bits remain as they were previously defined.

```
.SETT [7] NUM
```

As a result, when IND processes the following command line, control branches to label SUB1:

```
.IFT [7] NUM .GOTO SUB1
```

See Section 5.5.20.5 for more information on the .IFT and .IFF directives.

5.5.32.2 .SETF Directive -- When you use the .SETF directive with a logical symbol, IND assigns the value of false.

When you use the .SETF directive with [mask], for every bit in the mask that is set to 1 the corresponding bit in the numeric symbol is set to 0. For every bit in the mask that is set to 1, the corresponding bit in the symbol remains unchanged.

For example, in the following command line IND assigns a value of false to the logical symbol LOG.

```
.SETF LOG
```

In the next example, IND sets bits 1 and 3 of the numeric symbol NUM to 0. The rest of the bits remain as they were previously defined.

```
.SETF [12] NUM
```

As a result, when IND processes the following command line, control branches to the label SUB1.

```
.IFF [22] .GOTO SUB1
```

See Section 5.5.20.5 for more information on the .IFT and .IFF directives.

5.5.33 Terminate Processing (.STOP)

The .STOP directive halts control file processing. When you use the .STOP directive to halt processing, IND prints the following message on the console:

```
@ (EOF)
```

The .STOP directive has the following syntax:

```
.STOP
```

The .STOP directive has the same effect as the logical end-of-file directive (/).

5.5.34 Test a Symbol (.TEST)

You can use the .TEST directive to determine the symbol type of a symbol, to test the characters of a string symbol for their type (alphanumeric or RAD50), to find the starting position of an ASCII string within a character string, or to test a numeric symbol for its radix.

The .TEST directive has the following syntax:

```
.TEST symbol
.TEST strsym matchstring
```

where:

symbol	represents the symbol you want to test
strsym	represents the string symbol you want to test
matchstring	represents the ASCII string whose starting position within a character string you want to find. The variable matchstring can represent an ASCII string, a string symbol, or an expression.

When you use the .TEST directive to test for a symbol type, IND indicates the symbol's type by storing the proper numeric code in <SYMTYP>. If the symbol is found to be a string symbol, IND sets the special symbols <ALPHAN>, <RAD50>, and <STRLEN> accordingly. If the symbol is found to be a numeric symbol, IND reports the radix of the symbol by setting the special symbol <OCTAL> to true if the symbol's radix is octal, or setting <OCTAL> to false if the symbol's radix is decimal.

To search for an ASCII string within a character string, use the .TEST strsym matchstring directive. Then test the contents of <STRLEN>. If <STRLEN> is 0, the match string you specified was not found in the character string. A nonzero value represents the absolute position (in octal) of the match string in the character string, rather than an offset to the starting position. For example, the following command lines test the string symbol ADDRES for the position of the match string "STREET", and then check to see whether the string was found:

```
.SETS ADDR "STREET"
.TEST ADDR "STREET"                    !LOOK FOR "STREET"
                                       !IN STRING CONTAINED IN
                                       !STRING SYMBOL ADDR
.IF <STRLEN> = 0 .GOTO NOTFND          !STRING NOT FOUND
.IF <STRLEN> <> 0 .GOTO POS            !STARTING POS. OF STRING
```

In the example above if <STRLEN> has a value of 1, the string "STREET" begins on the first character in the string represented by ADDR. Note that the match string will be found within the target string only if both are in uppercase characters or both are in lowercase characters.

In the following example, IND enters the number of characters in the string symbol A into <STRLEN> and sets <ALPHAN> and <RAD50> accordingly.

```
.TEST A
```

The special numeric symbol <STRLEN> is then available to compare the length of string symbol A to a numeric constant or expression.

In the following example a control file contains these lines:

```
.ASKN [1:10.] UNITS # OF UNITS ERROR LOGGER SUPPORTS?
.ASKN [1:7] DEVSLT # EXTRA DEVICE SLOTS WANTED?
.IF DEVSLT > 1 .GOTO DEVSUB
.TEST UNITS
```

IND tests the numeric symbol UNITS for radix. In this case, since the range specification contained a decimal point after the number 10, the response would be interpreted as decimal. Therefore, when IND tests the symbol UNITS, the special symbol <OCTAL> will be false because the radix of the symbol is decimal.

5.5.35 Test for Installed Device (.TESTDEVICE)

The .TESTDEVICE directive obtains information on the specified device.

The .TESTDEVICE directive has the following syntax:

```
.TESTDEVICE device
```

where:

device	represents the device you want to test. The colon (:) following the device name is optional.

The results of the test are stored in the special symbol <EXSTRI>. You can use the .PARSE or .TEST directive to move the information in <EXSTRI> to separate string symbols for inspection.

If the device name you specify is invalid or the device is not installed, <EXSTRI> contains the characters NSD (no such device). If the device is valid, eight fields of information are returned in <EXSTRI>:

Field	Contents
1	Physical device name; if no unit number is specified, unit 0 is assumed.
2	Device size, displayed as a decimal number (with a decimal point). If the device has a variable-sized handler and no volume is mounted in the drive, the size returned is the smallest for that device.
3, 4, 5	Three fields, each containing 0.
6	LOD (device handler loaded) or UNL (device handler not loaded).
7	ONL (on-line), OFL (off-line), or UNK (unknown). IND checks for this attribute by attempting to read from the device. If a volume is mounted in the device and IND is able to read it, the ONL attribute is returned in field 7. If a read error occurs, such as when no volume is mounted, the OFL attribute is returned. Also, if the device tested is a logical disk, and the file assigned to the logical disk is for any reason not accessible, the OFL attribute is returned. Devices that are write-only, and sequential-access devices that are not non-RT-11 directory-structured (such as TT: and CR:), return the attribute UNK.
8	MTD (mounted) or NMT (not mounted). The MTD attribute is returned for logical disk devices that have been assigned with the keyboard monitor MOUNT command. Any device not assigned with the MOUNT command, or any device that is not a logical disk, returns NMT in field 8.

In <EXSTRI>, each field of information is followed by a comma, including field 8.

In the following example, LD2: (logical disk unit 2) is associated with the file DL1:MASTER.DSK, and also assigned the logical device name SEC. The example shows the contents of <EXSTRI> when the .TESTDEVICE directive is used with SEC.

```
MOUNT LD2: DL1:MASTER SEC
.TESTDEVICE SEC
!'<EXSTRI>'
!LD2:,5000.,0,0,0,UNL,ONL,NTE,
```

The contents of <EXSTRI> show that the physical device name is LD2, whose size is 5000 (decimal) blocks. The LD handler is not loaded, the device on which LD2: resides (DL1:) is on-line, and LD2: has been assigned with the MOUNT command.

In the next example, device DY0: is tested.

```
.TESTDEVICE DY:
!'<EXSTRI>'
!DY0:,494.,0,0,0,LOD,OFL,NMT
```

The contents of <EXSTRI> show that the physical device name is DY0, and the device's size is 494 (decimal) blocks. The DY handler is loaded, but the there is no volume in DY0: so the device is off-line. (Since DY has a variable-size handler, the smallest device size is displayed.) DY has not been assigned with the MOUNT command.

5.5.36 Test for File (.TESTFILE)

The .TESTFILE directive checks to see whether a specified file exists. The directive then places the name of the specified file in the special symbol <FILSPC> and the results of the test in <FILERR>.

The .TESTFILE directive has the following syntax:

```
.TESTFILE  filespec
```

where:

filespec — represents the file you wish to verify

The default device specification is DK:, and the default file type is .DAT.

5.5.37 Obtain Volume Identification (.VOL)

The .VOL directive assigns the volume identification of a volume to a string symbol.

The syntax of the .VOL directive is:

```
.VOL  strsym  dev
```

where:

strsym — represents the string symbol in which to store the volume identification

dev — represents the device that contains the volume from which to read the volume identification. You can use either a string symbol in single quotes (when substitution mode is enabled), or a character string, to specify the device. The colon following the device mnemonic is ignored.

In order for IND to read the volume identification, the volume must be an RT-11 file-structured device.

IND reads only the volume identification from the specified volume and stores it in the specified symbol. The owner name is not stored. The length of the volume identification is predetermined (12 spaces); if the volume identification is less than 12 characters, the value of the symbol is padded with blank characters up to 12 spaces.

Part III
Text Editing

You use an editor to create and modify textual material. Part III describes the RT-11 text editor, EDIT, and explains how to use it.

Chapter 6
Text Editor (EDIT)

The text editor (EDIT) is a program that creates or modifies ASCII source files for use as input to other system programs such as the MACRO assembler or the FORTRAN compiler. EDIT, which accepts commands you type at the terminal, reads ASCII files from any input device, makes specific changes, and writes on any output device. EDIT allows efficient use of VT11 or VS60 display hardware, if they are part of the system configuration.

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

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

Note that you cannot perform any edit operations on a protected file. To disable a file's protected status, see the RENAME and UNPROTECT command descriptions in Section 4.6.

6.1 Calling EDIT

You can call the text editor when you are at monitor level. The syntax of the command is:

See Section 4.6 for a description of the EDIT command and its options.

6.2 Modes of Operation

The editor operates in either command mode or text mode. In command mode the editor interprets all input you type on the keyboard as commands to perform some operation. In text mode the editor interprets all typed input

as text to replace, insert into, or append to the contents of the text buffer. You can use a special editing mode, called immediate mode, with the VT-11 display hardware. Section 6.7.2 describes this mode.

Immediately after being loaded into memory and started, the editor is in command mode. EDIT prints an asterisk at the left margin of the console terminal page to indicate that it is ready to accept a command. Terminate all commands by pressing the ESCAPE key twice in succession. Execution of commands proceeds from left to right. When EDIT encounters an error before beginning execution of a command string, it prints an error message followed by an asterisk at the beginning of a new line, indicating that it is still in command mode, that it is waiting for a command, and that execution of the illegal command has not occurred. You should retype the command correctly.

To enter text mode, type a command that must be followed by a text string. These commands insert, replace, exchange, or otherwise manipulate text. When you type one of these commands, EDIT recognizes all succeeding characters as part of the text string until it encounters an ESCAPE character. The ESCAPE terminates the text string and causes the editor to reenter command mode.

6.3 Special Key Commands

Table 6-1 lists the EDIT key commands. Type a control command by holding down the CTRL key and typing the appropriate character. Control commands are documented as CTRL/C, CTRL/R, etc. Throughout this chapter, the $ character is used in examples to represent the ESCAPE key. ^C is used in examples to represent CTRL/C, and ^X is used in examples to represent CTRL/X.

Table 6-1: EDIT Key Commands

Key	Explanation
ESCAPE, ALTMODE, or SEL	ESC key, echoes as $. A single ESCAPE terminates a text string. A double ESCAPE (two consecutive ESCAPEs) executes the command string. For example: *G(MOV A,B)$$ The first ESCAPE ($) terminates the text object (MOV A,B) of the Get command. The double ESCAPE terminates the Delete command and causes execution of the entire statement with the result that the character S will be deleted.
CTRL/C	Echoes as ^C. If EDIT encounters a CTRL/C as a command in command mode, it terminates execution and returns control to the monitor. You can restart the editor by typing R EDIT or REENTER in response to the monitor's prompt. If EDIT encounters a CTRL/C in a text object, the CTRL/C is included in the text object, as if it were just like any other character. If the editor is executing a lengthy command and you want to stop EDIT, type two CTRL/C commands in succession. This will abort the command, generate

Table 6–1: EDIT Key Commands (Cont.)

Key	Explanation
	the *(EDIT) Command aborted* error message, and return the editor to command mode. For example:

```
*I^C^C^C$$
      *^C$$
```

In the first command, the three CTRL/C characters are part of the text object of the Insert command. EDIT treats them like any other character. In the second command string, the CTRL/C occurs at command level, which causes the editor to terminate.

If no commands (other than CLOSE) are executed between the time you terminate the editor and the time you issue a REENTER command, the text buffer is preserved as it was at program termination. However, only the text buffer is preserved. The input and output files are closed, and the save and undo buffers are reinitialized.

If you inadvertently terminate an editing session before the output file can be closed, you can often use the monitor CLOSE command to make permanent the portion of the output file that has already been written (see Section 4.5). You can then reenter the editor, open a new output file, and continue the editing session.

CTRL/O — Echoes as ^O and a carriage return. Inhibits printing on the terminal until completion of the current command string. Typing a second CTRL/O resumes output.

CTRL/U — Echoes as ^U and a carriage return. Deletes all characters on the current terminal input line. (Typing CTRL/U has the same effect as pressing the RUBOUT key until all the characters back to the beginning of the line are deleted.)

DELETE or RUBOUT — Deletes a character from the current command line; echoes as a backslash (\) followed by the character deleted. Each succeeding DELETE you type deletes and echoes another character. An ensuing backslash prints when you type a key other than DELETE. The erasure is done from right to left. Since EDIT accepts multiple-line commands, DELETE can delete past the carriage return and line feed combination and delete characters on the previous line. You can use DELETE in both command and text modes.

TAB — Spaces to the next hardware tab stop. Tab stops are positioned every eight spaces on the terminal; pressing the TAB key causes the carriage to advance to the next tab position.

CTRL/X — Echoes as ^X and a carriage return. CTRL/X causes the editor to ignore the entire command string you are currently entering. The editor prints a carriage return and line feed combination and an asterisk to indicate that you can enter another command. For example:

```
*I ABCD
EFGH^X
*
```

A CTRL/U would cause only deletion of EFGH. CTRL/X erases the entire command.

If you are running a system job, you must SET TTNOFD to enable this function of CTRL/X. If you do not and you type CTRL/X, the system intercepts the CTRL/X and prompts you for a system job name.

6.4 Command Structure

EDIT commands fall into eight general categories. Table 6–2 lists these categories, the commands they include, and the sections of this chapter where you will find information on the commands.

Table 6–2: EDIT Command Categories

Category	Commands	Section
File open and close		
	Edit Backup (EB)	6.6.1.2
	Edit Read (ER)	6.6.1.1
	Edit Write (EW)	6.6.1.2
	End File (EF)	6.6.1.4
File input/output		
	Exit (EX)	6.6.2.4
	Next (nN)	6.6.2.3
	Read (R)	6.6.2.1
	Write (nW)	6.6.2.2
Intensifier mode		
	CTRL/D	6.7.2
	CTRL/G	6.7.2
	CTRL/N	6.7.2
	CTRL/V	6.7.2
	DELETE or RUBOUT	6.7.2
	ESCAPE or ALTMODE	6.7.2
Pointer location		
	Advance (nA)	6.6.3.3
	Beginning (B)	6.6.3.1
	Jump (nJ)	6.6.3.2
Search		
	Find (nF)	6.6.4.2
	Get (nG)	6.6.4.1
	Position (nP)	6.6.4.3
Text listing		
	List (nL)	6.6.5.1
	Verify (V)	6.6.5.2
Text modification		
	Change (nC)	6.6.6.4
	Delete (nD)	6.6.6.2
	Exchange (nX)	6.6.6.5
	Insert (I)	6.6.6.1
	Kill (nK)	6.6.6.3
Utility		
	Edit Console (EC)	6.7.1
	Edit Display (ED)	6.7.1
	Edit Lower (EL)	6.6.7.6
	Edit Upper (EU)	6.6.7.6
	Edit Version (EV)	6.6.7.5
	Execute Macro (nEM)	6.6.7.4
	Macro (M)	6.6.7.3
	Save (nS)	6.6.7.1
	Unsave (U)	6.6.7.2

The general syntax for all the EDIT commands, except immediate mode commands, is:

[n]C[text]$$

or

[n]C$$

where:

n	represents one of the arguments from Table 6–3
C	represents a 1- or 2-letter command
text	represents a string of ASCII characters

As a rule, commands are separated from one another by a single ESCAPE; however, if the command requires no text, the separating ESCAPE is not necessary. Commands are terminated by a single ESCAPE; typing a second ESCAPE begins execution. (You use ESCAPE differently when immediate mode is in effect; see Section 6.7.2.)

The syntax of display editor commands is different from the normal editing command format, and is described in Section 6.7.

6.4.1 Arguments

An argument is typed before a command letter. It specifies either the particular portion of text to be affected by the command or the number of times to perform the command. With some commands, this specification is implicit and no argument is needed; other editing commands require an argument. Table 6–2 lists the possible arguments and their meanings.

Table 6–3: Command Arguments

Argument	Meaning
n	Represents an integer in the range −16383 to +16383 (decimal) and may, except where noted, be preceded by a plus (+) or minus (−) sign. If no sign precedes n, it is assumed to be a positive number. The absence of n implies a 1 (or −1 if a minus sign precedes a command). This argument can represent the number of characters or lines forward (+) or backward (−) to move the pointer, or it can represent the number of times to execute the operation.
0	Indicates the text between the beginning of the current line and the location pointer (see Section 6.4.3).
:	Refers to the text between the location pointer and the end of the text in the buffer.
=	Represents −n, where n is equal to the length of the last text argument used. Use this argument with the Jump, Delete, and Change commands only.

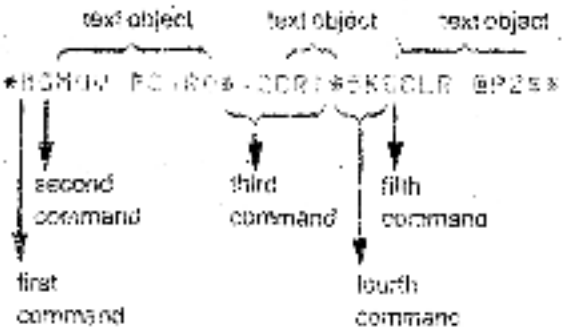
The roles of all arguments are explained in the following sections.

6.4.2 Command Strings

All EDIT command strings are terminated by two successive ESCAPE characters. Use spaces, carriage returns, and line feeds within a command string to increase command readability. EDIT ignores them unless they appear in a text string.

Commands to insert text can contain text strings that are several lines long. Each line you enter is terminated by the carriage return, which inserts both a carriage return and a line feed character into the text. The entire command is terminated by a double ESCAPE.

You can string several commands together and execute them in sequence. For example:



where:

B	is the first command
GMOV PC,R0	is the second command (MOV PC,R0 is the text object)
-2CR1	is the third command (R1 is the text object)
5K	is the fourth command
GCLR @R2	is the fifth command (CLR @R2 is the text object)
$	represents the ESCAPE key; separates the end of each text object from the following command
$$	represents the ESCAPE key pressed twice; executes the commands

Execution of a command string begins when you type the double ESCAPE and proceeds from left to right. EDIT ignores spaces, carriage returns, line feeds, and single ESCAPEs, except when they are part of a text string. Thus, these two examples have the same result:

```
*B$GMOV PC,R0$-2CR1$5KGCLR @R2$$

*B$ GMOV PC,R0$
-2CR1$
5K GCLR @R2$$
```

6.4.3 Current Location Pointer

Most EDIT commands function with respect to a movable location pointer that is normally located between the most recent character operated on and the next character in the buffer. It is important to think of this pointer as being between two characters, and never directly on a character.

At the start of editing operations, the pointer precedes the first character in the buffer, although it is not displayed on the console terminal. At any time during the editing procedure, think of the pointer as representing the current position of the editor in the text. The pointer moves during editing operations according to the type of editing operation being performed. Refer to text in the buffer as so many characters or lines preceding or following the pointer.

6.4.4 Character- and Line-Oriented Command Properties

EDIT commands are either character-oriented or line-oriented: character-oriented commands affect a specified number of characters preceding or following the pointer; line-oriented commands operate on entire lines of text.

6.4.4.1 Character-Oriented Commands — The argument of character-oriented commands specifies the number of characters in the buffer on which to operate. If *n* is unsigned (positive), the command moves the pointer in a forward direction. If *n* is preceded by a minus sign (negative), the command moves the reference pointer backward. Line feeds, carriage returns, and null characters, although not printed, are embedded in text lines, counted as characters in character-oriented commands, and treated as any other text characters.

When you press the RETURN key, both a carriage return and a line feed character are inserted into the text. For example, assume the pointer is positioned as indicated in the following text (↑ represents the current position of the pointer):

```
MOV     #VECT,R2
CLR     @R2↑
```

The EDIT command –2J moves the pointer back two characters to precede the carriage return character.

```
MOV     #VECT,R2
CLR     @R2↑
```

The command 10J advances the pointer forward ten characters and places it between the ⟨LF⟩ and ⟨CR⟩ characters at the end of the second line. Note that the tab character preceding @R2 is also counted as a single character.

```
MOV     #VECT,R2
CLR     @R2↑
```

Finally, to place the pointer after the C in the first line, use a –1J command. The J (Jump) command is explained in Section 6.6.3.2.

```
MOV     #VECT,R2
CLR     @R2(R0)
```

6.4.4.2 Line-Oriented Commands – When you use line-oriented commands, the command argument specifies the number of lines on which to operate. Because EDIT counts the line-terminating characters to determine the number of lines on which to operate, an argument n does not affect the same number of lines forward (positive) as it affects backward (negative).

For example, the argument –1 applies to the line beginning with the first character following the second previous end-of-line and ending with the character preceding the pointer. The argument 1 in a line-oriented command, however, applies to the text beginning with the first character following the pointer and ending at the first end-of-line. Thus, if the pointer is at the center of the line, the argument –1 affects one and one-half lines backward from the pointer and the argument 1 affects one-half line beyond the pointer.

For example, assume the buffer contains:

```
MOV     PC,R1(R0)
ADD     #DRIV-.,R1(R0)
MOV     #VECT,R2(R0)
CLR     @R2(R0)
```

The command to advance the pointer one line (1A) causes the following change:

```
MOV     PC,R1(R0)
ADD     #DRIV-.,R1(R0)
MOV     #VECT,R2(R0)
CLR     @R2(R0)
```

The command 2A moves the pointer over two CR/LF combinations to precede the fourth line:

```
MOV PC,R1(R0)
ADD     #DRIV-.,R1(R0)
MOV     #VECT,R2(R0)
CLR     @R2(R0)
```

For another example, assume the buffer contains:

```
MOV     PC,R1(R0)
ADD     #DRIV-.,R1(R0)
MOV     #VECT,R2(R0)
CLR     @R2(R0)
```

A command of –1A moves the pointer back by one and one-half lines to precede the second line.

```
MOV     PC,R1[illegible]
*ADD    #DRIV-.,R1[illegible]
MOV     #VECT,R2[illegible]
CLR     BR2[illegible]
```

Now a command of –1A moves the pointer back by only one line.

```
*MOV    PC,R1[illegible]
ADD     #DRIV-.,R1[illegible]
MOV     #VECT,R2[illegible]
CLR     BR2[illegible]
```

6.4.5 Command Repetition

You can execute portions of a command string more than once by enclosing the portions in angle brackets (<,>) and preceding the left angle bracket with the number of iterations you desire. The syntax is:

n<command>

For example:

C1$C2$n<C3$C4$>C5$

where:

C	represents a command
n	represents an iteration argument

Commands C1 and C2 each execute once, then commands C3 and C4 execute *n* times. Finally, command C5 executes once and the command line is finished. The iteration argument (*n*) must be a positive number (in the range 1 through 16383 decimal); if unspecified, it is assumed to be 1. If the number is negative or too large, an error message prints. You can nest iteration brackets up to 20 levels. Before execution, EDIT checks command lines to make certain the brackets are correctly used and match.

Enclosing a portion of a command string in iteration brackets and preceding it with an iteration argument (*n*) has the same result as typing that portion of the string n times. Thus, these two examples are equivalent:

```
*3GARP$3C-0(6$-J)V$$
*3GARP$-0(6$-J-0(6$-J-0(6$-J-0V$$
```

Similarly, the following two strings are equivalent:

```
*2<A0>0$$
*A0A0A0A0$$
```

The following bracket structures are examples of legal usage:

```
<<><<><>>>>
<<<>><><>
```

The following bracket structures are examples of combinations that will cause an error message:

```
><><
<<<>>
```

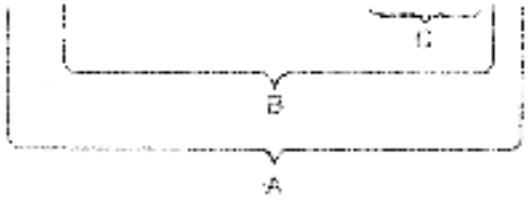
During command repetition, execution proceeds from left to right until a right bracket is encountered. EDIT then returns to the last left bracket encountered, decreases the iteration counter, and executes the commands within the brackets. When the counter is decreased to 0, EDIT looks for the next iteration count to the left and repeats the same procedures. The overall effect is that EDIT works its way to the innermost brackets and then works its way back again.

The most common use for iteration brackets is found in commands, such as Unsave (U), that do not accept repeat counts. For example:

```
#0<U>**
```

Assume you want to read a file called SAMP (stored on device DK:), and you want to change the first four occurrences of the instruction MOV #200,R0 on each of the first five pages to MOV #244,R4. Enter the following command line:

```
*ESAMP$*S4<N0<BCMOV #200,R0$=.$2<G0$=4,,,>>E$$
```



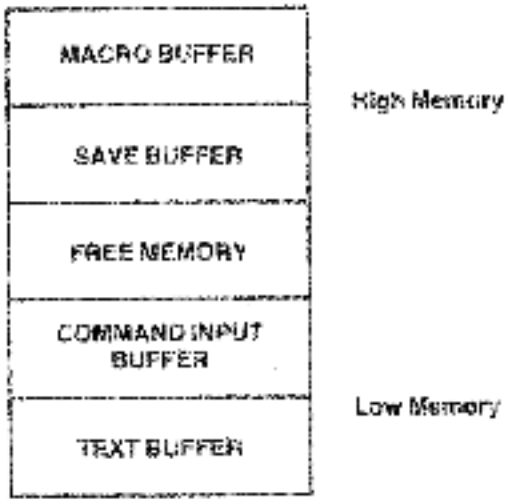
The command line contains three sets of iteration loops (A, B, C) and executes as follows:

Execution initially proceeds from left to right; EDIT opens the file SAMP for input and reads the first page into memory. EDIT moves the pointer to the beginning of the buffer and initiates a search for the character string MOV #200,R0. When it finds the string, EDIT positions the pointer at the end of the string, but the -4J command moves the pointer back, so that it is positioned immediately preceding the string. At this point, execution has passed through each of the first two sets of iteration loops (A, B) once. The innermost loop (C) is next executed three times, changing the 0s to 4s. Control now moves back to pick up the second iteration of loop B, and again moves from left to right. When loop C has executed three times, control again

moves back to loop B. When loop B has executed a total of four times, control moves back to the second iteration of loop A, and so forth, until all iterations have been satisfied.

6.5 Memory Usage

The memory area used by the editor is divided into four logical buffers as follows:



The text buffer contains the current page of text you are editing, and the command input buffer holds the command you are currently typing at the terminal. If a command you are currently entering is within ten characters of exceeding the space available in the command buffer, the following message prints on the terminal.

```
?EDIT-W-Command buffer almost full
```

If you can complete the command within ten characters, you can finish entering the command; otherwise you should press the ESCAPE key twice to execute that portion of the command line already completed. The warning message prints each time you enter a character in one of the last ten spaces.

If you attempt to enter more than ten characters, EDIT prints the following message and aborts the command.

```
?EDIT-F-Command buffer full: no command(s) executed
```

The save buffer contains text stored with the Save (S) command, and the macro buffer contains the command string macro entered with the Macro (M) command. (Both commands are explained in Section 6.6.7.)

EDIT does not allocate space for the macro and save buffers until an M or S command executes. Once you enter an M or S command, a subsequent 0M or 0U (Unsave) command executes to return that space to the free area.

The size of each buffer automatically expands and contracts to accommodate the text you are entering; if there is not enough space available to accommodate required expansion of any of the buffers, EDIT prints the error message:

```
TECO-F-Insufficient memory
```

6.6 Editing Commands

This section describes the commands and procedures required to:

- Read text from the input files to the buffer
- Create a backup version of the file
- List the contents of the buffer on the terminal
- Move the reference pointer
- Locate characters or strings of characters within the text buffer
- Insert, relocate, or delete text in the buffer
- Close the output file
- Terminate the editing session

The following sections are arranged, in order, by category of command function, as illustrated in Table 6–2.

6.6.1 File Open and Close Commands

You can use file open and close commands to:

- Open an existing file for input and prepare it for editing (ER)
- Open a file for output of newly created or edited text (EW)
- Open an existing file for editing and create a backup version of it (EB)
- Close an open output file (EF)

6.6.1.1 Edit Read – The Edit Read (ER) command opens an existing file for input and prepares it for editing. Only one file can be open for input at a time.

The syntax of the command is:

```
ERdev:filnam.typ
```

The argument dev:filnam.typ is limited to 19 characters and specifies the file to be opened. If you do not specify a device, DK: is assumed. If a file is currently open for input, EDIT closes that file and opens the new one.

Edit Read does not input a page of text nor does it affect the contents of the other user buffers.

With Edit Read you can close a file that is already open for input and reposition EDIT at the beginning of the file. The first Read command following any Edit Read command inputs the first page of the file.

This command string opens the file SAMP.MAC on device DT1:.

```
*ERDT1:SAMP.MAC$
```

NOTE

If you enter EDIT with the monitor EDIT/INSPECT or EDIT/OUTPUT command, an Edit Read command is automatically performed on the file named in the EDIT command.

6.6.1.2 Edit Write — The Edit Write (EW) command opens a file for output of newly created or edited text. However, no text is output and the contents of the buffers are not affected. Only one file can be open for output at a time. EDIT closes any output files currently open and preserves any edits made to the file.

The syntax of the command is:

EWdev:filnam.typ[n]

The argument dev:filnam.typ[n] is limited to 19 characters and is the name you assign to the output file being opened. If you do not specify a device, DK: is assumed. The optional argument [n] is a decimal number that represents the length of the file to be opened. Note that the square brackets ([]) are part of this argument and must be typed. If you do not specify [n], the system will default to either the larger of one-half the largest available space, or the second largest available space. If this is not adequate for the output file size, you must close this file and open another when this one becomes full. You should use the [0] construction whenever there is doubt as to whether enough space is available on the device for one output file.

If a file with the same name already exists on the device, EDIT deletes the existing file when you type an Exit, End File, or another Edit Write command. EDIT then prints the warning message:

```
?EDIT-W-Superseding existing file
```

The following command, for example, opens FILE.BAS on device DK: and allocates 11 blocks of space for it.

```
*EWFILE.BAS[11]$
```

NOTE

If you enter EDIT with the monitor EDIT/CREATE command, an Edit Write command is automatically performed on the file named in the EDIT command. If you enter EDIT with the monitor EDIT/OUTPUT command, an Edit Write is automatically performed on the file named with the /OUTPUT option.

6.6.1.3 Edit Backup -- The Edit Backup (EB) command opens an existing file for editing and at the same time creates a backup version of the file. EDIT closes any input and output files currently open. No text is read or written with this command.

The syntax of the command is:

EBdev:filnam.typ[n]

The argument dev:filnam.typ[n] is limited to 15 characters. If you do not specify a device, DK: is assumed. The argument [n] is optional and represents the length of the file to be opened; if you do not specify [n], the system defaults to the larger of either one-half the largest available space or the second largest available space.

The file you indicate in the command line must already exist on the device you designate, because text will be read from this file as input. At the same time, EDIT opens an output file under the same file name and file type. When the output file is closed, EDIT renames the original file (used as input) with the current file name and a .BAK file type, and deletes any previous file with this file name and a .BAK file type. EDIT closes the new output file and assigns it the name you specify in the EB command. This renaming of files takes place when an Exit, End File, or subsequent Edit Write or Edit Backup command executes. If you terminate the editing session with a CTRL/C command before the output file is closed, the new output file is not made permanent, and the renaming of the current version to .BAK does not take place. Thus:

```
*^C
```

This command opens BAS1.MAC on device SY:. When editing is complete, the old BAS1.MAC becomes BAS1.BAK and the new file becomes BAS1.MAC. EDIT deletes any previous version of BAS1.BAK.

NOTE

In EB, ER, and EW commands, leading spaces between the command and the file name are not permitted because EDIT assumes the file name to be a text string. All dev:filnam.typ specifications for EB, ER, and EW commands conform to RT-11 conventions for file naming. File names entered in command strings used with other system programs have identical specifications.

If you enter EDIT with the monitor EDIT command, an Edit Backup command is automatically performed on the file named in the EDIT command.

6.6.1.4 End File -- The End File (EF) command closes the current output file and makes it permanent. You can use the EF command to create an output file from a section of a large input file, or to close an output file that is full before you open another file. Modifiers are illegal with an EF command. Note that an implied EF command is included in EW and EB commands.

The syntax of the command is:

EF

Table 6–4 illustrates the relationship between the file open and close commands and the buffers and files themselves.

Table 6–4: EDIT Commands and File Status

Command	File Status		
	Input	Text Buffer	Output
ERXXX	Opens XXX for input; closes existing input file, if any	Unchanged	Unchanged
EWXXX	Unchanged	Unchanged	Opens XXX for output; closes existing output file, if any; performs .BAK renaming if EB is in effect
EBXXX	Opens XXX for input; closes existing input file, if any	Unchanged	Opens temporary file for output; closes existing output file, if any; performs .BAK renaming if EB is in effect
EF	Unchanged	Unchanged	Closes output file; performs .BAK renaming if EB is in effect
EA	Copies to output file	Copies to output file	Closes output file after copying completed; performs .BAK renaming if EB is in effect

6.6.2 File Input/Output Commands

You use file input/output commands to:

- Read text from an input file into the buffer
- Copy lines of text from the buffer into an output file
- Terminate the editing session

6.6.2.1 Read – Before you can edit text, you must read the input file into the buffer. The Read (R) command reads a page of text from the input file (previously specified in an ER or EB command) and appends it to the current contents of the text buffer (contents can be empty).

The syntax of the command is:

R

No arguments are used with the R command. If the buffer contains text when the R command is executed, the pointer does not move; however, if the buffer does not contain text, the pointer is placed at the beginning of the

buffer. EDIT transfers text to the buffer until one of the following conditions occurs:

- A form feed character, signifying the end of the page, is encountered.
- The text buffer is 500 characters from being full. (When this condition occurs, the Read command inputs up to the next carriage return and line feed combination, then returns to command mode. An asterisk prints as though the read were complete, but text will not have been fully input.)
- An end-of-file is encountered. (The *[EDIT]F-End of input file* message prints when all text in the file has been read into memory and no more input is available.)

The maximum number of characters you can bring into memory with an R command depends on the system configuration and the memory requirements of other system components. EDIT prints an error message if the read exceeds the memory available or if no input is available.

The following example creates a file using the EB and R commands.

```
*ESJK1.BAS$E
```

This command opens SJK1.BAS on DK; and permits modification.

```
*R/L$$
THIS IS PAGE ONE OF
FILE SJK1.BAS;
```

This command reads the first page of SJK1.BAS into the buffer. The pointer is placed at the beginning of the buffer. The /L command lists the contents of the buffer on the terminal, beginning at the pointer and ending with the last character in the buffer.

6.6.2.2 Write -- The Write (nW) command copies lines of text from the text buffer to the output file (as specified in the EW or EB command). The contents of the buffer are not altered and the pointer is left unchanged (unless an output error occurs).

NOTE

EDIT uses a system of intermediate buffers to store output before it writes the data to an output file. The Write command logically writes to the file, but output to a device does not occur until the intermediate buffer fills. When the editor closes a file (that is, after you issue an EF, EB, EX, or EW command), text is written from the buffer to the file and the file is complete. If the editor does not close a file (if you exit with CTRL/C and use the CLOSE command), it is possible that the output file will be missing the last 512 characters.

The syntax of the command is:

nW

The argument you supply with the W command determines the lines of text to copy. Table 6–5 lists the arguments for the W command.

Table 6–5: Write Command Arguments

Argument	Function
n	Writes n lines of text, beginning at the pointer and ending with the nth end-of-line character, to the output file.
-n	Writes n lines of text to the output file beginning with the first character on the -nth line and terminating at the pointer.
0	Writes to the output file the current line up to the pointer.
/.	Writes to the output file the text between the pointer and the end of the buffer.

If the buffer is empty when the write executes, no characters are output.

The following examples illustrate the use of the W command.

```
*5W$$
```

This command writes the five lines of text following the pointer into the current output file.

```
*-2W$$
```

This command writes the two lines of text preceding the pointer into the current output file.

```
*0/W$$
```

This command writes the entire text buffer to the current output file.

NOTE

If an output file fills while a Write command is executing, EDIT prints the *?EDIT-F-Output file full* message. In this case, EDIT positions the reference pointer after the last character it wrote successfully. You can then use the following recovery procedure:

1. Close the current output file (EF command).
2. Open a new output file (EW command).

3. Delete the characters just written by using -nD or -nK, where n is any arbitrary number that exceeds the number of lines or characters in the buffer.

4. Resume output.

6.6.2.3 Next—The Next (nN) command writes the contents of the text buffer to the output file, deletes the text from the buffer, and reads the next page of the input file into the buffer. The pointer is positioned at the beginning of the buffer.

The syntax of the command is:

nN

If you specify the argument n with the Next command, the sequence is executed n times. The N command operates in a forward direction only; therefore, you cannot specify negative arguments with an N command.

If EDIT encounters the end of the input file when trying to execute an N command, it prints *?EDIT-F-End of input file* to indicate that no further text remains in the input file. Since the contents of the buffer have already been transferred to the output file, the buffer is empty.

Using the N command is a quick way to write edited text to the output file and set up the next page of text in the buffer. The N command functions as though it were a combination of the Write, Delete, Read, and Beginning commands. (Delete is a text modification command, described in Section 6.6.6.2; the Beginning command is a pointer relocation command, described in Section 6.6.3.1.) Using the N command with an argument is a convenient way to set up text in the buffer, if you already know its page location.

In the following example, an N command copies an input file with more than one page of text to the output file.

```
*EFOO=TEST.MAC/E
```

This command opens the file TEST.MAC on device DK: and creates a new file entitled TEST.MAC for output.

```
*N/L/E
THIS IS PAGE ONE OF
FILE TEST.MAC>
```

This command reads the first page of the input file, TEST.MAC, into the buffer and lists the entire page on the terminal.

```
*N/L/E
?EDIT-F-End of input file
*
```

This command transfers the contents of the buffer to the output file, clears the buffer, and encounters the end of the file. Because it cannot complete the

N sequence, EDIT prints *[EDIT-F-End of input file* on the terminal. The buffer is empty and the entire input file has been written to the output file.

6.6.2.4 Exit -- Type the Exit (EX) command to terminate an editing session. The Exit command:

- Writes the text buffer to the output file
- Transfers the remainder of the input file to the output file
- Closes all open files
- Renames the backup file with a .BAK file type if an EB command is in effect
- Returns control to the monitor

The syntax of the command is:

```
EX
```

No arguments are accepted. Essentially, Exit copies the remainder of the input file into the output file and returns to the monitor. Exit is legal only when there is an output file open. If an output file is not open and you want to terminate the editing session, return to the monitor with CTRL/C.

NOTE

> You must issue an EF or EX command in order to make an output file permanent. If you use CTRL/C to return to the monitor without issuing an EF command, the current output file will not be saved. (You can, however, make permanent that portion of the text file that has already been written out, by using the monitor CLOSE command.)

An example of the contrasting uses of the EF and EX commands follows. Assume an input file, SAMPLE, contains several pages of text. The first and second pages of the file will be made into separate files called SAM1 and SAM2, respectively; the remaining pages of text will then make up the file SAMPLE. This can be done by using these commands:

```
*EWSAM1$$
*EFSAMPLE$$
*[NEF$$
*EWSAM2$$
*NEF$$
*EXSAMPLE$EX$$
```

Note that the EF commands are not necessary in this example, since the EW command closes a currently open output file before opening another.

6.6.3 Pointer Relocation Commands

Pointer relocation commands allow you to change the current location of the pointer within the text buffer.

6.6.3.1 Beginning — The Beginning (B) command moves the current location of the pointer to the beginning of the text buffer.

The syntax of the command is:

```
B
```

There are no arguments.

For example, assume the buffer contains:

```
MOVB    5(R1),@R2
ADD     R1,(R2)+
CLR     @R2
MOVB    6(R1),@R2
```

The B command moves the pointer to the beginning of the text buffer.

```
*B$$
```

The text buffer now looks like this:

```
*MOVB    5(R1),@R2
ADD     R1,(R2)+
CLR     @R2
MOVB    6(R1),@R2
```

6.6.3.2 Jump — The Jump (nJ) command moves the pointer past a specified number of characters in the text buffer.

The syntax of the command is:

```
nJ
```

Table 6–6 shows the arguments for the J command.

Table 6–6: Jump Command Arguments

Argument	Function
(+ or –) n	Moves the pointer (forward or backward) n characters
0	Moves the pointer to the beginning of the current line (equivalent to 0A)
/	Moves the pointer to the end of the text buffer (equivalent to /A)
=	Moves the pointer backward n characters, where n equals the length of the last text argument used

Negative arguments move the pointer toward the beginning of the buffer; positive arguments move it toward the end. Jump treats carriage returns, line feeds, and form feed characters the same as any other characters, counting one buffer position for each one.

The following examples illustrate the J command.

```
*3J$$
```

This command moves the pointer ahead three characters.

```
*-4J$$
```

This command moves the pointer back four characters.

```
*$$CABC$$-J$$
```

This command moves the pointer so that it immediately precedes the first occurrence of ABC in the buffer.

6.6.3.3 Advance – The Advance (nA) command is similar to the Jump command, except that it moves the pointer a specific number of lines (rather than single characters) and leaves it positioned at the beginning of the line.

The syntax of the command is:

nA

Table 6–7 lists the arguments for the A command and their functions.

Table 6–7: Advance Command Arguments

Argument	Function
n	Moves the pointer forward *n* lines and positions it at the beginning of the *n*th line
-n	Moves the pointer backward past *n* carriage returns and line feed combinations and positions it at the beginning of the -*n*th line
0	Moves the pointer to the beginning of the current line (equivalent to 0J)
/	Moves the pointer to the end of the text buffer (equivalent to /J)

The following examples use the A command.

```
*3A$$
```

This command moves the pointer ahead three lines.

Assume the buffer contains:

```
CLR     SP2
        ↑
```

The following command moves the pointer to the beginning of the current line:

```
*0A$$
```

Now the buffer looks like this:

```
*CLR     @R2
```

6.6.4 Search Commands

Use search commands to locate characters or strings of characters within the text buffer.

NOTE

Search commands always have positive arguments. They search ahead in the file. This means that to search for a character string that has already been written to the output file, you must first close the currently open files (with EX) and then edit the file that was just used for output (with EB).

6.6.4.1 Get – The Get (nG) command is the basic search command in EDIT. It searches the current text buffer for the nth occurrence of a specific text string, starting at the current location of the pointer. If you do not supply the argument n, EDIT searches for the first occurrence of the text object.

The search terminates when EDIT either finds the nth occurrence or encounters the end of the buffer. If the search is successful, EDIT positions the pointer to follow the last character of the text object. EDIT notifies you of an unsuccessful search by printing *?EDIT-F Search failed*. In this instance, EDIT positions the pointer after the last character in the buffer.

The syntax of the command is:

nGtext

The argument n must be positive. If you omit it, EDIT assumes it to be 1.

The text string may be any length and must immediately follow the G command. EDIT makes the search on the portion of the text between the pointer and the end of the buffer.

For example, assume the pointer is at the beginning of the buffer shown below.

```
*MOV     PC,R1
ADD      #OPEN-.,R1
MOV      #"EC",R2
CLR      @R2
MOVB     3(R1),@R2
ADD      R1,(R2)+
CLR      @R2
MOVB     3(R1),@R2
```

The following command searches for the first occurrence of the characters ADD following the pointer and places the pointer after the searched characters.

```
*2ADD<ESC>
```

Now the buffer looks like this:

```
MOV     R0,R1
@R0+    *@R1,-(R2)
```

The next command searches for the third occurrence of the characters @R2 following the pointer and leaves the pointer immediately following the text object.

```
*3@R2$$
```

The buffer is changed to:

```
MOV     R1,-(R2)*
CLR     @R2+
```

After successfully completing a search command, EDIT positions the pointer immediately following the text object. Using a search command in combination with −J places the pointer in front of the text object, as follows:

```
*@TEST$$-J$$
```

This command combination places the pointer before TEST in the text buffer.

6.6.4.2 Find – The Find (nF) command starts at the current pointer location and searches the entire input file for the nth occurrence of the text string. If EDIT does not find the nth occurrence of the text string in the current buffer, it automatically performs a Next command and continues the search on the new text in the buffer. When the search is successful, EDIT leaves the pointer immediately following the nth occurrence of the text string.

If the search fails (that is, EDIT detects the end-of-file for the input file and does not find the nth occurrence of the text string), EDIT prints *?EDIT-F-Search failed*. In this instance, EDIT positions the pointer at the beginning of an empty text buffer. When you use the F command, EDIT deletes the contents of the buffer after writing it to the output file.

The syntax of the command is:

nF*text*

The argument n must be positive. EDIT assumes it to be 1 if you do not supply another value.

You can use an F command to copy all remaining text from the input file to the output file by specifying a nonexistent text object. The Find command functions like a combination of the Get and Next commands.

The following example uses the F command.

```
*ZFRED$$        G:R1,.@R2$$
```

This command searches the entire input file for the second occurrence of the text string MOVB 6(R1),@R2. EDIT places the pointer following the text string. EDIT writes the contents of each unsuccessfully searched buffer to the output file.

6.6.4.3 Position -- The Position (nP) command is identical to the Find (F) command with one exception. The F command transfers the contents of the text buffer to the output file as each page is unsuccessfully searched, but the P command deletes the contents of the buffer after it is searched without writing any text to the output file.

The syntax of the command is:

nPtext

The argument n must be positive. If you omit it, EDIT assumes it to be 1.

The nP command searches each page of the input file for the nth occurrence of the text object, starting at the pointer and ending with the last character in the buffer. If EDIT finds the nth occurrence, it positions the pointer following the text object, deletes all pages preceding the one containing the text object, and positions the page containing the text object in the buffer.

If the search is unsuccessful, EDIT clears the buffer but does not transfer any text to the output file. EDIT positions the pointer at the beginning of an empty text buffer.

The P command is a combination of the Get, Delete, and Read commands; it is most useful as a means of placing the pointer in the input file. For example, if your aim in the editing session is to create a new file from the second half of the input file, a P search saves time.

The following example uses the P command.

```
*P3$$
```

This command searches the input file for the first occurrence of the text object, 3. EDIT positions the pointer after the text object.

```
*PL$$
INPUT FILE PAGE 3
```

This command lists on the terminal the current line up to the pointer.

6.6.5 Text Listing Commands

Two EDIT commands print lines of text on the terminal: the nL (List) command and the V (Verify) command.

6.6.5.1 List -- The List (nL) command prints at the terminal lines of text as they appear in the buffer.

The syntax of the command is:

```
nL
```

An argument preceding the L command indicates the portion of text to print. For example, the command, 2L, prints on the terminal the text beginning at the pointer and ending with the second end-of-line character. The pointer is not altered by the L command. Table 6–8 lists arguments and their effect on the List command.

Table 6–8: List Command Arguments

Argument	Function
n	Prints all the textual n lines beginning at the pointer and ending with the nth end-of-line character
-n	Prints all characters beginning with the first character on the -nth line and terminating at the pointer
0	Prints the current line up to the pointer. Use this command to locate the pointer within a line
/	Prints the text between the pointer and the end of the buffer

These examples illustrate the use of the L command.

```
*-2L$$
```

This command prints all characters starting at the beginning of the second preceding line and ending at the pointer.

```
*4L$$
```

This line prints all characters beginning at the pointer and terminating at the fourth carriage return and line feed combination.

Assuming the pointer location is:

```
MOVB    5(R1),@R2
ADD     R1,(R2)+
```

The following command prints the previous one and one-half lines up to the pointer:

```
*-1L$$
```

The terminal output now looks like this:

```
MOVB    5(R1),@R2
ADD
```

6.6.5.2 Verify — The Verify (V) command prints at the terminal the entire line in which the pointer is located. It provides a ready means of determining the location of the pointer after a search completes and before you give any editing commands. (The V command combines the two commands 0LL.) You can also type the V command after an editing command to allow proofreading of the results.

The syntax of the command is:

```
V
```

No arguments are allowed with the V command. The location of the pointer does not change.

6.6.6 Text Modification Commands

You can use the following commands to insert, change, reference, and delete text in the text buffer.

6.6.6.1 Insert — The Insert (I) command is the basic command for inserting text. EDIT inserts the text you supply at the location of the pointer and then places the pointer after the last character of the new text.

The syntax of the command is:

```
Itext
```

No arguments are allowed with the insert command. The text string is limited only by the size of the text buffer and the space available. All characters are legal, except ESCAPE which terminates the text string.

NOTE

If you forget to type the I command, the editor will interpret the text as commands.

EDIT automatically protects the text buffer from overflowing during an insert. If the I command is the first command in a multiple command line, EDIT ensures that there will be enough space for the insert to be executed at least once. If repetition of the command exceeds the available memory, an error message prints.

The following example illustrates the I command.

```
*MOV     #BUFF,R2
*MOV     #LINE,R1
*MOVB    -1(R2)+,(R2)++
*
```

This command inserts the text at the current location of the pointer and leaves the pointer positioned after R0.

DIGITAL recommends that you insert large amounts of text into the file in small sections rather than all at once. This way, you are less vulnerable to loss of time and effort due to machine failure or human error. This is the recommended procedure for inserting large amounts of text:

1. Open the file with the EB command.
2. Insert or edit a few pages of text.
3. Insert a unique text string (like mrkple) to mark your place.
4. Use the Exit command to preserve the work you have done so far.
5. Start again, using the F command to search for the unique string you used to mark your place.
6. Delete your marker and continue editing.

6.6.6.2 Delete — The Delete (nD) command is a character-oriented command that deletes n characters in the text buffer, beginning at the current location of the pointer.

The syntax of the command is:

nD

If you do not specify n, EDIT deletes the character immediately following the pointer. On completion of the D command, EDIT positions the pointer immediately before the first character following the deleted text. Table 6–9 lists arguments for the D command.

Table 6–9: Delete Command Arguments

Argument	Function
n	Deletes n characters following the pointer. Places the pointer before the first character following the deleted text.
-n	Deletes n characters preceding the pointer. Places the pointer before the first character following the deleted text.
0	Deletes the current line up to the pointer. The position of the pointer does not change (equivalent to 0D).
/	Deletes the text between the pointer and the end of the buffer. Positions the pointer at the end of the buffer (equivalent to 0)
-	Deletes -n characters, where n equals the length of the last text argument used.

The following examples illustrate the use of the D command.

This command deletes the two characters immediately preceding the pointer.

```
*&&-MOV  R1,-(R2)$
```

This command string deletes the text string MOV R1 ,-(R2) in combination with a search command deletes the indicated text string).

Assume the text buffer contains the following:

```
MOV     R1,-(R2)+
CLR     @R2
```

The following command deletes the current line up to the pointer:

```
-0K$
```

The buffer now contains:

```
CLR     @R2
```

6.6.5.3 Kill — The Kill (nK) command removes n lines of text (including the carriage return and line feed characters) from the page buffer, beginning at the pointer and ending with the nth end-of-line.

The syntax of the command is:

nK

EDIT places the pointer at the beginning of the line following the deleted text. Table 6–10 describes each argument and its effect on the Kill command.

Table 6–10: Kill Command Arguments

Argument	Function
n	Removes the character string (including the carriage return and line feed combination) beginning at the pointer and ending at the nth end-of-line.
-n	Removes the current line up to the pointer and n full lines preceding the current line. Thus, if the pointer is at the center of a line, the modifier -1 deletes one and one-half lines preceding it.
0	Removes the current line up to the pointer (equivalent to 0D).
/	Removes the characters beginning at the pointer and ending with the last line in the text buffer (equivalent to /D).

The following examples illustrate the K command.

```
*2K$
```

This command deletes lines starting at the current location of the pointer and ending at the second carriage return and line feed combination.

Assume the text buffer contains the following:

```
ADD     R1,(R2)+
CLR     R2
MOVB    B(R1),@R2
```

This command removes the characters beginning at the pointer and ending with the last line in the text buffer:

```
*/ETB
```

The buffer now contains:

```
ADD     R1,(R2)+
CLR
```

Kill and Delete commands perform the same function, except that Kill is line-oriented and Delete is character-oriented.

6.6.6.4 Change – The Change (nC) command changes a specific number of characters preceding or following the pointer.

The syntax of the command is:

nCtext

A C command is equivalent to a Delete command followed by an Insert command. You must insert a text object following the nC command. Table 6–11 lists each argument and its effect on the C command.

Table 6–11: Change Command Arguments

Argument	Function
n	Replaces n characters following the pointer with the specified text. Places the pointer after the inserted text.
-n	Replaces n characters preceding the pointer with the specified text. Places the pointer after the inserted text.
0	Replaces the current line up to the pointer with the specified text. Places the pointer after the inserted text (equivalent to 0X).
/	Replaces the text beginning at the pointer and ending with the last character in the buffer. Places the pointer after the inserted text (equivalent to X).
=	Replaces n characters with the indicated text string, where n represents the length of the last text argument used

The size of the text is limited only by the size of the text buffer and the space available. All characters are legal except ESCAPE which terminates the text string.

If the C command is to be executed more than once (that is, it is enclosed in angle brackets) and if there is enough space available for the command to be entered, it will be executed at least once (provided it appears first in the command string). If repetition of the command exceeds the available memory, an error message prints.

The following examples illustrate the C command.

```
*5C*#VECT*$
```

This command replaces the five characters to the right of the pointer with #VECT.

Assume the text buffer contains the following:

```
CLR     @R2
MOV$    5(R1),@R2
```

The next command replaces the current line up to the pointer with the specified text.

```
*0C#LOOP1*$
```

The buffer now contains:

```
CLR     #R2
ADD$    5(R1),@R2
```

You can use =C with a Get command to replace a specific text string. Here is an example:

```
*GF(FIVE)=C(FIVE)*$
```

This command finds the text string FIFTY and replaces it with FIVE.

6.6.6.5 Exchange — The Exchange (nX) command is similar to the change command, except that it changes lines of text instead of a specific number of characters.

The syntax of the command is:

nXtext

The nX command is identical to an nK command followed by an Insert command. Table 6–12 lists the Exchange command arguments.

Table 6–12: Exchange Command Arguments

Argument	Function
n	Replaces n lines, including the carriage return and line feed characters, following the pointer. Places the pointer after the inserted text.
-n	Replaces n lines, including the carriage return and line feed characters, preceding the pointer. Positions the pointer after the inserted text.
0	Replaces the current line up to the pointer with the specified text. Positions the pointer after the inserted text (equivalent to 0C).
/string/	Replaces the text beginning at the pointer and ending with the last character in the buffer with the specified text (equivalent to 3C). Positions the pointer after the inserted text.

All characters are legal in the text string except ESCAPE which terminates the text.

If the X command is enclosed within angle brackets to allow more than one execution, and if there is enough memory space available for the X command to be entered, EDIT executes it at least once (provided it is first in the command string). If repetition of the command exceeds the available memory, an error message prints.

The following example illustrates the X command.

```
*2:400   R1,(R2)+
CLR      @R2
**
*
```

This command exchanges the two lines to the right of the pointer with the text string.

6.6.7 Utility Commands

During the editing session, you can store text in external buffers and subsequently restore this text when you need it later in the editing session. The following sections describe the commands that perform this function.

6.6.7.1 Save – The Save (nS) command lets you store text in an external buffer called a save buffer (described previously in Section 6.5), and subsequently insert it in several places in the text.

The syntax of the command is:

nS

The Save command copies n lines, beginning at the pointer, into the save buffer. The S command operates only in the forward direction; therefore, you cannot use a negative argument. The Save command destroys any previous contents of the save buffer; however, EDIT does not change the location of the pointer or the contents of the text buffer.

If you specify more characters than the save buffer can hold, EDIT prints *?EDIT-F-Insufficient memory*. None of the specified text is saved.

For example, assume the text buffer contains the following assembly language subroutine:

```
;SUBROUTINE MSGTYP
;WHEN CALLED, EXPECTS R0 TO POINT TO AN
;ASCIZ MESSAGE THAT ENDS IN A ZERO BYTE,
;TYPES THAT MESSAGE ON THE USER TERMINAL

MSGTYP: TSTB   (R0)             ;DONE?
        BEQ    #DONE            ;YES,RETURN
#LOOP:  TSTB   @#177564         ;NO-IS TERMINAL READY?
        BPL    #LOOP            ;WAIT
        MOVB   (R0)+,@#177566   ;YES,PRINT CHARACTER
        BR     MSGTYP           ;LOOP
#DONE:  RTS    PC               ;RETURN
```

The following command stores the entire subroutine in the save buffer (assuming the pointer is at the beginning of the buffer):

```
*15S*
```

You can insert the contents of the save buffer into a program whenever you choose by using the Unsave command.

6.6.7.2 Unsave—The Unsave (U) command inserts the entire contents of the save buffer into the text buffer at the pointer and leaves the pointer positioned following the inserted text. You can use the U command to move blocks of text or to insert the same block of text in several places.

Table 6–13 lists the U commands and their functions.

Table 6–13: Unsave Command Arguments

Command	Function
U	Inserts the contents of the save buffer into the text buffer
0U	Clears the save buffer and reclaims the area for text

The only argument the U command accepts is 0.

The contents of the save buffer are not destroyed by the U command (only by the 0U command) and can be unsaved as many times as desired. If the Unsave command causes an overflow of the text buffer, the *?EDIT-F-Insufficient memory* error message prints, and the command does not execute.

For example:

```
*U$$
```

This command inserts the contents of the save buffer into the text buffer.

6.6.7.3 Macro – The Macro (M) command inserts a command string, called a macro, into the EDIT macro buffer.

Table 6–14 lists the M commands and their functions.

Table 6–14: M Command and Arguments

Command	Function
M*command string*	Stores the command string in the macro buffer
0M or M	Clears the macro buffer and reclaims the area for text

The slash (/) represents the delimiter character. The delimiter is always the first character following the M command, and can be any character that does not appear in the macro command string itself.

Starting with the character following the delimiter, EDIT places the command string characters into its internal macro buffer until the delimiter is encountered again. At this point, EDIT returns to command mode. The Macro command does not execute the command string; it merely stores the command string so that the Execute Macro (EM) command can execute later. The Macro command does not affect the contents of the text or save buffers.

All characters except the delimiter are valid macro command string characters, including single ESCAPEs to terminate text commands. All commands, except the M and EM commands, are valid in a command string macro.

In addition to using the 0M command, you can type the M command immediately followed by two identical characters (assumed to be delimiters) and two ESCAPE characters to clear the macro buffer.

The following examples illustrate the use of the M command.

```
*M//$$
```

This command clears the macro buffer.

```
*M/J0$$R1$$/$$
```

This command stores a macro to change R0 to R1.

NOTE

Be careful to choose infrequently used characters as macro delimiters; choosing frequently used characters can lead to errors. For example:

```
*M GMOV R0$=CADD R1$ $$
?EDIT-F-No file open for input
```

In this case, it was intended that the macro be GMOV R0$ = CADD R1$, but since the delimiter character (the character following the M) is a space, the space following MOV is used as the second delimiter, terminating the macro. EDIT then returns an error when it interprets the R as a Read command.

6.6.7.4 Execute Macro — The Execute macro (nEM) command executes a command string previously stored in the macro buffer by the M command.

The syntax of the command is:

nEM

The argument n must be positive. The macro is executed n times and then returns control to the next command in the original command string.

The following example uses the EM command.

```
*M/0$50$-C1$/$$
*E1000EM$$
?EDIT-F-Search failed
*
```

This command sequence stores a command in the macro buffer and then executes that command. EDIT prints an error message when it reaches the end of the buffer. (This macro changes all occurrences of R0 in the text buffer to R1.)

```
*IMOV PC,R1$ZEMICLR @R2$$
*
```

This command inserts MOV PC,R1 into the text buffer and then executes the command in the macro buffer twice before inserting CLR @R2 into the text buffer.

6.6.7.5 Edit Version — The Edit Version (EV) command displays the version number of the editor in use on the console terminal.

The syntax of the command is:

EV

This example displays the running version of EDIT:

```
*EV$$
005.00
*
```

6.6.7.6 Uppercase and Lowercase Commands – If you have a terminal that has both uppercase and lowercase characters as part of your hardware configuration, you can take advantage of two editing commands, Edit Lower (EL) and Edit Upper (EU).

When the editor is started with the EDIT command, uppercase mode is assumed – that is, all characters you type are automatically translated to uppercase. To allow processing of both uppercase and lowercase characters, enter the Edit Lower command. For example:

```
*EL$$
*I You can enter text and commands in UPPER and lower case.$$
*
```

The editor now accepts and echoes uppercase and lowercase characters received from the keyboard, and prints text on the terminal in uppercase and lowercase.

To return to uppercase mode, use the Edit Upper command:

```
*EU$$
```

Control also reverts to uppercase mode on exit from the editor (with EX or CTRL/C).

Note that when you issue an EL command, you can enter EDIT commands in either uppercase or lowercase. Thus, the following two commands are equivalent:

```
*GTEXT$=Chou text$V$$

*gTEXT$=chou text$v$$
```

The editor automatically translates (internally) all commands to uppercase without reference to EL or EU.

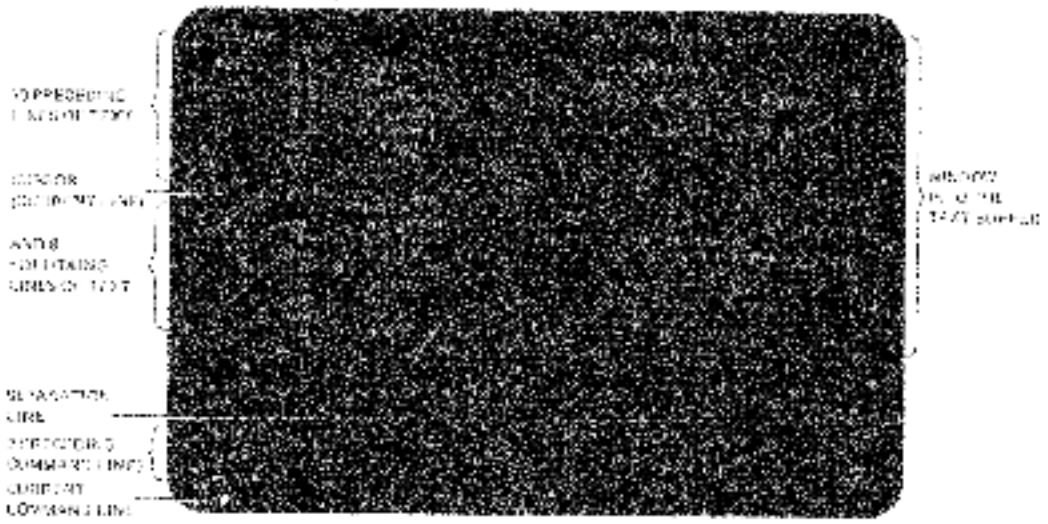
NOTE

When you use EDIT in EL mode, make sure that text arguments you specify in search commands have the proper case. The command GTeXt$, for example, will not match TEXT, text, or any combination other than TeXt.

6.7 Display Editor

In addition to all functions and commands mentioned thus far, the editor can use VT-11 and VS-60 display hardware as part of the system configuration (GT40, GT44, DECLAB 11/40, DECLAB 11/34). The most obvious feature provided by this hardware is the use of the display screen rather than the console terminal for printing terminal input and output. Another feature is that the top of the display screen functions like a window into the text buffer. When all the features of the display editor are in use, a 12-inch screen displays text as shown in Figure 6–1.

Figure 6–1: Display Editor Format, 12-Inch Screen



The major advantage is that you can see immediately where the pointer is, because it appears between characters on the screen as a blinking L-shaped cursor. Remember that pressing the RETURN key causes both a carriage return and a line feed character to be inserted into the text. Note that if the pointer is placed between a carriage return and line feed, it appears in an inverted position at the beginning of the next line.

In addition to displaying the current line (the line containing the cursor), the 15 lines of text preceding the current line and the 14 lines following it are also in view on a 17-inch screen. Each time you execute a command string (with a double ESCAPE), EDIT refreshes the text portion of the screen so that it reflects the results of the commands you just performed.

The lower section of the 17-inch screen contains eight lines of editing commands. The command line you are currently entering is last, preceded by the most recent command lines. A horizontal line of dashes separates this section from the text portion of the screen. As you enter new command lines, previous command lines scroll upward off the command section so that only eight command lines are ever in view.

A 12-inch screen displays 20 lines of text and 4 command lines, as shown in Figure 6–1.

6.7.1 Using the Display Editor

The display features of the editor are automatically invoked whenever the system scroller is in use (a monitor GT ON command is in effect) and you start the editor. However, if the system does not contain display hardware, the display features are not enabled.

If the system contains display hardware and you wish to use the screen during the editing session, you can activate it in one of two ways, whether or not the display is in use. (All editing commands and functions previously discussed in this chapter are valid for use.)

1. If the scroller is in use (the GT ON monitor command is in effect), EDIT automatically uses the screen for display of text and commands. However, it rearranges the scroller so that a window into the text buffer appears in the top two-thirds of the screen, while the bottom third displays command lines. This arrangement is shown in Figure 6–1.

   You can use the Edit Console (EC) command to return the scroller to its normal mode so that text and commands use the full screen, and the window is eliminated.

   The command is:

   ```
   EC
   ```

   This example uses the EC command:

   ```
   *6A$ 0 2! $$
   ```

   This command lists the second and third lines of the current buffer on the screen; there is no window into the text buffer at this point.

   EDIT ignores subsequent EC commands if the window into the text buffer is not being displayed.

   To recall the window, use the Edit Display (ED) command:

   ```
   *ED
   ```

   The screen is again arranged as shown in Figure 6–1.

2. Assume the scroller is not in use (the GT ON command is not in effect). When you call EDIT with the EDIT command, an asterisk appears on the console terminal. Use the ED command at this time to provide the window into the text buffer; however, commands continue to be echoed to the console terminal.

   When you use ED in this case, it must be the first command you issue. Otherwise, it becomes an invalid command (the memory used by the display buffer and code, amounting to over 600 words, is reclaimed as working space). You cannot use the display again until you load a fresh copy of EDIT.

While the display of the text window is active, EDIT ignores ED commands.

Typing the EC command clears the screen and returns all output to the console terminal.

NOTE

After completing an editing session that uses the ED command, clear the screen by typing the EC command or by returning to the monitor and using the monitor RESET command. Failure to do this may cause unpredictable results.

6.7.2 Immediate Mode

An additional mode is available to provide easier and faster interaction during the editing session. This mode is called immediate mode, which combines the most frequently used functions of the text and command modes — namely, repositioning the pointer and deleting and inserting characters.

You can use immediate mode only when the VT-11 display hardware is active and the editor is running. To enter immediate mode type two ESCAPEs (only) in response to the command mode asterisk:

```
*
```

The editor responds by displaying an exclamation point (!) on the screen. The exclamation character remains on the screen as long as immediate mode is in effect.

Once you enter immediate mode, you can use only the commands in Table 6–15. Any other commands or characters are treated as text to be inserted. None of these commands echoes, but the text appearing on the screen is constantly refreshed and updated during the editing process.

To return control from immediate mode to normal command mode, type a single ESCAPE. The editor responds with an asterisk and you may proceed using all normal editing commands. Immediate mode commands you type at this time will be accepted as command mode input characters. To return control to the monitor from immediate mode, type ESCAPE to return to command mode, then type CTRL/C followed by two ESCAPEs.

Table 6–13: Immediate Mode Commands

Command	Function
CTRL/N	Advances the pointer (cursor) to the beginning of the next line (equivalent to A)
CTRL/P	Moves the pointer (cursor) to the beginning of the previous line (equivalent to -A)
CTRL/D	Moves the pointer (cursor) forward by one character (equivalent to J)
CTRL/V	Moves the pointer (cursor) backward by one character (equivalent to -J)
DELETE or RUBOUT	Deletes the character immediately preceding the pointer (cursor) (equivalent to -D)
ESCAPE or ALTMODE	Single character returns control to command mode; double character redirects control to immediate mode
Any character other than those above	Inserts the character as text positioned immediately before the pointer (cursor) (equivalent to I)

6.8 EDIT Example

The following example illustrates the use of EDIT commands to change a program stored on the device DK:. Sections of the terminal output are coded by letter, and corresponding explanations follow the example.

```
A  EDIT TEST1.MAC
   *0$$
   */L$$
   ;TEST PROGRAM
   START:  MOV     #1000,SP        ;INITIALIZE STACK
           MOV     #MSG,R0         ;POINT R0 TO MESSAGE
B          JSR     PC,@#61'P       ;PRINT IT
           HALT                    ;STOP
   MSG:    .ASCII/IT WORKS/
           .BYTE 15
           .BYTE 12
           .BYTE 0
C  *B$1Js50$$
   *GPROGRAM$$
D  *OL$$
   ;PROGRAM#I TO TEST SUBROUTINE MSG'YP, TYPES
E  I"THE TEST PROGRAM WORKS"
   ION THE TEM(\IM\RMINAL$$
F  *P,ASCII/$$
   *BOTHE TEST PROGRAM WORKS$$
   *P,BYTE'%
G  *F,BYTE 0$V$$
           .BYTE 0
```

```
*I
        .END

$S/L*$
!PROGRAM TO TEST SUBROUTINE MSGTYP, TYPES
!"THE TEST PROGRAM WORKS"
!ON THE TERMINAL

START:      MOV#1000,SP         !INITIALIZE STACK
            MOV#MSG,R0          !POINT R0 TO MESSAGE
            JSR PC,MSGTYP       !PRINT IT
            HALT                !STOP
MSG:        .ASCII/THE TEST PROGRAM WORKS/
            .BYTE 15
            .BYTE 12
            .BYTE 0
            .END

*EX/$

*
```

A Calls the EDIT program and prints *. The input file is TEST1.MAC; the output file is TEST2.MAC. Reads the first page of input into the buffer.

B Lists the buffer contents.

C Places the pointer at the beginning of the buffer. Advances the pointer one character (past the ;) and deletes the TEST.

D Positions the pointer after PROGRAM and verifies the position by listing up to the pointer.

E Inserts text. Uses DELETE to correct typing error.

F Searches for .ASCII/ and changes IT WORKS to THE TEST PROGRAM WORKS.

G Types CTRL/X to cancel the P command. Searches for .BYTE 0 and verifies the location of the pointer with the V command.

H Inserts text. Returns the pointer to the beginning of the buffer and lists the entire contents of the buffer.

I Closes the input and output files after copying the current text buffer as well as the rest of the input file into the output file. EDIT returns control to the monitor.

6.9 EDIT Error Conditions

The editor prints an error message whenever it detects an error. EDIT checks for three general types of error conditions: syntax errors, execution errors, and macro execution errors. This section describes the error message form for each type of error condition.

Before it executes any commands, EDIT first scans the entire command string for errors in command syntax, such as illegal arguments or an illegal combination of commands. If the editor finds an error of this type, it prints a message of this form:

```
?EDIT-F-Message(#no command(s) executed
```

You should retype the command.

If a command string is syntactically correct, EDIT begins execution. Execution errors, such as buffer overflow or input and output errors, can still occur. In this case, EDIT prints a message of the form:

```
?EDIT-F-Message
```

EDIT executes all commands preceding the one in error. It does not execute the command in error or any commands that follow it.

When an error occurs during execution of a macro, EDIT prints a message of the form:

```
?EDIT-F-Message in macro(#n command(s) executed
```

or

```
?EDIT-F-Message in macro
```

Most errors are syntax errors. These are usually easy to correct before execution.

The *RT-11 System Message Manual* contains a complete list of the EDIT error messages, along with recommended corrective action for each error.

Part IV
Setting Hardware Characteristics

Part IV shows you how to set hardware characteristics for your terminal, printer, and system clock.

Chapter 7
Hardware Characteristics Program (SETUP)

The SETUP program sets terminal (video and hard-copy), line printer, and system clock modes using English-language commands. SETUP commands are especially useful when you include them in start-up indirect command files or IND control files.

You cannot set a mode for a device that does not recognize (support) the mode; the device will ignore the mode. Refer to your device's user guide for guidance in selecting proper SETUP modes.

SETUP commands are explained in Section 7.1. SETUP modes are listed and described in Section 7.2. Terminal modes are located in Section 7.2.1, terminal and printer modes in Section 7.2.2, and system clock modes in Section 7.2.3.

Section 7.3 lists SETUP controls that provide HELP, and let you save, reset, and show the modes selected from Section 7.2.

Section 7.4 contains a table which lists all SETUP modes and commands with applicable devices.

7.1 SETUP Commands

This section describes the SETUP mode and SETUP control commands.

7.1.1 Mode Commands

To issue a SETUP mode command, type in response to the keyboard monitor prompt (.):

```
.SETUP destination mode[,mode...]
```

The variable destination can be TERMINAL (or TT), PRINTER (or LP or LS), or CLOCK:

- Destination TERMINAL (device TT) sends modes to your terminal. In the command, TT can be substituted for TERMINAL. TERMINAL is the default, and need not be specified.
- Destination PRINTER (device LP or LS) sends modes to your line printer. If you specify destination PRINTER, SETUP looks first for device LP, and if not found then looks for device LS to send the mode to your printer. PRINTER modes are supported by LA50, LA100, and LQP02 printers.

In the command, LP or LS can be substituted for PRINTER. However, if LP or LS is explicitly specified, SETUP only looks for that particular device and returns an error message if not found. Destination PRINTER or LP or LS must be specified to send modes to your printer.

- Destination CLOCK sends modes to your system clock. You need not specify destination CLOCK to send clock modes to your system clock.

The variable mode represents a mode from Section 7.2 that you want to set. 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,DISCLOCK,WRAP
```

7.1.2 Control Commands

To issue a SETUP control command, type in response to the keyboard monitor prompt (.):

```
.SETUP destination control,control,...)
```

The variable destination represents the device to which you send the control. Refer to the description of the destination variable in Section 7.1.1.

The variable control represents a control from Section 7.3 that lets you save, reset, or show SETUP modes. You can set several controls in one SETUP control command by separating each control with a comma (,).

7.2 SETUP Modes

The following sections list SETUP modes for terminals, serial line printers, and system clocks.

Default settings for Professional integrated terminals are indicated with an asterisk (*) except where the default is a numeric argument following the mode. Numeric argument defaults are listed in the description. For VT200 and VT100 series defaults, refer to your terminal user guide.

7.2.1 Terminal Modes

The SETUP program lets you set terminal modes for:

- Professional integrated terminals used with Professional 325 and 350 processors
- VT200 series terminals used with other PDP-11 processors
- VT100 series terminals used with other PDP-11 processors
- Hard-copy terminals that accept standard ASCII escape sequences, such as DIGITAL's Correspondent

SETUP is the only means of setting certain modes for the Professional 325 and 350. The VT200 series terminals can be set up using their local menu or SETUP commands. You can set video modes on VT100 series terminals using the local set-up mode as described in the appropriate VT100 series user's guide, or with SETUP commands. Hard-copy terminals can be set up using the particular terminal's set-up procedure or with SETUP commands.

Some SETUP modes are valid for only Professional integrated terminals, as indicated.

The following SETUP modes are suitable for only your terminal. SETUP modes that you can send to your terminal or printer are listed in Section 7.2.3.

The shortest valid abbreviation is underlined.

Mode	Description

ANSWERBACK:/message/	Applies to Professional terminals only. ANSWERBACK mode 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 carriage returns in the message. Let SETUP prompt for input by typing:
	
	`>SETUP ?0`
	`Characteristics?:ANSWERBACK:/message/`
	
	You can now use lowercase letters and embed carriage returns in your message. The message will continue to be read until the final delimiter (/) is reached, regardless of carriage returns.

BLOCK* UNDERSCORE	Applies to Professional terminals only. When you set BLOCK, the cursor appears as a blinking block (■) on your screen. When you set UNDERSCORE, the cursor appears as a blinking underline (_) on your screen.

CAPS*
SHIFT

Applies to Professional terminals only. When you set CAPS, pressing the LOCK key causes all letters to be entered as uppercase. In this mode the LOCK key serves the same function as the CAPS LOCK key on the VT100 series keyboard.

When you set SHIFT, pressing the LOCK key causes 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.

CLICK*
NOCLICK

Applies to Professional terminals only. Setting CLICK causes the keys to click when you press them.

Setting NOCLICK causes the keys to make no sound when you press them.

[COLOR1] [ON COLOR2]

Applies to Professional color terminals only. When you set COLOR1 ON COLOR2 you specify the color of the text (COLOR1), and the color of the video screen background (COLOR2). You can specify any of the following colors for COLOR1 and COLOR2: red, yellow, blue, green, black, white, magenta, and cyan.

For example, the following command displays green text on a blue background:

```
.SETUP GREEN ON BLUE
```

Colors can also be specified for just the text or video screen background. For example, the following command assigns blue for the text but does not change the background color:

```
.SETUP BLUE
```

The following command assigns blue for the video 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.

CURSOR* NOCURSOR	Applies to Professional terminals only. When you set CURSOR, the terminal screen displays a blinking cursor. The cursor's form is set using SETUP BLOCK or SETUP UNDERSCORE. When you set NOCURSOR, the terminal screen does not display a cursor.
DARK* LIGHT	When you set DARK, the screen displays light characters on a dark background, or reverses text and background colors on a Professional color video monitor. When you set LIGHT, the screen displays dark characters on a light background, or reverses text and background colors on a Professional color video monitor.
GENERIC100	Does not apply to VT100 series terminals. When you set GENERIC100, your Professional or VT200 series terminal replies to a device attributes request with a code sequence identifying the terminal as a non-specific 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 series terminals when SETUP GENERIC100 is set. For a complete explanation of the device attributes exchange for the Professional terminal, and therefore the VT200 series terminals, see the *RT–11 Software Support Manual*.
INTERLACE NOINTERLACE*	Applies to VT100 series terminals only. When you set INTERLACE, the screen can display high-resolution graphics. Set NOINTERLACE when you are not using high-resolution graphics.
MARGINBELL NOMARGINBELL*	Applies to Professional terminals only. MARGINBELL causes the terminal to beep when the cursor is eight characters from the right margin. NOMARGINBELL disables the terminal's beep when the cursor is eight characters from the right margin.

NEWLINE NONEWLINE*	When you set NEWLINE, pressing the RETURN key generates a <CR><LF> (carriage return, line feed) combination. RT-11 automatically generates a line feed, so setting NEWLINE sends two line feeds. When you set NONEWLINE, pressing the RETURN key generates only a carriage return. However, RT–11 generates a line feed so setting NONEWLINE generates one <CR><LF> (carriage return, line feed) combination.
REPEAT* NOREPEAT	When you set REPEAT, a key automatically repeats at the rate of about 30 characters per second when you press the key for more than 1/2 second. When you set NOREPEAT, a character is displayed only once no matter how long you press the key.
SMOOTH JUMP*	When you set SMOOTH, new lines are scrolled on the screen at a maximum of 6 lines per 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. When you set JUMP, new lines are scrolled on the screen as fast as the computer sends them to the terminal. Reading lines in JUMP mode is difficult at higher baud rates. For processors other than the Professional 300 series, NOAUTOX must be set using the local set-up mode or the local set-up menu when you use JUMP.
VT100* VT52	When you set VT100 mode, the terminal transmits and decodes 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 Software Support Manual* for a complete explanation of the device attributes exchange.

When you set VT52 mode, the terminal transmits and decodes according to the requirements of the earlier VT52 terminal. VT52 mode is not supported for the Professional terminal under RT–11 V5.1.

132COLUMNS
80COLUMNS*

When you set 132COLUMNS, your video terminal displays 132 characters per line. 132COLUMNS cannot be used with the Professional series computers when running under the FB monitor.

When you set 80COLUMNS, your video terminal displays 80 characters per line.

7.2.2 Terminal and Printer Modes

The following SETUP modes can be sent to your terminal (video or hard-copy) or printer, depending on the description. If no qualification is given, a mode can be sent to either. You do not need to specify destination TERMINAL (or TT) to send a command to your terminal, but you must specify PRINTER (or LP or LS) to send a command to your line 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.

The shortest valid abbreviation is underlined.

Mode	Description
BOLD NOBOLD*	When you set BOLD, characters printed on your terminal screen or printer are enhanced with bolding. BOLD may not 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.
CLEAR	The SETUP CLEAR command clears the screen, and executes a shift in (SI) function and the DEC 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 DEC standard SOFT RESET:

- Sets the current rendition to NORMAL
- Resets scrolling margins to 1 and 24
- Sets 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

DENSE

When you set DENSE, your dot matrix printer or hard-copy terminal operates at 30 characters/second. It overlaps two sets of dots for a close approximation of letter-quality printing. DENSE prints a more legible character than MEMO.

DOWN

When you set DOWN, the print line in your printer or hard-copy terminal is lowered 1/12 inch relative to the top-of-form. The paper is moved forward (upward) 1/12 inch. DOWN can be used for remote adjustment of the top-of-form position. DOWN requires the partial line adjustment feature; consult your user guide.

DRAFT

When you set DRAFT, the printer or hard-copy terminal is set to an 8 X 9 dot matrix. DRAFT mode is the same as DATA mode used with some printers.

GRAPHIC
TEXT*

Does not apply to VT100 series terminals. Setting GRAPHIC places your terminal or printer in graphic mode. Graphic mode selectively prints 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.

When you set TEXT, your terminal or printer exits from graphic mode. Text mode prints predefined characters from a selected character set.

HORIZONTAL:n

When you set HORIZONTAL, argument n sets the number of characters printed per horizontal inch. Valid arguments for n are 5, 6, 8.25, 10, 12, or 16.5. HORIZONTAL can be used with some hard-copy terminals.

LANG:code

Does not apply to Professional series terminals. When you set LANG: you can determine the character set of your console terminal and line printer. The command sets the keyboard's character set to the specified code.

code will be one of the following:

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*

LETTER

When you set LETTER, the dot matrix serial line printer or hard-copy terminal is automatically set to DENSE and HORIZONTAL:10.

LISTING

When you set LISTING, the dot matrix printer or hard-copy terminal is automatically set to DRAFT and HORIZONTAL:16.5.

MEMO	When you set MEMO, the printer or hard-copy terminal dot matrix changes from 8 X 9 to 15 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.
PAGELENGTH:n	When you set PAGELENGTH, argument n sets the number of lines printed per page. Valid argument parameters are 0 to 252. If the argument is 0, the printer or hard-copy terminal ignores form feed.
TABS[:n,n,n...] NOTABS	When you set TABS:n, argument n represents the column number 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 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.
UNDERLINE NOUNDERLINE*	When you set UNDERLINE, text displayed or printed is underlined. If destination is TERMINAL and SETUP BOLD is in effect, issuing SETUP NOBOLD automatically issues SETUP NOUNDERLINE. See SETUP BOLD. When you set NOUNDERLINE, text displayed or printed is not underlined.
UP	When you set UP, the print line in your printer or hard-copy terminal is raised 1/12 inch relative to the top-of-form. The paper is moved backward (downward) 1/12 inch. UP can be used for remote adjustment of the top-of-form position. Requires support of partial line adjustment feature; check your user guide.

VERTICALn	When you set VERTICAL, argument n sets the number of lines printed per vertical inch. Valid arguments for n are 2, 3, 4, 6, 8, or 12. VERTICAL can be used with some hard-copy terminals.
WRAP NOWRAP*	When you set WRAP, characters you type or print that exceed the right margin are automatically moved to the beginning of the following screen line. When you set NOWRAP, characters you type or print that exceed the right margin overwrite the last character of the current screen line and are lost.

7.2.3 Clock Modes

The SETUP program lets you set modes for system clocks in Professional 300 series and other PDP–11 computers. There are differences in support between Professional 300 series and other PDP–11 computers, as explained.

The following SETUP modes can only be sent to CLOCK, the system clock. System clock modes do not have default settings. The shortest valid abbreviation for each mode is underlined.

Mode	Description
12HOUR 24HOUR	Applies to Professional 300 series computers only. When you set 12HOUR, you convert the contents of the time-of-year clock to 12 hour mode, and put the clock in 12 hour mode. When you set 24HOUR, you convert the contents of the time-of-year clock to 24 hour mode, and put the clock in 24 hour mode. For example, 1:00 PM is represented as 13:00.
BCD NOBCD	Applies to Professional 300 series computers only. When you set BCD (binary coded decimal), you convert the contents of the time-of-year clock to BCD, and put the clock in BCD mode. When you set NOBCD, you convert the contents of the time-of-year clock to binary, and put the clock in binary mode.

CLOCK SHOW	Applies to Professional 300 series computers only. When you set CLOCK SHOW, you print current settings of: Date mode (BCD or binary) 12/24HOUR mode (12 or 24 hour) Daylight saving time (DST) enable (on or off)
DATE[:dd-mm-yy]	When you set DATE with a Professional 300 series computer, you set the specified date into 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 Professional computer maintains the clock, so that you need set it only once. When you set DATE with a PDP-11 computer other than the Professional 300 series, only the monitor date is set.
DST NODST	Applies to Professional 300 series computers only. When you set DST, the time-of-year clock is put in daylight saving time mode. When you set NODST, the time-of-year clock is taken out of daylight saving mode.
TIME[:hh:mm:ss]	When you set TIME on a Professional 300 series computer, you set 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 Professional computer maintains TIME, so that you need set it only once. When you set TIME in a PDP-11 computer other than the Professional 300 series, only the monitor time is set.

7.3 SETUP Controls

SETUP controls provide help, and save, reset, and show the current mode settings. Control modes do not have default settings. The shortest valid abbreviation is underlined.

Control	Description
DEFAULT	Applies to Professional terminals only. DEFAULT sets the terminal modes to the factory settings.
HELP	HELP provides a complete list of the SETUP modes with a one-line explanation of each mode.
RESET	RESET sets the modes to the settings that were set using the most recent SETUP SAVE command, or by a hardware SAVE (see your device's user guide).
SAVE	Applies to Professional terminals only. When you issue the SETUP SAVE command, current SETUP modes are saved and become the default modes. If the SETUP modes are changed, they may be restored by issuing SETUP RESET.
SHOW	Applies to Professional terminals only. When you issue the SETUP SHOW command, current SETUP mode settings are printed on the screen. For example, if the terminal is set for the original (factory-set) default modes, issuing the SETUP SHOW command prints the following report:

```
BLOCK           is ENABLED
CAPS            is ENABLED
CLICK           is ENABLED
CURSOR          is ENABLED
DARK            is ENABLED
JUMP            is ENABLED
LIGHT           is DISABLED
MARGINBELL      is DISABLED
NEWLINE         is DISABLED
REPEAT          is ENABLED
SHIFT           is DISABLED
SMOOTH          is DISABLED
UNDERSCORE      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
```

7.4 Summary of SETUP Modes and Controls

The following table lists all SETUP modes and controls, with devices for which they apply. Column three (Other PDP-11 Terminals) also refers to some hard-copy terminals. The shortest valid abbreviation for each mode is located before the asterisk (*).

Modes and Controls	PC300 Series Terminals	Other PDP-11 Terminals	Printer	PC300 Series Clocks	Other PDP-11 Clocks
A*NSWERBACK	Yes	—	—	—	—
[NO]BC*D	—	—	—	Yes	—
BLO*CK	Yes	—	—	—	—
[NO]BO*LD	Yes	Yes	Yes	—	—
CA*PS	Yes	—	—	—	—
CLE*AR	Yes	Yes	Yes	—	—
[NO]CLI*CK	Yes	—	—	—	—
COLOR	Yes	—	—	—	—
[NO]CU*RSOR	Yes	—	—	—	—
DAR*K	Yes	Yes	—	—	—
DAT*E	—	—	—	Yes	Yes
DEF*AULT	Yes	—	—	—	—
DEN*SE	—	Yes	Yes	—	—
DO*WN	—	Yes	Yes	—	—
DR*AFT	—	Yes	Yes	—	—
[NO]DS*T	—	—	—	Yes	—
GE*NERIC100	Yes	Yes	—	—	—
GRA*PHIC	Yes	Yes	Yes	—	—
HE*LP	Yes	Yes	Yes	Yes	Yes
HO*RIZONTAL	—	Yes	Yes	—	—
[NO]I*NTERLACE	—	Yes	—	—	—
J*UMP	Yes	Yes	—	—	—
LA*NG	—	Yes	Yes	—	—
LE*TTER	—	Yes	Yes	—	—
LIG*HT	Yes	Yes	—	—	—
LIS*TING	—	Yes	Yes	—	—
[NO]MAR*GINBELL	Yes	—	—	—	—
ME*MO	—	Yes	Yes	—	—
[NO]NE*WLINE	Yes	Yes	—	—	—
P*AGELENGTH	—	Yes	Yes	—	—
[NO]REP*EAT	Yes	Yes	—	—	—
RES*ET	Yes	Yes	Yes	—	—
SA*VE	Yes	—	—	—	—
SHI*FT	Yes	—	—	—	—
SHO*W	Yes	—	—	Yes	—
SM*OOTH	Yes	Yes	—	—	—

[NO]TA*BS:n	Yes	Yes	Yes	----	----
TE*XT	Yes	Yes	Yes	----	----
TI*ME	----	----	----	Yes	Yes
[NO]UNDERL*INE	Yes	Yes	Yes	----	----
UNDERS*CORE	Yes	----	----	----	----
UP*	----	Yes	Yes	----	----
VE*RTICAL	----	Yes	Yes	----	----
VT100*	Yes	Yes	----	----	----
VT52*	----	Yes	----	----	----
[NO]WR*AP	Yes	Yes	Yes	----	----
132*COLUMNS	Yes	Yes	----	----	----
80*COLUMNS	Yes	Yes	----	----	----
12*HOUR	----	----	----	Yes	----
24*HOUR	----	----	----	Yes	----

Appendix A
Monitor Command Abbreviations and System Utility Program Equivalents

This appendix provides a table of correspondence between the keyboard monitor commands with their options and the system utility programs with their options. Remember that the syntax you use to issue a keyboard monitor command is different from the syntax that the Command String Interpreter requires for input and output specifications for the system utility programs. Bear in mind that there are many differences between issuing a monitor command and running a utility program.

The following table lists all the keyboard monitor commands and options. A dash under the corresponding system program or option column indicates that the command has no real system program equivalent, that the function is inherent in the keyboard monitor, or that the function is the default mode of operation. The minimum abbreviation for each command and option is in red

Monitor Command	Option	System Utility Program	Option
ABORT		—	—
ASSIGN		—	—
B		—	—
BACKUP		BUP	—
	/DEVICE	BUP	/I
	/RESTORE	BUP	/X
BOOT		DUP	/O
	/FOREIGN	DUP	/Q
	/WAIT	DUP	/W
CLOSE		—	
COMPILE			—
	/ALLOCATE:size		/n
	/ALPHABETIZE	DIBOL	/A
	/BUFFERING	DIBOL	/B
	/CODE:type	FORTRAN	/I:type
	/CROSSREFERENCE[:type1, :type2]	MACRO,DIBOL	/C
	/DIAGNOSE	FORTRAN	/D
	/DIBOL	DIBOL	—

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/DISABLE:type[,...:type]	MACRO	/D
	/ENABLE:type[,...:type]	MACRO	/E
	/EXTEND	FORTRAN	/E
	/FORTRAN	FORTRAN	--
	/HEADER	FORTRAN	/O
	/LA	FORTRAN	/T
	/LIBRARY	MACRO	/M
	/LINENUMBERS	DIBOL,FORTRAN	--
	/NOLINENUMBERS	DIBOL	/O
		FORTRAN	/S
	/LP:[filespec]	--	2nd output spec
	/LIST	DIBOL	/G
	/MACRO	MACRO	--
	/OBJECT:[filespec]	--	1st. output spec
	/NOOBJECT	--	null 1st output spec.
	/ONDEBUG	DIBOL,FORTRAN	/D
	/PAGE:n	DIBOL	/Pn
	/RECORD:length	FORTRAN	/R:length
	/SHOW:type	FORTRAN,MACRO	/L:value
	/NOSHOW:type	MACRO	/N:value
	/STATISTICS	FORTRAN	/A
	/SWAP	FORTRAN	--
	/NOSWAP	FORTRAN	/U
	/TABLES	DIBOL	/S
	/UNITS:n	FORTRAN	/N:n
	/VECTORS	FORTRAN	--
	/NOVECTORS	FORTRAN	/V
	/WARNINGS	DIBOL	
		FORTRAN	/W
	/NOWARNINGS	DIBOL	/W
		FORTRAN	--
COPY		PIP	--
	/ALLOCATE:size	--	[n]
	/ASCII	PIP,FILEX	/A
	/[BE]FORE[:date]	PIP	/[:date]
	/BINARY	PIP	/B
	/BOOT:[dev:]	DUP	/U:[dev:]
	/CONCATENATE	PIP	/U
	/DATE[:date]	PIP	/C[:date]
	/DELETE	PIP	/D
	/DEVICE	DUP	/I
	/DOS	FILEX	/S
	/ENDO:n	DUP	/E:n
	/EXCLUDE	PIP	/P
	/FILES	DUP	/F
	/IGNORE	PIP	/G

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/IMAGE	PIP,FILEX	/I
	/INFORMATION	PIP	/X
	/INTERCHANGE[:size]	FILEX	/U[:n]
	/LOG	PIP	/W
	/NOLOG	PIP	—
	/MULTIVOLUME	PIP	/V
	/NEWFILES	PIP	/C
	/OWNER[:num,num]	FILEX	[UIC]
	/PACKED	FILEX	/P
	/POSITION[:n]	PIP	/M[:n]
	/PRODELETE	PIP	/O
	/PROTECTION	PIP	/F
	/NOPROTECTION	PIP	/Z
	/QUERY	PIP,FILEX	/Q
	/NOQUERY	PIP	—
	/REPLACE	DUP	/R
	/NOREPLACE	PIP	/N
	/RETAIN	DUP	/R
	/SETDATE:date]	PIP	/T[:date]
	/SINCE[:date]	PIP	/D[:date]
	/SLOWLY	PIP	/S
	/START:n	DUP	/G
	/SYSTEM	PIP	/Y
	/TOPS	FILEX	/T
	/VERIFY	PIP,	/H
		DUP	/H
	/WAIT	DUP, FILEX	/W
		PIP	/E
CREATE		DUP	/C
	/ALLOCATE:size	DUP	[n]
	/EXTENSION:n	DUP	/E:size
	/START:n	DUP	/G:n
D			
DATE			
DEASSIGN			
DELETE		PIP	/D
	/BEFORE[:date]	PIP	/U[:date]
	/DATE[:date]	PIP	/C[:date]
	/LOG	FILEX	/S
	/ENTRY	QUEMAN	/M
	/EXCLUDE	PIP	/P
	/INFORMATION	PIP	/X
	/INTERCHANGE	FILEX	/U
	/LOG	PIP	/W
	/NEWFILES	PIP	/C
	/POSITION:n]	PIP	/M[:n]
	/QUERY	PIP	/Q
	/NOQUERY	PIP	—
	/SINCE[:date]	PIP	/R[:date]

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/SYSTEM	PIP	/Y
	/WAIT	PIP	/E
		FILEX	/W
DIBOL		RDIBOL	—
	/ALLOCATE:size	—	[n]
	/ALPHABETIZE	DIBOL	/A
	/BUFFERING	DIBOL	/B
	/CROSSREFERENCE	DIBOL	/C
	/LINENUMBERS	DIBOL	–
	/NOLINENUMBERS	DIBOL	/O
	/LIST[:filespec]	DIBOL	2nd output spec.
	/LOG	DIBOL	/G
	/OBJECT[:filespec]	DIBOL	1st output spec.
	/NOOBJECT	DIBOL	null 1st output spec.
	/ONDEBUG	DIBOL	/D
	/PAGEn	DIBOL	/P:n
	/TABLES	DIBOL	/S
	/WARNINGS	DIBOL	–
	/NOWARNINGS	DIBOL	/W
DIFFERENCES		RSRCCOM	—
	/ALLOCATE:size	—	[n]
	/ALWAYS	BINCOM	/O
	/AUDITTRAIL	SRCCOM	/A
	/BINARY	BINCOM	—
	/BLANKLINES	SRCCOM	/B
	/BYTES	BINCOM	/B
	/CHANGEBAR	SRCCOM	/D
	/COMMENTS	SRCCOM	—
	/NOCOMMENTS	SRCCOM	/C
	/DEVICE	BINCOM	/D
	/END[:n]	BINCOM	/E[:n]
	/FORMFEED	SRCCOM	/F
	/MATCH[:n]	SRCCOM	/L[:n]
	/OUTPUT:filespec	SRCCOM,	1st output spec.
		BINCOM	1st output spec.
	/PRINTER	SRCCOM	LP: as 1st output spec.

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
		BINCOM	LP: as 1st output spec
	/QUIET	BINCOM	/Q
	/SIP:filespec	BINCOM	2nd output spec
	/SLP:filespec	SRCCOM	2nd output spec.
	/SPACES	SRCCOM	--
	/NOSPACES	SRCCOM	/S
	/START[:n]	BINCOM	/S[:n]
	/TERMINAL	SRCCOM,	TT: as 1st output spec.
		BINCOM	TT: as 1st output spec.
	/TRIM	SRCCOM	--
	/NOTRIM	SRCCOM	/T
DIRECTORY		DIR	--
	/ALLDATE[:size]	---	[n]
	/ALPHABETIZE	DIR	/A
	/BACKUP	BUP	/L
	/BADBLOCKS	DUP	/K
	/BEFORE[:date]	DIR	/K[:date]
	/BEGIN	DIR	/G
	/BLOCKS	DIR	/B
	/BRIEF	DIR, FILEX	/P
	/COLUMNS:n	DIR	/Cn
	/DATE[:date]	DIR	/D[:Date]
	/DELETED	DIR	/Q
	/DOS	FILEX	/3
	/END:n	DUP	/En
	/EXCLUDE	DIR	/F
	/FAST	DIR, FILEX	/S
	/FILES	DUP	/F
	/FREE	DIR	/M
	/FULL	DIR	/E
	/INTERCHANGE	FILEX	/U
	/NEWFILES	DIR	/D
	/OCTAL	DIR	/O
	/ORDER[:category] /S:category]	DIR	
	/OUTPUT:filespec	DIR,	1st output file spec.

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
		FILEX	1st output file spec.
	/OWNER[:nnn,nnn]	FILEX	[U]C]
	/POSITION	DIR	/S
	/PRINTER	DIR,	LP: as 1st output spec.
		FILEX	LP: as 1st output file spec.
	/PROTECTION	DIR	/T
	/NOPROTECTION	DIR	/U
	/REVERSE	DIR	/R
	/SINCE[:date]	DIR	/B[:date]
	/SORT[:category]	DIR	
	/S[:category]		
	/SPARTn	DUP	/Gn
	/SUMMARY	DIR,	/N
	/TERMINAL	DIR,	TT: as 1st output spec.
		FILEX	TT: as 1st output spec.
	/TOPS	FILEX	/T
	/VOLUMEID[:ONLY]	DIR,	
		FILEX	/V[:ONLY]
	/WAIT	DUP,FILEX	/W
DISMOUNT		LD	/L
DUMP		R DUMP	—
	/ALLOCATE:size	—	—
	/ASCII	DUMP	—
	/NOASCII	DUMP	/N
	/BYTES	DUMP	/B
	/END:n	DUMP	/E:n
	/FOREIGN	DUMP	/T
	/IG:NONE	DUMP	/G
	/ONLY:n	DUMP	/O:n
	/OUTPUT:filespec	DUMP	1st output file spec.
	/PRINTER	DUMP	LP: as 1st output spec.

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/RAD50	DUMP	/X
	/START:n	DUMP	/S:n
	/TERMINAL	DUMP	TT: as 1st output spec.
	/WORDS	DUMP	/W
E		—	—
EDIT		EDIT,TECO,KED, EIS	
	KEX, K52	—	—
	/ALLOCATE:size	—	[n]
	/CREATE	EDITOR	—
	/EDIT	EDIT	—
	/EXECUTE:filespec	TECO	—
	/INSPECT	EDITOR	—
	/KED	KED	—
	/KEX	KEX	—
	/K52	K52	—
	/OUTPUT:filespec	EDITOR	—
	/TECO	TECO	—
EXECUTE		—	—
	/ALLOCATE:size	—	[n]
	/ALPHABETIZE	DIBOL	/A
	/BOTTOM:n	LINK	/B:n
	/BUFFERING	DIBOL	/B
	/CODE:type	FORTRAN	/I:type
	/CROSSREFERENCE[:type[,...:type]]	DIBOL,MACRO	/C
	/DEBUG[:filespec]	LINK	—
	/DIAGNOSE	FORTRAN	/B
	/DIBOL	DIBOL	—
	/DISABLE:type[,...:type]	MACRO	/D
	/DUPLICATE	LINK	/D
	/ENABLE:type[,...:type]	MACRO	/E
	/EXECUTE[:filespec]	LINK	1st output file spec.
	/EXTEND	FORTRAN	/E
	/FORTRAN	FORTRAN	—
	/GLOBAL	LINK	/N
	/HEADER	FORTRAN	/O
	/I4	FORTRAN	/T
	/LIBRARY	MACRO	/M
	/LINENUMBERS	DIBOL, FORTRAN	—
	/NOLINENUMBERS	DIBOL, FORTRAN	/O /S
	/LINKLIBRARY:filespec	LINK	—
	/LIST[:filespec]	—	2nd output file spec.

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/LOG	DIBOL	/G
	/MACRO	MACRO	—
	/MAP[:filespec]	LINK	2nd output file spec.
	/OBJECT[:filespec]	—	1st output file spec.
	/NODEBUG	DIBOL, FORTRAN	/D
	/PAGE:n	DIBOL	/P:n
	/PROMPT	LINK	/-
	/RECORD:length	FORTRAN	/R:length
	/RUN	RUN	—
	/NORUN	—	—
	/SHOW[:type]	FORTRAN, MACRO	/L[:value]
	/NOSHOW[:type]	MACRO	/N[:value]
	/STATISTICS	FORTRAN	/A
	/SWAP	FORTRAN	—
	/NOSWAP	FORTRAN	/D
	/TABLES	DIBOL	/S
	/UNITS:n	FORTRAN	/Nn
	/VECTORS	FORTRAN	—
	/NOVECTORS	FORTRAN	/V
	/WARNINGS	FORTRAN	/W
		DIBOL	—
	/NOWARNINGS	DIBOL	/W
		FORTRAN	—
	/WIDE	LINK	/W
FORMAT		—	—
	/PATTERN[:value]	FORMAT	/P[:value]
	/QUERY	FORMAT	—
	/NOQUERY	FORMAT	/Y
	/SINGLEDENSITY	FORMAT	/S
	/VERIFY[:ONLY]	FORMAT	/V[:ONLY]
	/WAIT	FORMAT	/W
FORTRAN		R FORTRAN	—
	/ALLOCATE:size	—	[n]
	/CODE:type	FORTRAN	/[type
	/DIAGNOSE	FORTRAN	/B
	/EXTEND	FORTRAN	/E
	/HEADER	FORTRAN	/O
	/I4	FORTRAN	/T
	/LINENUMBERS	FORTRAN	—
	/NOLINENUMBERS	FORTRAN	/S
	/LIST[:filespec]	FORTRAN	2nd output file spec.
	/OBJECT[:filespec]	FORTRAN	1st output file spec.

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/NOOBJECT	FORTRAN	null 1st output spec.
	/ONDEBUG	FORTRAN	/D
	/RECORD:length	FORTRAN	/R
	/SHOW[:type]	FORTRAN	/L:value:
	/STATISTICS	FORTRAN	/S
	/SWAP	FORTRAN	—
	/NOSWAP	FORTRAN	/U
	/UNITS:n	FORTRAN	/N:n
	/VECTORS	FORTRAN	—
	/NOVECTORS	FORTRAN	/V
	/WARNINGS	FORTRAN	/W
	/NOWARNINGS	FORTRAN	—
FRUN		—	—
	/BUFFER:n	—	—
	/NAME:name	—	—
	/PAUSE	—	—
	/TERMINAL:n	—	—
GET		—	—
GT OFF		—	—
GT ON		—	—
	/L:n	—	—
	/T:n	—	—
HELP		—	—
	/PRINTER	—	—
	/TERMINAL	—	—
INITIALIZE		DUP	/Z
	/BACKUP	BUP	/Z
	/BADBLOCKS[:RET]	DUP	/B
	/DOS	FILEX	/S
	/FILE:filespec	DUP	1st output spec.
	/INTERCHANGE	FILEX	/U
	/QUERY	DUP, FILEX	—
	/NOQUERY	BUP, DUP, FILEX	/Y
	/REPLACE[:RET]	DUP	/R
	/RESTORE	DUP	/D
	/SEGMENTS:n	DUP	/N:n
	/VOLUMEID:ONLY	DUP, FILEX	/V[:ONLY]
	/WAIT	DUP,FILEX	/W
INSTALL		—	—
LIBRARY		RLIBR	—
	/ALLOCATE size	—	[size]
	/CREATE	LIBR	—
	/DELETE	LIBR	/D

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/EXTRACT	LIBR	/E
	/INSERT	LIBR	—
	/LIST[:filespec]	LIBR	2nd output file spec.
	/MACRO[:n]	LIBR	/M[:n]
	/OBJECT[:filespec]	LIBR	1st output file spec.
	/NOOBJECT	LIBR	null 1st output spec.
	/PROMPT	LIBR	//
	/REMOVE	LIBR	/G
	/REPLACE	LIBR	/R
	/UPDATE	LIBR	/O
LINK		R LINK	—
	/ALLOCATE:size	—	/n
	/ALPHABETIZE	LINK	/A
	/BITMAP	LINK	—
	/NOBITMAP	—	/X
	/BOTTOM:value	LINK	/B:n
	/BOUNDARY:value	LINK	/Y:value
	/DEBUG[:filespec]	LINK	—
	/DUPLICATE	LINK	/D
	/EXECUTE[:filespec]	LINK	1st output file spec.
	/NOEXECUTE	LINK	null 1st output spec.
	/EXTEND:n	LINK	/En
	/FILL:n	LINK	/Zn
	/FOREGROUND[:stacksize] /R[:stacksize]	LINK	
	/GLOBAL	LINK	/N
	/INCLUDE	LINK	/I
	/LDA	LINK	/L
	/LIBRARY:filespec	LINK	—
	/LIMIT[:n]	LINK	/K[:n]
	/LINKLIBRARY:filespec	LINK	—
	/MAP[:filespec]	LINK	2nd output file spec.
	/PROMPT	LINK	//
	/ROUND:n	LINK	/U:n
	/RUN	LINK, RUN	—

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/SLOWLY	LINK	/S
	/STACK[:value]	LINK	/M[:n]
	/SYMBOLTABLE[:filespec]	LINK	Srd output file spec.
	/TOP[:value]	LINK	/H[:value]
	/TRANSFER[:value]	LINK	/T[:n]
	/WIDE	LINK	/W
	/XM	LINK	/V
LOAD		—	—
MACRO		MACRO	—
	/ALLOCATE:size	—	/n
	/CROSSREFERENCE[:type[...type]]	MACRO	/C
	/DISABLE:type[...type]	MACRO	/D
	/ENABLE:type[...type]	MACRO	/E
	/LIBRARY	MACRO	/M
	/LIST[:filespec]	MACRO	2nd output file spec.
	/OBJECT[:filespec]	MACRO	1st output file spec.
	/NOOBJECT	MACRO	null 1st output spec.
	/SHOW:type	MACRO	/i
	/NOSHOW:type	MACRO	/N
MOUNT	/D	/L	
	/WRITE	LD	/W
	/NOWRITE	LD	/R
PRINT		—	—
	/BEFORE[:date]	PIP, QUEMAN	/I[:date]
	/COPIES:n	PIP, QUEMAN	/K:n
	/DATE[:date]	PIP, QUEMAN	/C[:date]
	/DELETE	PIP, QUEMAN	/D
	/FLAGPAGE:n	QUEMAN	/H:n
	/[NO]FLAGPAGE	QUEMAN	/N
	/INFORMATION	PIP, QUEMAN	/X
	/LOG	PIP, QUEMAN	/W
	/NOLOG	PIP, QUEMAN	—
	/NAME:[dev:]jobname	QUEMAN	1st output file spec.

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/NEWFILES	PIP,QUEMAN	/C
	/PRINTER	PIP	/LP: as list output spec.
	/PROMPT	QUEMAN	/
	/QUERY	PIP,QUEMAN	/Q
	/SINCE[:date]	PIP, QUEMAN	/[:date]
	/WAIT	PIP	/E
PROTECT		PIP	/P
	/BEFORE[:date]	PIP	/J[:date]
	/DATE[:date]	PIP	/C[:date]
	/EXCLUDE	PIP	/O
	/INFORMATION	PIP	/X
	/LOG	PIP	/W
	/NOLOG	PIP	—
	/NEWFILES	PIP	/C
	/QUERY	PIP	/Q
	/SETDATE[:date]	PIP	/T[:date]
	/SINCE[:date]	PIP	/J[:date]
	/SYSTEM	PIP	/Y
	/WAIT	PIP	/E
R		—	
REENTER		—	—
REMOVE		—	—
RENAME		PIP	—
	/BEFORE[:date]	PIP	/J[:date]
	/DATE[:date]	PIP	/C[:date]
	/INFORMATION	PIP	/X
	/LOG	PIP	/W
	/NOLOG	PIP	—
	/NEWFILES	PIP	/C
	/PROTECTION	PIP	/P
	/NOPROTECTION	PIP	/Z
	/QUERY	PIP	/Q
	/REPLACE	PIP	—
	/NOREPLACE	PIP	/N
	/SETDATE[:date]	PIP	/T[:date]
	/SINCE[:date]	PIP	/J[:date]
	/SYSTEM	PIP	/Y
	/WAIT	PIP	/E
RESET		—	—
RESUME		—	—
RUN		—	
SAVE		—	—
SET		—	—

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
SHOW	ALL	RESORC	/A
	CONFIGURATION	RESORC	/C
	DEVICES[:dd]	RESORC	/D[:dd]
	ERRORS	ERROUT	–
	/ALL	ERROUT	/A
	/FILE:filespec	ERROUT	input file spec.
	/FROM[:date]	ERROUT	/F
	/OUTPUT:filespec	ERROUT	list output file spec.
	/PRINTER	ERROUT	LP: as list output spec.
	/SUMMARY	ERROUT	/S
	/TERMINAL	ERROUT	TT: as list output spec.
	/TO[:date]	ERROUT	/T
	JOBS	RESORC	/J
	MEMORY	RESORC	/A
	QUEUE	QUEMAN	/Q
	SUBSET	RESORC	/S
	TERMINALS	RESORC	/T
SQUEEZE		DUP	/S
	/OUTPUT:filespec	DUP	list output file spec.
	/QUERY	DUP	–
	/NOQUERY	DUP	/Y
	/WAIT	DUP	/W
SRUN			
	/BUFFER:n	–	–
	/LEVEL:n	–	–
	/NAME:logical jobname	–	–
	/PAUSE	–	–
	/TERMINAL:n	–	–
START		–	–
SUSPEND		–	–
TIME		–	–
TYPE		–	–
	/BEFORE[:date]	PIP	/L[:date]
	/COPIES:n	PIP	/K:n
	/DATE[:date]	PIP	/C[:date]

(Continued on next page)

Monitor Command	Option	System Utility Program	Option
	/DISMTC	PIP	/D
	/INFORMATION	PIP	/X
	/LOG	PIP	/W
	/NOLOG	PIP	—
	/NEWFILES	PIP	/C
	/QUERY	PIP	/Q
	/SINCE[:date]	PIP	/H[:date]
	/WAIT	PIP	/E
UNLOAD		—	—
UNPROTECT			
		PIP	/P
	/BEFORE[:date]	PIP	/J:date
	/DATE[:date]	PIP	/G[:date]
	/EXCLUDE	PIP	/F
	/INFORMATION	PIP	/X
	/LOG	PIP	/W
	/NOLOG	PIP	—
	/NEWFILES	PIP	/C
	/QUERY	PIP	/Q
	/SETDATE[:[E]date]	PIP	/T[:date]
	/SINCE[:[d]ate]	PIP	/H[:date]
	/SYSTEM	PIP	/Y
	/WAIT	PIP	/E

INDEX

ABORT keyboard command, 4–24
ABORT operating mode (IND), 5–40
  effect on CTRL/C, 5–41
Absolute addresses
  assigning, 4–146
Absolute Binary Loader
  creating files for, 4–150
Absolute binary loader
    and .LDA files, 3–2
Advance (A) command (EDT), 6–21
  arguments, 6–21
/ALLOCATE option
  COMPILE keyboard command, 4–36
  COPY keyboard command, 4–44
  CREATE keyboard command, 4–58
  DIBOL keyboard command, 4–68
  DIFFERENCES keyboard command,
      4–73
  DIRECTORY keyboard command,
      4–82
  DUMP keyboard command, 4–94
  EDIT keyboard command, 4–100
  EXECUTE keyboard command, 4–104
  FORTRAN keyboard command, 4–118
  LIBRARY keyboard command, 4–140
  LINK keyboard command, 4–147
  MACRO keyboard command, 4–157
/ALPHABETIZE option
  COMPILE keyboard command, 4–36
  DIBOL keyboard command, 4–68
  DIRECTORY keyboard command,
      4–82
  EXECUTE keyboard command, 4–105
  LINK keyboard command, 4–147
<ALPHAN>
  IND special symbol, 5–20
<ALTMODE>
  IND special symbol, 5–20
/ALWAYS option
  DIFFERENCES keyboard command,
      4–73
American Standard Code for Information
      Interchange
  See ASCII format
ANSI codes
  setting terminal for, 7–6
ANSWERBACK=message
  SETUP mode, 7–3
^ (append symbol)
  used with UCL, 4–18
ASCII files, 3–2
  copying with /ASCII, 4–44
ASCII format, 3–2
/ASCII option
  COPY keyboard command, 4–44
  DUMP keyboard command, 4–94
.ASK directive (IND), 5–25
  maximum number of characters
      allowed in prompt for, 5–2
  question display, 5–26
  responses to prompts printed by, 5–25
.ASKN directive (IND), 5–27
  maximum number of characters
      allowed in prompt for, 5–2

question display, 5–29
radix, 5–25
responses, 5–29
.ASKS directive (IND), 5–30
maximum number of characters allowed in prompt for, 5–2
question display, 5–32
responses, 5–32
Assembler
function of, 2–1
Assembly listing
creating a, 4–136
generating, 4–38
including symbol and label table in, 4–121
including symbol cross-reference listing in, 4–37
including symbol cross-reference section in, 4–105
Assembly listings
creating, 4–159
including a symbol cross-reference section in, 4–167
ASSIGN keyboard command, 4–25
* (asterisk)
used with DCL, 4–16
Audit trail
specifying for SLP, 4–74
/AUDITTRAIL option
DIFFERENCES keyboard command, 4–74

B keyboard command, 4–27
Background job
communicating with
See CTRL/B
Background jobs
executing, 4–185
running, 4–182
Backing up files or volumes
for storage
See BACKUP keyboard command
BACKSPACE key
with single-line editor, 4–14
BACKUP keyboard command, 4–22
backup process, 4–28
/DEVICE option, 4–29
initializing backup volumes for, 4–29
/RESTORE option, 4–29
using wildcards with, 4–28
valid input devices for, 4–28
valid output devices for, 4–28
/BACKUP option
DIRECTORY keyboard command, 4–82
INITIALIZE keyboard command, 4–132
Backup utility program
See BUP
Backup volumes (BUP)
initializing, 4–132
Bad block scans
performing, 4–133, 4–134
specifying last block for, 4–86
specifying starting block for, 4–90
Bad block table
preserving on output volume with /RETAIN, 4–58
Bad blocks
covering with .BAD files, 4–133
finding, 4–88
listing files that contain, 4–86
replacing, 4–134
treatment during a squeeze operation, 4–216
.BAD files
copying, 4–43
deleting, 4–63
renaming, 4–178
/BADBLOCKS option
DIRECTORY keyboard command, 4–83
INITIALIZE keyboard command, 4–133
Banner pages
preceding files with, 4–166
suppressing printing of, 4–167
Base-line monitor
See BL monitor
BATCH
description of, 1–9
BCD
SETUP mode, 7–11
/BEFORE option
COPY keyboard command, 4–44
DELETE keyboard command, 4–64
DIRECTORY keyboard command 4–83
PRINT keyboard command, 4–166
PROTECT keyboard command, 4–172
RENAME keyboard command, 4–179
TYPE keyboard command, 4–224
UNPROTECT keyboard command, 4–230
.BEGIN directive (IND), 5–33
/BEGIN option

DIRECTORY keyboard command, 4–81
Begin-end blocks (IND), 5–33, 6–46
  exiting, 5–47
Beginning (B) command (EDIT), 6–20
Binary coded decimal
  See BCD
Binary Comparison Program
  See BINCOM
Binary files
  comparing, 4–74
  comparing to create SIPP input file, 4–76
  copying with /BINARY, 4–44
Binary format, 3–2
Binary number
  format of (figure), 4–180
/BINARY option
  COPY keyboard command, 4–44
  DIFFERENCES keyboard command, 4–74
BINCOM
  description of, 1–6
Bit-map graphics
  setting terminal for, 7–8
/BITMAP option
  LINK keyboard command, 4–147
BL monitor
  advantages of, 1–3
  features of, 1–3
/BLANKLINES option
  DIFFERENCES keyboard command, 4–75
BLOCK
  SETUP mode, 7–8
Block-replaceable devices
  See Devices
/BLOCKS option
  DIRECTORY keyboard command, 4–84
BOLD
  SETUP mode, 7–7
BOOT keyboard command, 4–32
  /FOREIGN option, 4–30
  valid devices for, 4–32
  /WAIT option, 4–33
/BOOT option
  COPY keyboard command, 4–45
Bootable volume
  creating, 4–25
Bootstrap
  copying with /BOOT, 4–45
Bootstrapping a monitor file, 4–32
Bootstrapping the system, 3–1
  hardware bootstrap, 4–32
  software bootstrap, 4–32
  with a single-disk system, 4–33
/BOTTOM option
  EXECUTE keyboard command, 4–165
  LINK keyboard command, 4–148
/BOUNDARY option
  LINK keyboard command, 4–148
/BRIEF option
  DIRECTORY keyboard command, 4–84
/BUFFER option
  FRUN keyboard command, 4–124
  SRUN keyboard command, 4–212
/BUFFERING option
  COMPILE keyboard command, 4–36
  DIBOL keyboard command, 4–68
  EXECUTE keyboard command, 4–166
BUP
  description of, 1–6
/BYTES option
  DIFFERENCES keyboard command, 4–75
  DUMP keyboard command, 4–94

CAPS
  SETUP mode, 7–4
Card reader
  modifying card codes for, 4–190
  setting device conditions for, 4–190
Cassettes
  deleting files from, 4–66
CCL
  function of, 4–233
  syntax, 4–233
  using in control files, 5–1
CF
  LINK listing logical device, 4–150
/CHAIN directive (IND), 5–34
Change (C) command (EDIT), 6–29
  arguments, 6–29
/CHANGEBAR option
  DIFFERENCES keyboard command, 4–75
Character strings
  parsing in control files, 5–56
CLEAR
  SETUP mode, 7–7
CLICK
  SETUP mode, 7–4
Clock modes
  See SETUP
CLOCK SHOW
  SETUP mode, 7–12

/CLOSE directive (IND), 5–84
CLOSE keyboard command, 4–34
  closing a file opened with EDT, 6–9
/CODE option
  COMPILE keyboard command, 4–36
  EXECUTE keyboard command, 4–105
  FORTRAN keyboard command, 4–119
Color monitors
  changing color on, 7–4
COLOR or COLOR
  SETUP mode, 7–4
/132COLUMNS
  SETUP mode, 7–7
/80COLUMNS
  SETUP mode, 7–7
/COLUMNS option
  DIRECTORY keyboard command, 4–84
COMMAN
  IND local string symbol, 5–8
Command lines
  abbreviating commands and options, 4–5
  abbreviating using file specification factoring, 4–1
  continuing to the next line, 4–2
  deleting
    See Single-line editor, 4–13
  deleting characters in, 4–14
  editing
    See Single-line editor
  reproducing
    See Single-line editor
Commands
  multiline
    using in a control file, 5–4
Comments
  in control files, 5–5
    external, 5–5
    internal, 5–5
/COMMENTS option
  DIFFERENCES keyboard command, 4–75
Compilation listing
  creating (DIBOL), 4–69
  creating a, 4–105
  DIBOL
    including line numbers in, 4–69
    including symbol and label table in, 4–71
    including symbol and label tables in, 4–71
    including symbol cross-reference in, 4–69
    suppressing line numbers in, 4–69
  FORTRAN
    creating, 4–119
  generating, 4–36
  including symbol cross-reference listing in, 4–37
  including symbol cross-reference section in, 4–105
COMPILE keyboard command, 4–33
  /ALLOCATE option, 4–36
  /ALPHABETIZE option, 4–36
  /BUFFERING option, 4–36
  /CODE option, 4–36
  /CROSSREFERENCE option, 4–37
  default file types, 4–35
  /DIAGNOSE option, 4–37
  /DIBOL option, 4–37
  /DISABLE option, 4–37
  /ENABLE option, 4–37
  /EXTEND option, 4–37
  /FORTRAN option, 4–37
  /HEADER option, 4–37
  /I4 option, 4–37
  /LIBRARY, 4–37
  /LINENUMBERS option, 4–38
  /LIST option, 4–38
  /LOG option, 4–39
  /MACRO option, 4–39
  /NOLINENUMBERS option, 4–38
  /NOOBJECT option, 4–40
  /NOSHOW option
    and MACRO .LIST directive, 4–40
  /NOSWAP option, 4–41
  /NOVECTORS option, 4–41
  /NOWARNINGS option, 4–41
  /OBJECT option, 4–39
  /ONDEBUG option, 4–40
  /PAGE option, 4–40
  /RECORD option, 4–40
  /SHOW option, 4–40
  /STATISTICS option, 4–40
  /SWAP option, 4–41
  /TABLES option, 4–41
  /UNITS option, 4–41
  /VECTORS option, 4–41
  /WARNINGS option, 4–41
/CONCATENATE option
  COPY keyboard command, 4–46
Concise command language
  See CCL
Console
  assigning to another terminal, 4–203
  setting width for, 4–205
  with hardware tabs, 4–205

with simulated tab stops, 4–205
Console output
resuming, 4–204
suspending, 4–204
Control files, 5–2
See also IND, IND command lines
See Indirect control files
arithmetic operators in, 6–3
begin-end blocks in, 5–33, 5–46
exiting, 5–47
branching
to a command line, 5–48
to subroutines, 5–48
when errors occur, 5–54
CCL commands in, 5–4
chaining between, 5–34
closing open files, 5–34
command line in (example), 5–2
commenting
See Comments
contents of, 5–1
creating, 5–1
general rules for, 5–1
debugging with T, 5–8
default file type, 5–6
delaying processing of, 5–36
deleting after processing completes, 5–7
displaying processing of command lines in, 5–46
ESCAPE recognition in, 5–42
executing, 5–5
from keyboard monitor level, 5–6, 5–7
exiting, 5–47
formatting, 5–2
global symbol definition in, 5–43
global symbols in, 5–17
IND directives in, 5–8
keyboard commands in, 5–3, 5–4
suppressing display of, 5–8, 5–44
suppressing execution of, 5–8
labels in, 5–2
See Labels
local symbols in, 5–17
logical symbols in
See also logical symbols
lowercase characters in, 5–43
multiline commands in, 5–4
nesting, 5–9
numeric expressions in, 5–16, 5–18
numeric symbols in
See Numeric symbols
opening data files from, 5–55
parsing strings from within, 5–56
passing parameters when executing, 5–8
returning from subroutines within, 5–55
sample line from, 5–3
string symbols in
See String symbols
symbols in, 5–16
terminating processing of, 5–24, 5–63
testing device characteristics, 5–65
testing existence of a file, 5–67
testing the starting position of an ASCII string, 5–64
uses for, 5–1
CONTROL-Z operating mode (IND), 5–41
/COPIES option
PRINT keyboard command, 4–166
TYPE keyboard command, 4–224
COPY keyboard command
/ALLOCATE option, 4–44
/ASCII option, 4–44
assigning a date, 4–54
/BEFORE option, 4–44
/BINARY option, 4–44
/BOOT option, 4–45
/CONCATENATE option, 4–46
copying files in image mode, 4–48
/DATE option, 4–46
/DELETE option, 4–46
/DEVICE option, 4–46
/DOS option, 4–47
/END option, 4–47
/EXCLUDE option, 4–48
/FILES option, 4–48
function of, 4–42
/IGNORE option, 4–48
/IMAGE option, 4–48
/INFORMATION option, 4–49
/INTERCHANGE option, 4–49
/LOG option, 4–49
/MULTIVOLUME option, 4–50
/NEWFILES option, 4–50
/NOLOG option, 4–50
/NOPROTECTION option, 4–52
/NOQUERY option, 4–53
/NOREPLACE option, 4–53
/OWNER option, 4–51
/PACKED option, 4–51
/POSITION option, 4–51
/PREDELETE option, 4–52
/PROTECTION option, 4–52
/QUERY option, 4–53
/REPLACE option, 4–53
/RETAIN option, 4–53

/SETDATE option, 4–54
/SINCE option, 4–54
/SLOWLY option, 4–54
/START option, 4–54
syntax for specifying date, 4–45
/SYSTEM, 4–54
/TOPS option, 4–55
using wildcards with, 4–43
/VERIFY option, 4–55
/WAIT option, 4–55
Copy operations
changing volumes during, 4–56
verifying with /VERIFY, 4–55
CREATE keyboard command, 4–58
/ALLOCATE option, 4–58
/EXTENSION option, 4–59
/START option, 4–59
/CREATE option
EDIT keyboard command, 4–101
LIBRARY keyboard command, 4–140
CREFTMP
assigning a device to, 4–150
LINK temporary file, 4–150
/CROSSREFERENCE option
COMPILE keyboard command, 4–97
DIBOL keyboard command, 4–69
EXECUTE keyboard command, 4–105
MACRO keyboard command, 4–157
CSR addresses
changing in device handlers, 4–191, 4–197, 4–199
CTRL key, 3–7
CTRL/A, 3–7
CTRL/B, 3–7
treating as a program control character, 4–209
CTRL/C, 3–7
including in text with EDIT, 6–3
with EDIT, 6–2
CTRL/E, 3–7
CTRL/F, 3–8
treating as a program control character, 4–209
CTRL/O, 3–8
disabling, 4–183, 4–204
enabling, 4–204
with EDIT, 6–3
CTRL/Q, 3–8
CTRL/R, 3–8, 4–15
CTRL/S, 3–8
disabling, 4–204
enabling, 4–204
CTRL/U, 3–8, 4–14
with EDIT, 6–3
CTRL/W, 3–8, 4–15
CTRL/X, 3–8
treating as a program control character, 4–209
with EDIT, 6–3
CTRL/Z, 3–8
CURSOR
SETUP mode, 7–5

/D
END option, 5–7
D keyboard command, 4–60
DARK
SETUP terminal mode, 7–5
.DATA directive (IND), 5–34
creating an indirect command file with, 5–4, 5–35
Data format
ASCII, 3–2
binary, 3–2
DATA operating mode (IND), 5–41
Date
displaying the, 4–61
setting the, 4–61
DATE keyboard command, 4–61
/DATE option
COPY keyboard command, 4–46
DELETE keyboard command, 4–64
DIRECTORY keyboard command, 4–86
PRINT keyboard command, 4–166
PROTECT keyboard command, 4–172
RENAME keyboard command, 4–179
TYPE keyboard command, 4–224
UNPROTECT keyboard command, 4–230
<DATE>
IND special symbol, 5–22
DATE[dd:mm:yy]
SETUP mode, 7–12
Daylight saving time
See DST
DCL operating mode, 5–42
DEASSIGN keyboard command, 4–62
/DEBUG option
EXECUTE keyboard command, 4–105
LINK keyboard command, 4–148
Debugging a program, 4–148
Debugging programs
See ODT and VDT
.DEC directive (IND), 5–35
DECsystem–10 files
transferring to RT–11 format with /TOPS, 4–55

DECsystem-10 format files
obtaining directory listings of, 4–91
<DEFAULT>
IND special symbol, 5–20
and .ASK directive, 5–26
and .ASKN, 5–30
DEFAULT
SETUP control, 7–13
.DELAY directive (IND), 5–36
Delete (D) command (EDIT), 6–27
arguments (table), 6–27
DELETE key
for single-line editor, 4–13
with EDIT, 6–3
DELETE keyboard command, 4–63
/BEFORE option, 4–64
/DATE option, 4–64
/DOS option, 4–64
/ENTRY option, 4–64
/EXCLUDE option, 4–65
/INFORMATION option, 4–65
/INTERCHANGE option, 4–65
/LOG option, 4–65
/NEWFILES option, 4–65
/NOQUERY option, 4–66
/POSITION option, 4–66
/QUERY option, 4–66
/SINCE option, 4–67
/SYSTEM option, 4–67
/WAIT option, 4–67
DELETE operating mode (IND), 5–42
/DELETE option
COPY keyboard command, 4–46
LIBRARY keyboard command, 4–140
PRINT keyboard command, 4–166
TYPE keyboard command, 4–224
/DELETED option
DIRECTORY keyboard command, 4–85
Deleting characters on console, 4–205
DENSE
SETUP mode, 7–8
Device assignments
displaying, 4–207
Device attributes reply
setting like VT100, 7–6
setting to GENERIC100, 7–5
Device driver
See Device handler
Device handler
definition, 1–5
Device handlers
changing CSR addresses in, 4–191, 4–197, 4–199
changing error logging retry attempt number in, 4–191
changing vector addresses in, 4–192, 4–198
changing vectors in, 4–198, 4–200
loading into memory, 4–155
releasing if not loaded, 4–183
setting to log only unsuccessful I/O transfers, 4–192
setting to log successful I/O transfers and errors, 4–192
unloading from memory, 4–227
Device names
removing from the system tables, 4–177
/DEVICE option
BACKUP keyboard command, 4–29
COPY keyboard command, 4–46
DIFFERENCES keyboard command, 4–75
Device utility program
See DUP
Devices
assigning to a job, 4–156
binary comparison of
See Volumes
block-replaceable, 3–6
copying
See COPY keyboard command
file-structured, 3–6
installing into the system, 4–138
nonfile-structured, 3–6
random-access, 3–6
RT–11 directory-structured, 3–8
sequential-access, 3–6
structures of (table), 3–7
testing characteristics of, from control files, 5–45
testing in control files to see if loaded, 5–51
Devices handlers
displaying status of, 4–210
/DIAGNOSE option
COMPILE keyboard command, 4–37
EXECUTE keyboard command, 4–105
FORTRAN keyboard command, 4–119
DIBOL compiler
directing to use single buffering, 4–69
using single buffering, 4–105
DIBOL keyboard command, 4–68
/ALLOCATE option, 4–68
/ALPHABETIZE option, 4–68
/BUFFERING option, 4–68
/CROSSREFERENCE option, 4–69
/LINENUMBERS option, 4–69

/LIST option, 4–69
/LOG option, 4–70
/NOLINENUMBERS option, 4–69
/NOOBJECT option, 4–71
/NOWARNINGS option, 4–71
/OBJECT option, 4–70
/ONDEBUG option, 4–71
/PAGE option, 4–71
/TABLES option, 4–71
/WARNINGS option, 4–71
DIBOL keyboard monitor command
default file types for, 4–65
DIBOL object file
allocating space for, 4–68
/DIBOL option
COMPILE keyboard command, 4–87
EXECUTE keyboard command, 4–105
DIBOL program
including line numbers in, 4–38
DIBOL programs
compiling, 4–35
See DIBOL keyboard command
compiling with single buffering, 4–38
debugging, 4–71, 4–110
debugging with /ONDEBUG, 4–40
excluding line numbers from, 4–38
Differences file
excluding source program comments from, 4–75
including changebars in, 4–75
including source program comments in, 4–75
DIFFERENCES keyboard command, 4–72
/ALLOCATE option, 4–73
/ALWAYS option, 4–73
/AUDITTRAIL option, 4–74
/BINARY option, 4–74
/BLANKLINE option, 4–75
/BYTES option, 4–75
/CHANGEBAR option, 4–75
/COMMENTS option, 4–75
default file types for, 4–72
/DEVICE option, 4–75
/END option, 4–75
/FORMFEED option, 4–75
/MATCH option, 4–75
/NOCOMMENTS option, 4–75
/NOSPACES option, 4–76
/NOTRIM option, 4–76
/OUTPUT option, 4–76
/PRINTER option, 4–76
/QUIET option, 4–76
/SIP option, 4–76
/SLP option, 4–76
/SPACES option, 4–76
/START option, 4–77
/TERMINAL option, 4–77
/TRIM option, 4–79
using wildcards with, 4–72
Differences listing
creating a, 4–76
displaying on the console, 4–77
example, 4–77
including form feeds in, 4–75
interpretation of, 4–78
printing, 4–76
suppressing terminal display of, 4–76
DIR
description of, 1–7
Direct access labels (IND), 5–24
Directories
clearing, 4–132
DIRECTORY keyboard command, 4–80
/ALLOCATE option, 4–82
/ALPHABETIZE option, 4–82
/BACKUP option, 4–82
/BADBLOCKS option, 4–83
/BEFORE option, 4–83
/BEGIN option, 4–84
/BLOCKS option, 4–84
/BRIEF option, 4–84
/COLUMNS option, 4–84
/DATE option, 4–85
/DELETED option, 4–85
/DIRECTORY option, 4–90
/DOS option, 4–86
/END option, 4–86
/EXCLUDE option, 4–86
/FAST option, 4–86
/FILES option, 4–86
/FREE option, 4–86
/FULL option, 4–87
/INTERCHANGE option, 4–87
/NEWFILES option, 4–87
/NOPROTECTION option, 4–89
/OCTAL option, 4–87
/ORDER option, 4–88
/OUTPUT option, 4–89
/OWNER option, 4–89
/POSITION option, 4–89
/PRINTER option, 4–89
/PROTECTION option, 4–89
/REVERSE option, 4–90
/SINCE option, 4–90
/START option, 4–90
/SUMMARY option, 4–90
/TERMINAL option, 4–91

/TOPS option, 4–91
/VOLUMEID option, 4–91
/WAIT option, 4–91
Directory listing program
See DIR
Directory listings
abbreviated, 4–84, 4–86
displaying on the console, 4–91
for backup volumes created with BUP, 4–82
for magtapes, 4–89
how to read, 4–80
including file starting block numbers, 4–84
including files sizes in octal, 4–87
including volume ID and owner name, 4–91
obtaining on a single-disk system, 4–91
of DECsystem–10 format volumes, 4–91
of deleted files, 4–85
of DOS format volumes, 4–86
of interchange diskettes, 4–87
of protected files, 4–89
of RSTS/E format volumes, 4–86
of unprotected files, 4–89
of unused areas on volumes, 4–86
printing, 4–89
reverse sorting for, 4–90
sorting, 4–88, 4–90
by date, 4–88
by file name, 4–88
by file size, 4–88
by file type, 4–88
by position on volume, 4–88
sorting alphabetically, 4–82
specifying the number of columns in, 4–84
Directory segments
default number of (table), 4–136
number of
changing, 4–136
.DISABLE directive (IND), 5–36
.DISABLE OCTAL directive (IND)
effect on .ASKN directive, 5–28
/DISABLE option
COMPILE keyboard command, 4–37
EXECUTE keyboard command, 4–106
MACRO keyboard command, 4–159
DISMOUNT keyboard command, 4–92
/DOS option
COPY keyboard command, 4–47
DELETE keyboard command, 4–64
DIRECTORY keyboard command, 4–86
INITIALIZE keyboard command, 4–134
DOS–11 format files
deleting, 4–64
obtaining a directory of, 4–86
DOS–11 format volume
initializing, 4–134
Double-density diskettes
formatting for single-density, 4–118
DOWN
SETUP mode, 7–8
DRAFT
SETUP mode, 7–8
.DSABL directive arguments (MACRO), 4–158
DST
SETUP mode, 7–12
DU handler
changing CSR address in, 4–193
changing vector in, 4–193
defining disk partition size in, 4–193
defining ports in, 4–193
defining valid unit plug numbers in, 4–193
DUMP
description of, 1–7
.DUMP directive (IND), 5–37
DUMP keyboard command, 4–93
/ALLOCATE option, 4–94
/ASCII option, 4–94
/BYTES option, 4–94
/END option, 4–94
/FOREIGN option, 4–94
/IGNORE option, 4–94
/NOASCII option, 4–94
/ONLY option, 4–94
/OUTPUT option, 4–94
/PRINTER option, 4–94
/RAD50 option, 4–94
/START option, 4–94
/TERMINAL option, 4–94
/WORDS option, 4–96
Dump utility program
See DUMP
DUP
description of, 1–7
/DUPLICATE option
EXECUTE keyboard command, 4–106
LINK keyboard command, 4–148
DW handler
disable writing to, 4–194
not verifying output to, 4–194
verifying output to, 4–193
write-enabling, 4–194

E keyboard command, 4–98
EDIT, 6–1 to 6–41
  buffers, 6–11
  calling, 6–1
  character deletion, 6–3
  character-oriented commands, 6–7
  command arguments (table), 6–5
  command mode, 6–1
  command repetition, 6–9
  command strings, 6–6
  command syntax, 6–5
  commands, 6–12, 6–15
    advance by lines, 6–21
    change characters, 6–29
    change lines of text, 6–30
    delete characters, 6–27
    delete lines of text, 6–28
    display EDIT version number, 6–34
    effects on output files (table), 6–15
    effects on text buffer (table), 6–15
    enable uppercase or lowercase mode, 6–35
    execute command stored in Macro buffer, 6–34
    for closing files, 6–14
    for creating a backup file, 6–14
    for opening files, 6–12, 6–13
    for reading files, 6–12
    for writing files, 6–13
    Immediate mode, 6–39
    insert text, 6–26
    insert text saved in external buffer, 6–32
    list lines of text buffer, 6–24
    move location pointer a number of spaces, 6–20
    move location pointer to text buffer beginning, 6–20
    read files into text buffer, 6–15
    save text in external (macro) buffer, 6–33
    save text in external (save) buffer, 6–31
    search, 6–22
    search entire file for text string, 6–23
    search for text string and write buffer to output file, 6–24
    search text buffer for text string, 6–22
    terminate editing session, 6–19
    terminating, 6–2
    text listing, 6–24
    text modification, 6–26
    using arguments in, 6–5
    utility, 6–31
    verify current line, 6–26
    write text buffer to output file, 6–16, 6–18
  commands (table), 6–4
  current location pointer (cursor), 6–7
    determining the location of, 6–26
  deleting of all characters on current line, 6–3
  display editor, 6–36
    format, 6–36
    using with graphics terminals, 6–37
  error conditions, 6–40
  example, 6–39
  function of, 6–1
  ignoring current command string with, 6–3
  immediate mode for graphics terminals, 6–38
  key commands (table), 6–2
  line-oriented commands, 6–8
  memory usage, 6–11
  processing, 6–1
  terminating, 6–2
  text buffer, 6–11
    filling, 6–11
  text mode, 6–1
Edit Backup (EB) command (EDIT), 6–14
Edit Console (EC) command (EDIT), 6–37
Edit Display (ED) command (EDIT), 6–37
Edit File (EF) command (EDIT), 6–14
EDIT keyboard command, 4–99
  /ALLOCATE option, 4–100
  /CREATE option, 4–101
  /EDIT option, 4–101
  /EXECUTE option, 4–101
  /INSPECT option, 4–101
  /K52 option, 4–101
  /KED option, 4–101
  /OUTPUT option, 4–102
  /TECO option, 4–102
Edit Lower (EL) command (EDIT), 6–35
/EDIT option
  EDIT keyboard command, 4–101
Edit Read (ER) command (EDIT), 6–12
EDIT text editor
  calling, 4–101
  processing, 4–99
  setting as default editor, 4–104
Edit Upper (EU) command (EDIT), 6–35

Edit Version (EV) command (EDIT), 6–34
Edit Write (EW) command (EDIT), 6–13
Editors
  See Text editors
.ENABL directive arguments (MACRO), 4–158
.ENABLE DATA directive (IND)
  using to create an indirect command file, 5–4
.ENABLE directive (IND), 5–39
.ENABLE GLOBAL directive (IND)
  defining global symbols with, 5–17
.ENABLE OCTAL directive (IND)
  effect on .ASKN directive, 5–28
/ENABLE option
  COMPILE keyboard command, 4–37
  EXECUTE keyboard command, 4–106
  MACRO keyboard command, 4–158
.ENABLE SUBSTITUTION directive (IND), 5–22
.END directive (IND), 5–48
/END option
  COPY keyboard command, 4–47
  DIFFERENCES keyboard command, 4–75
  DIRECTORY keyboard command, 4–86
  DUMP keyboard command, 4–94
/ENTRY option
  DELETE keyboard command, 4–64
<EOF>
  IND special symbol, 5–20
.ERASE directive (IND), 5–46
Erasing a line
  See CTRL/U
Error Logger
  description of, 1–8
  displaying errors logged by, 4–211
  logging only unsuccessful I/O transfers, 4–192
  logging successful I/O transfers, 4–192
  SJ
    clearing internal buffer, 4–196
    disabling, 4–194
    enabling, 4–194
Error logging program
  See Error Logger
Error severity level to abort indirect command files
  changing, 4–195
Error severity level to abort keyboard commands, 4–195
<ERROR>
  IND special symbol, 5–20
Errors
  during copy operations
    overcoming with /IGNORE, 4–48
  during copy operations
    overcoming with /SLOWLY, 4–54
ESCAPE key
  with EDIT, 6–2
ESCAPE operating mode (IND), 5–42
<ESCAPE>
  IND special symbol, 5–20
    and .ASK directive, 5–27
    and .ASKN, 5–30
    and .ASKS, 5–33
Exchange (X) command (EDIT), 6–30
  arguments (table), 6–31
/EXCLUDE option
  COPY keyboard command, 4–48
  DELETE keyboard command, 4–65
  DIRECTORY keyboard command, 4–86
  PROTECT keyboard command, 4–172
  UNPROTECT keyboard command, 4–230
Executable files
  creating at link time, 4–149
  suppressing creation of at link time, 4–149
EXECUTE keyboard command, 4–103
  /ALLOCATE option, 4–104
  /ALPHABETIZE option, 4–105
  /BOTTOM option, 4–105
  /BUFFERING option, 4–105
  /CODE option, 4–105
  /CROSSREFERENCE option, 4–105
  /DEBUG option, 4–165
  default file types, 4–103
  /DIAGNOSE option, 4–105
  /DUAL option, 4–105
  /DISABLE option, 4–106
  /DUPLICATE option, 4–106
  /ENABLE option, 4–106
  /EXECUTE option, 4–106
  /EXTEND option, 4–107
  /FORTRAN option, 4–107
  /GLOBAL option, 4–107
  /HEADER option, 4–107
  /I4 option, 4–107
  /LIBRARY option, 4–107
  /LINENUMBERS option, 4–108
  /LINKLIBRARY option, 4–108
  /LIST option, 4–108
  /LOG option, 4–109
  /MACRO option, 4–109

/MAP option, 4–109
/NOLINENUMBER option, 4–108
/NORUN option, 4–111
/NOSHOW option, 4–111
/NOSWAP option, 4–111
/NOVECTORS option, 4–112
/NOWARNINGS option, 4–112
/OBJECT option, 4–109
/ONDEBUG option, 4–110
/PAGE option, 4–110
/PROMPT option, 4–110
/RECORD option, 4–111
/RUN option, 4–111
/SHOW option, 4–111
/STATISTICS option, 4–111
/SWAP option, 4–112
/TABLES option, 4–111
/UNITS option, 4–112
/VECTORS option, 4–112
/WARNINGS option, 4–112
/WIDE option, 4–112
Execute Macro (EM) command (EDT), 6–34
/EXECUTE option
EDIT keyboard command, 4–104
EXECUTE keyboard command, 4–106
LINK keyboard command, 4–149
Executing programs, 4–111
Executing sequential commands
See Indirect command files
Exit (EX) command (EDT), 6–19
.EXIT directive (IND), 5–47
Exit status
of control files, 5–20
<EXSTAT>
IND special symbol, 5–20
<EXSTR1>
IND special symbol, 5–22
/EXTEND option
COMPILE keyboard command, 4–37
EXECUTE keyboard command, 4–107
FORTRAN keyboard command, 4–119
LINK keyboard command, 4–149
Extended memory monitor
See XM monitor
/EXTENSION option
CREATE keyboard command, 4–56
/EXTRACT option
LIBRARY keyboard command, 4–141

<FALSE>
IND special symbol, 5–20
/FAST option
DIRECTORY keyboard command, 4–88
FB monitor
advantages of, 1–4
features of, 1–4
minimum requirements, 1–4
processing priorities, 1–4
File exchange utility program
See FILEX
File names, 3–4
/FILE option
INITIALIZE keyboard command, 4–134
File specifications
factoring, 4–4
restrictions, 4–4
syntax of, 4–3
File types, 3–4
default, 4–6
standard, 3–4
File-structured devices
See Devices
<FILERR>
IND special symbol, 5–22
Files
ASCII
See ASCII files
binary
See Binary format
See Binary files, 4–44
changing versions while deleting, 4–87
comparing, 4–72
binary, 4–74
by bytes, 4–76
excluding spaces and tabs, 4–76
including spaces and tabs, 4–76
concatenating
See /CONCATENATE option, 4–46
copying
See COPY command
creating, 4–58, 4–59
creating over a tentative file, 4–58
creating with a text editor, 4–101
deleted
recovering, 4–85
restoring (example), 4–59
deleting
using DELETE command, 4–68
deleting after copy, 4–46
deleting on output volume before copy, 4–52
dumping contents of, 4–93
dumping contents of (example), 4–96
extending, 4–58

naming, 3–4
object
See Object Files
object (MACRO)
creating, 4–160
suppressing creation of, 4–167
printing, 4–165
printing more that one copy of, 4–166
protecting during copy, 4–52
protecting from deletion, 4–171
removing protection from, 4–229
removing protection from on a single-disk system, 4–232
renaming, 4–178
showing status of, 4–207
tentative
making permanent, 4–34
/FILES option
COPY keyboard command, 4–48
DIRECTORY keyboard command, 4–86
Filespec
See File specifications
FILEX
description of, 1–7
/FILL option
LINK keyboard command, 4–149
<FILSPC>
IND special symbol, 5–32
Find (F) command (EDIT), 6–23
/FLAGPAGE option
PRINT keyboard command, 4–166
Foreground job
communicating with
Foreground jobs
aborting from system console, 4–24
assigning logical names to, 4–125
assigning terminals to interact with, 4–126
creating executable files for, 4–149
debugging, 4–126
FORTRAN
running, 4–124
running, 4–124
and reserving memory for, 4–124
suspending, 4–221
with assigned private console, 4–24
/FOREGROUND option
LINK keyboard command, 4–149
Foreground/background monitor
See FB monitor
/FOREIGN option
BOOT keyboard command, 4–33
DUMP keyboard command, 4–94
Form feeds
sending to the console, 4–204
sending to the line printer, 4–197, 4–199
FORMAT
description of, 1–7
FORMAT keyboard command, 4–113
devices supported by, 4–113
/NOQUERY option, 4–116
/PATTERN option, 4–114
/QUERY option, 4–116
/SINGLEDENSITY option, 4–116
/VERIFY option, 4–116
devices supported by, 4–113
/WAIT option, 4–117
Formatting a disk
reasons for, 4–113
Formatting a volume
on a single-disk system, 4–117
while the foreground job is loaded, 4–114
Formatting utility program
See FORMAT
/FORMFEED option
DIFFERENCES keyboard command, 4–75
FORTRAN
accessing multidimensional arrays using multiplication, 4–41
accessing multidimensional arrays using tables, 4–41
FORTRAN compiler
examining internal errors of, 4–105
FORTRAN keyboard command, 4–118
/ALLOCATE option, 4–118
/CODE option, 4–119
/DIAGNOSE option, 4–119
/EXTEND option, 4–119
/HEADER option, 4–119
/I4 option, 4–119
/LINENUMBERS option, 4–119
/LIST option, 4–119
/NOLINENUMBERS option, 4–119
/NOOBJECT option, 4–121
/NOSWAP option, 4–123
/NOVECTORS, 4–123
/NOWARNINGS option, 4–123
/OBJECT option, 4–121
/ONDEBUG option, 4–122
/RECORD option, 4–122
/SHOW option, 4–122
/STATISTICS option, 4–122
/SWAP option, 4–122
/UNITS option, 4–123

/VECTORS option, 4–123
/WARNINGS option, 4–123
FORTRAN listing codes (table), 4–122
FORTRAN logical units
overriding default number with /UNITS, 4–41
FORTRAN multidimensional arrays
accessing with multiplication, 4–123
accessing with multiplication operations, 4–112
accessing with tables, 4–112, 4–123
/FORTRAN option
COMPILE keyboard command, 4–37
EXECUTE keyboard command, 4–107
FORTRAN programs
changing to two-word default integer data type, 4–37
compiling, 4–35, 4–118
debugging with /ONDEBUG, 4–40
excluding line numbers from, 4–38
including line numbers in, 4–38
permitting USR to swap over, 4–41, 4–122
preventing USR from swapping over, 4–41, 4–111, 4–123
USR swapping over, 4–113
FORTRAN record length
overriding default with /RECORD, 4–40
/FREE option
DIRECTORY keyboard command, 4–86
FUN keyboard command, 4–124
/BUFFER option, 4–124
/NAME option, 4–125
/PAUSE option, 4–126
/TERMINAL option, 4–126
/FULL option
DIRECTORY keyboard command, 4–87

GENERIC100
SETUP mode, 7–5
Get (G) command (EDIT), 6–22
GET keyboard command, 4–127
GLOBAL operating mode (IND), 5–43
/GLOBAL option
EXECUTE keyboard command, 4–107
LINK keyboard command, 4–150
Global symbols
in control files, 5–17
including during link, 4–150
GOLD key
use of with Single-Line Editor, 4–10
GOSUB directive (IND), 5–48
GOTO directive (IND), 5–48
GRAPHIC
SETUP mode, 7–8
Graphics display paging
See CTRL/A
Graphics terminal display
with console terminal display
See CTRL/E
Graphics terminals
disabling, 4–128
display screen values (table), 4–129
enabling, 4–128
GT keyboard command, 4–128
/L option, 4–129
/T option, 4–129
GT OFF keyboard command, 4–128
GT ON keyboard command, 4–128
and EDIT, 6–37

Hardware
components (table), 1–2
Hardware characteristics program
See SETUP
Hardware configuration
displaying, 4–268
Hardware, minimum configuration, 1–1
/HEADER option
COMPILE keyboard command, 4–37
EXECUTE keyboard command, 4–107
FORTRAN keyboard command, 4–119
HELP
SETUP control, 7–13
Help key
Single-line editor, 4–10
HELP keyboard command, 4–130
/PRINTER option, 4–130
/TERMINAL option, 4–130
HORIZONTAL
SETUP mode, 7–9
12HOUR
SETUP mode, 7–11
24HOUR
SETUP mode, 7–11

I/O channels
open
purging, 4–183
I/O transfers
logging only unsuccessful, 4–192
logging successful, 4–192
/I4 option
COMPILE keyboard command, 4–37
EXECUTE keyboard command, 4–107

FORTRAN keyboard command, 4–119
IBM 3741-compatible diskettes
copying with /INTERCHANGE, 4–49
.IF directive (IND), 5–49
.IFDF directive (IND), 5–50
.IFDISABLED directive (IND), 5–50
.IFENABLED directive (IND), 5–50
.IFF directive (IND), 5–51
.IFLOA directive (IND), 5–51
.IFNDF directive (IND), 5–50
.IFNLOA directive (IND), 5–51
.IFT directive (IND), 5–51
/IGNORE option
COPY keyboard command, 4–48
DUMP keyboard command, 4–94
Image mode copying, 4–48
/IMAGE option
COPY keyboard command, 4–48
Immediate mode (EDIT), 6–38
commands, 6–39
.INC directive (IND), 5–53
/INCLUDE option
LINK keyboard command, 4–150
IND, 5–1 to 5–67
See also Control files
arithmetic operators (table), 5–16
characters with special meaning, 5–15
command string syntax, 5–6
logical operators (table), 5–16
logical tests, 5–49
operating modes
testing to see if enabled, 5–50
options, 5–7, 5–8
options (table), 5–7
parameter passing, 5–8
processing of command lines, 5–3
processing single command lines with, 5–7
relational operators (table), 5–16
running, 5–6
from the console (TT:), 5–7
symbol tables
displaying, 5–37
symbols, 5–16
global, 5–17
local, 5–17
terminating, 5–6
timeout count, 5–25, 5–27, 5–31, 5–46
IND command lines
See also Control files
CCL commands in, 5–4
comments in, 5–5
IND directives in, 5–3
keyboard commands in, 5–3, 5–4
labels in, 5–2
maximum number of characters allowed in, 5–2
sample, 5–3
IND command lines (example), 5–1
IND control files
See Indirect control files
IND directives, 5–23 to 5–67
functions of, 5–3
in control files, 5–3
separating from arguments, 5–3
IND directives (table), 5–11 to 5–13
IND operating modes (table), 5–14
IND symbols
deleting definitions of, 5–46
displaying definitions of, 5–38
enabling symbol substitution for, 5–45
logical
See Logical symbols
numeric
See Numeric symbols
special, 5–19
logical, 5–30
numeric, 5–20
string, 5–32
special (table), 5–20
string
See String symbols
substituting values for, 5–22
substituting values for (example), 5–32
testing to see if defined, 5–50
testing type of, 5–64
Indirect command files
accepting keyboard input, 4–17
changing error severity level to abort, 4–195
commenting, 4–19
compared to BATCH processing, 4–16
creating, 4–16
within a control file, 5–4
default file type, 4–16
definition of, 4–16
executing, 4–19
from within a control file, 5–10
executing a MACRO program in, 4–21
executing with SET KMON IND in effect, 4–19, 4–196
including CTRL/C in, 4–16
keyboard commands in, 4–16
nesting, 4–21
placing responses to prompts in, 4–17
running utility programs in, 4–16
specifying an overlays structure in, 4–18

start-up, 4–22
suppressing execution printout, 4–21
using commands that query in, 4–17
using INITIALIZE command in, 4–18
using LINK commands in, 4–17
Indirect Control File Processor
See IND
Indirect control files
See Control files
definition of, 4–16
echoing lines in, 4–204
executing with SET KMON IND in effect, 4–196
lengthy
partitioning, 4–17
setting severity of error that terminates execution, 4–21
suppressing echoing of lines in, 4–204
terminating, 4–24
using CTRL/Cs in, 4–17
Indirect files
See Indirect command files
creating an overlay structure in, 4–110
/INFORMATION option
COPY keyboard command, 4–49
DELETE keyboard command, 4–65
PRINT keyboard command, 4–157
PROTECT keyboard command, 4–172
RENAME keyboard command, 4–179
UNPROTECT keyboard command, 4–230
/INFORMATION option
TYPE keyboard command, 4–224
INITIALIZE keyboard command, 4–132
/BACKUP option, 4–132
/BADBLOCKS option, 4–133
/DOS option, 4–134
/FILE option, 4–134
/INTERCHANGE option, 4–134
/NOQUERY option, 4–134
/QUERY option, 4–134
/REPLACE option, 4–134
/RESTORE option, 4–136
/SEGMENTS option, 4–136
/VOLUMEID option, 4–136
/WAIT option, 4–137
Initializing a volume, 4–132
for use as a backup (BUP) volume, 4–132
on a single-disk system, 4–137
with protected files, 4–132
Input/output
See I/O
Insert (I) command (EDIT), 6–26
/INSERT option
LIBRARY keyboard command, 4–141
/INSPECT option
EDIT keyboard command, 4–101
INSTALL keyboard command, 4–138
Interchange diskettes
initializing, 4–134
obtaining directory listings of, 4–87
Interchange format files
deleting, 4–65
/INTERCHANGE option
COPY keyboard command, 4–49
DELETE keyboard command, 4–65
DIRECTORY keyboard command, 4–87
INITIALIZE keyboard command, 4–134
INTERLACE
SETUP mode, 7–5

Jobs
currently loaded
displaying status of, 4–212
displaying, 4–208
JUMP
SETUP mode, 7–6
Jump (J) command (EDIT), 6–20
arguments, 6–20

/K52 option
EDIT keyboard command, 4–100
K52 text editor
calling, 4–101
setting as default editor, 4–134
/KED option
EDIT keyboard command, 4–100
KED text editor
calling, 4–101
setting as default editor, 4–134
Keyboard click
enabling and disabling, 7–4
Keyboard command
E
See E keyboard command
SAVE
See SAVE keyboard command
Keyboard command options
mutually exclusive, 4–2
Keyboard commands
SQUEEZE
See SQUEEZE keyboard command
Keyboard commands
abbreviating, 4–5
ABORT

See ABORT, 4–24
ASSIGN
See ASSIGN
B
See B keyboard command
BACKUP
See BACKUP keyboard command
BOOT
See BOOT keyboard command
changing error severity level to abort, 4–236
CLOSE
See CLOSE keyboard command
COMPILE
See COMPILE keyboard command
COPY
See COPY keyboard command
CREATE
See CREATE keyboard command
D
See D Keyboard command
DATE
See DATE keyboard command
DEASSIGN
See DEASSIGN keyboard command
DELETE
See DELETE keyboard command
DIBOL
See DIBOL keyboard command
DIFFERENCES
See DIFFERENCES keyboard command
DIRECTORY
See DIRECTORY keyboard command
DISMOUNT
See DISMOUNT keyboard command
DUMP
See DUMP keyboard command
EDIT
See EDT keyboard command
error messages for, 4–23
EXECUTE
See EXECUTE keyboard command
FORMAT
See FORMAT keyboard command
FORTRAN
See FORTRAN keyboard command
FRUN
See FRUN keyboard command, 4–124
function of, 4–1
GET
See GET keyboard command
GT
See GT keyboard command
HELP
See HELP keyboard command
in control files, 5–3, 5–4
suppressing display of, 5–8
suppressing execution of, 5–8
INITIALIZE
See INITIALIZE keyboard command
INSTALL
See INSTALL keyboard command
LIBRARY
See LIBRARY keyboard command
LINK
See LINK keyboard command, 4–146
LOAD
See LOAD keyboard command, 4–155
MACRO
See MACRO keyboard command, 4–157
monitor restrictions, 4–22
MOUNT
See MOUNT keyboard command
PRINT
See PRINT keyboard command
prompts, 4–5
PROTECT
See PROTECT keyboard command
R
See R keyboard command
radix of arguments, 4–22
REENTER
See REENTER keyboard command
REMOVE
See REMOVE keyboard command
RENAME
See RENAME keyboard command
RESET
See RESET keyboard command
RESUME
See RESUME keyboard command, 4–184
RUN
See RUN keyboard command
SET
See SET keyboard command
SHOW
See SHOW keyboard command
SRUN
See SRUN keyboard command

START
See START keyboard command
suppressing control file display of, 6–44
SUSPEND
See SUSPEND keyboard command
syntax, 4–1
syntax illustration (sample), 4–3
syntax illustration conventions, 4–2
that should not be used in control files, 5–3
TIME
See TIME keyboard command
TYPE
See TYPE keyboard command
UNLOAD
See UNLOAD keyboard command
UNPROTECT
See UNPROTECT keyboard command
unrecognized, 4–22
Keyboard monitor
See KMON
Keyboard monitor commands
See Keyboard commands
Kill (K) command (EDIT), 6–26
arguments, 6–28
KMON
definition of, 1–3
KMON stack pointer
resetting, 4–183

T option
GT keyboard command, 4–129
Labels
in control files, 5–2
defining, 5–23
direct access, 6–24
processing of, 5–24
.label: directive (END), 5–23
LANGcode
SETUP mode, 7–9
Languages
supported by RT–11, 1–9
LD
description of, 1–7
LDA files
See Load image file
creating, 4–150
/LDA option
LINK keyboard command, 4–150
LETTER
SETUP mode, 7–9
/LEVEL option
SRUN keyboard command, 4–218
LIBR
description of, 1–7
Librarian
function of, 2–2
Librarian utility
See LIBR
Library files
accessing object modules in, 4–140
copying, 4–44
default system
See SYSLIB.OBJ
including during link, 4–151
macro
changing, 4–142
creating, 4–142
definition of, 4–139
object
creating, 4–140, 4–142
definition of, 4–139
deleting global symbols from the directory of, 4–143
deleting modules from, 4–140
extract modules from, 4–141
inserting modules into, 4–141
obtaining directory listings of, 4–142
replacing modules in, 4–144
suppressing creation of, 4–143
updating, 4–144
with duplicate module names, 4–141
specifying during MACRO assembly, 4–158
structure of, 4–139
system macro
See SYSMAC.SML
LIBRARY keyboard command, 4–139
/CREATE option, 4–140
/DELETE option, 4–140
/EXTRACT option, 4–141
/INSERT option, 4–141
/LIBRARY option, 4–140, 4–142
/MACRO option, 4–142
/NOOBJECT option, 4–143
/OBJECT option, 4–142
/PROMPT option, 4–143
prompting sequence (table), 4–144
/REMOVE option, 4–143
/REPLACE option, 4–144
specifying more than one line for, 4–143
/UPDATE option, 4–144
Library modules
duplicating, 4–106
duplicating in overlay segments, 4–148
/LIBRARY option

COMPILE keyboard command, 4–37
EXECUTE keyboard command, 4–107
LINK keyboard command, 4–151
MACRO keyboard command, 4–158
LIGHT
SETUP mode, 7–5
/LIMIT option
LINK keyboard command, 4–151
Line printer characteristics
See SETUP
Line printer handler
changing CSR addresses in, 4–197, 4–199
changing vectors in, 4–198, 4–200
Line printers
allowing corrective action while hung, 4–197, 4–199
generating an error when hung, 4–198, 4–199
passing nonprinting control characters to, 4–197, 4–199
sending carriage returns to, 4–196, 4–198
sending lowercase characters to, 4–198, 4–200
sending only uppercase characters to, 4–198, 4–200
sending spaces instead of TAB character to, 4–198, 4–200
sending TAB characters to, 4–198, 4–200
setting baud rate for, 4–200
setting bottom of page for, 4–198, 4–200
setting length of page for, 4–198, 4–200
setting width of output to, 4–198, 4–200
suppressing passing nonprinting control characters to, 4–197, 4–199
suppressing sending carriage returns to, 4–197, 4–199
/LINENUMBERS option
COMPILE keyboard command, 4–38
DIBOL keyboard command, 4–69
EXECUTE keyboard command, 4–108
FORTRAN keyboard command, 4–119
LINK
description of, 1–7
LINK keyboard command, 4–146
/ALLOCATE option, 4–147
/ALPHABETIZE option, 4–147
/BITMAP option, 4–147
/BOTTOM option, 4–148
/BOUNDARY option, 4–148
/DEBUG option, 4–148
default file types, 4–146
/DUPLICATE option, 4–148
entering on more than one line, 4–152
/EXECUTE option, 4–149
/EXTEND option, 4–149
/FILL option, 4–149
/FOREGROUND option, 4–149
/GLOBAL option, 4–150
/INCLUDE option, 4–150
/LDA option, 4–150
/LIBRARY option, 4–151
/LIMIT option, 4–151
/LINKLIBRARY option, 4–151
/MAP option, 4–151
/NOBITMAP option, 4–148
/NOEXECUTE option, 4–148
/PROMPT option, 4–152
prompting sequence, 4–147
/ROUND option, 4–152
/RUN option, 4–152
/SLOWLY option, 4–153
/STACK option, 4–153
/SYMBOLTABLE option, 4–153
/TOP option, 4–153
/TRANSFER option, 4–153
/WIDE option, 4–154
/XM option, 4–154
Link maps
See Load maps
Linker
function of, 2–2
Linker utility
See LINK
Linking a program
specifying the lowest address to use, 4–105
/LINKLIBRARY option
EXECUTE keyboard command, 4–108
LINK keyboard command, 4–151
List (L) command (EDT), 6–24
arguments, 6–25
.LIST directive (MACRO)
arguments (table), 4–161
specifying with MACRO command, 4–161
/LIST option
COMPILE keyboard command, 4–38
DIBOL keyboard command, 4–69
EXECUTE keyboard command, 4–108
FORTRAN keyboard command, 4–119
LIBRARY keyboard command, 4–142
MACRO keyboard command, 4–159
LISTING
SETUP mode, 7–9

Load image file, 3–2
LOAD keyboard command, 4–155
Load map
  global symbols
    listing in alphabetical order, 4–147
Load maps
  creating, 4–109, 4–151
  include global symbol cross-reference section in, 4–107
  including global symbol cross-reference section in, 4–150
  wide
    creating, 4–154
Load module
  initializing unused locations in, 4–149
  specifying lowest address to use in relocatable code in, 4–148
Load modules
  relocatable code in
    specifying highest address for, 4–153
Local symbols
  in control files, 5–17
LOCK key
  setting function of, 7–4
/LOG option
  COMPILE keyboard command, 4–39
  COPY keyboard command, 4–49
  DELETE keyboard command, 4–66
  DIBOL keyboard command, 4–70
  EXECUTE keyboard command, 4–106
  PRINT keyboard command, 4–168
  PROTECT keyboard command, 4–173
  RENAME keyboard command, 4–179
  TYPE keyboard command, 4–225
  UNPROTECT keyboard command, 4–231
Logical device names
  and device-independent programming, 4–25
  assigning, 4–25
    to logical disks, 4–163
  canceling, 4–62
  displaying assignments of, 4–208
  listing, 4–26
  syntax of, 4–25
Logical disk subsetting handler
  See LD
Logical disks
  assigning logical device names to, 4–163
  assigning to files, 4–163
  displaying assignments of, 4–208, 4–214
  freeing from current file assignment, 4–92
  verifying and correcting assignments, 4–168, 4–196
  write-enabling, 4–164, 4–196
  write-protecting, 4–164, 4–196
Logical symbols (IND)
  defining, 5–59, 5–62
  defining with .ASK directive, 5–25
  in control files, 5–17
  testing to see if true or false, 5–51
LOWERCASE operating mode (IND), 5–43

Macro (M) command (EDIT), 6–33
  arguments (table), 6–33
MACRO assembler
  calling, 4–109, 4–157
MACRO keyboard command, 4–157
  /ALLOCATE option, 4–157
  /CROSSREFERENCE option, 4–157
  /DISABLE option, 4–158
  /ENABLE option, 4–158
  /LIBRARY option, 4–158
  /LIST option, 4–159
  /NOOBJECT option, 4–161
  /NOSHOW option, 4–162
  /OBJECT option, 4–160
  /SHOW option, 4–161
Macro libraries
  definition of, 4–139
Macro library files
  changing, 4–142
  creating, 4–142
  identifying in a command line, 4–37, 4–107
/MACRO option
  COMPILE keyboard command, 4–39
  EXECUTE keyboard command, 4–109
  LIBRARY keyboard command, 4–142
MACRO programs
  assembling, 4–35, 4–39, 4–157
Magtape
  copying with /FILES, 4–48
  copying with /POSITION, 4–51
Magtapes
  bootable
    how to create, 4–134
  obtaining directory listings of, 4–89
  returning to 9-track default settings for, 4–200, 4–201
  setting 9-track density, 4–200
  setting density for, 4–201
  setting parity to even for, 4–201
  setting parity to odd for, 4–201
/MAP option

EXECUTE keyboard command, 4–105
LINK keyboard command, 4–151
<MAPPED>
IND special symbol, 5–20
MARGINBELL
SETUP mode, 7–5
/MATCH option
DIFFERENCES keyboard command, 4–75
MCR operating mode (IND), 5–44
MEMO
SETUP mode, 7–10
Memory
depositing values in with D keyboard command, 4–60
examining with E command, 4–98
Memory available on system
displaying, 4–208
Memory image files, 3–2
See .SAV files
Memory layout
displaying, 4–208, 4–212
Memory locations
writing contents to a file 4–197
Memory usage bitmap
creating, 4–147
suppressing creation of, 4–148
Minimum hardware configuration, 1–1
Monitor
definition, 1–3
<MONNAM>
IND special symbol, 5–22
MOUNT keyboard command, 4–163
/NOWRITE option, 4–164
/WRITE option, 4–164
/MULTIVOLUME option
COPY keyboard command, 4–50

/N
IND option, 5–8
/NAME option
FRUN keyboard command, 4–125
PRINT keyboard command, 4–168
SRUN keyboard command, 4–218
/NEWFILES option
COPY keyboard command, 4–50
DELETE keyboard command, 4–65
DIRECTORY keyboard command, 4–87
PRINT keyboard command, 4–168
PROTECT keyboard command, 4–173
RENAME keyboard command, 4–180
TYPE keyboard command, 4–225
UNPROTECT keyboard command, 4–231
NEWLINE
SETUP mode, 7–6
Next (N) command (EDIT), 6–18
.NLIST directive (MACRO)
arguments (table), 4–161
specifying with MACRO command, 4–162
/NOASCII option
DUMP keyboard command, 4–94
NOBCD
SETUP mode, 7–11
/NOBITMAP option
LINK keyboard command, 4–148
NOBOLD
SETUP mode, 7–7
NOCLICK
SETUP mode, 7–4
/NOCOMMENTS option
DIFFERENCES keyboard command, 4–75
NOCURSOR
SETUP mode, 7–5
NODST
SETUP mode, 7–12
/NOEXECUTE option
LINK keyboard command, 4–149
/NOFLAGPAGE option
PRINT keyboard command, 4–167
/NOLINENUMBERS option
COMPILE keyboard command, 4–38
DIBOL keyboard command, 4–69
EXECUTE keyboard command, 4–108
FORTRAN keyboard command, 4–119
/NOLOG option
COPY keyboard command, 4–60
PRINT keyboard command, 4–168
PROTECT keyboard command, 4–173
RENAME keyboard command, 4–180
TYPE keyboard command, 4–225
UNPROTECT keyboard command, 4–231
NOMARGINBELL
SETUP mode, 7–5
NONEWLINE
SETUP mode, 7–6
Nonfile-structured devices
See Devices
/NOOBJECT option
COMPILE keyboard command, 4–40
DIBOL keyboard command, 4–71
FORTRAN keyboard command, 4–121
LIBRARY keyboard command, 4–143

MACRO keyboard command, 4–161
/NOPROTECTION option
COPY keyboard command, 4–52
DIRECTORY keyboard command, 4–89
RENAME keyboard command, 4–180
/NOQUERY option
COPY keyboard command, 4–53
DELETE keyboard command, 4–66
FORMAT keyboard command, 4–116
INITIALIZE keyboard command, 4–164
SQUEEZE keyboard command, 4–217
NOREPEAT
SETUP mode, 7–6
/NOREPLACE option
COPY keyboard command, 4–53
RENAME keyboard command, 4–181
/NORUN option
EXECUTE keyboard command, 4–111
/NOSHOW option
COMPILE keyboard command, 4–40
EXECUTE keyboard command, 4–111
MACRO keyboard command, 4–162
/NOSPACES option
DIFFERENCES keyboard command, 4–78
/NOSWAP option
COMPILE keyboard command, 4–41
EXECUTE keyboard command, 4–111
FORTRAN keyboard command, 4–123
NOTABS
SETUP mode, 7–10
/NOTRIM option
DIFFERENCES keyboard command, 4–79
NOUNDERLINE
SETUP mode, 7–10
/NOVECTORS option
COMPILE keyboard command, 4–41
EXECUTE keyboard command, 4–112
FORTRAN keyboard command, 4–123
/NOWARNINGS option
COMPILE keyboard command, 4–41
DIBOL keyboard command, 4–71
EXECUTE keyboard command, 4–112
FORTRAN keyboard command, 4–123
NOWRAP
SETUP mode, 7–11
/NOWRITE option
MOUNT keyboard command, 4–164
Numeric expressions
evaluation of, in control files, 5–18
forming, in control files, 5–18
Numeric symbols (IND)
decrementing, 5–35
defining, 5–59, 5–60
in control files, 5–17
incrementing, 5–53
radix of, 5–17, 5–28
testing, 5–64

Object files, 3–2
creating, 4–109, 4–121
creating during DIBOL compilation, 4–70
creating with COMPILE keyboard command, 4–39
MACRO
creating, 4–160
suppressing creation of, 4–161
suppressing creation of, 4–40, 4–121
suppressing creation of during DIBOL compilation, 4–71
Object libraries
definition of, 4–139
Object library files
creating, 4–140, 4–142
deleting global symbols from the directory of, 4–143
deleting object modules from, 4–140
extracting object modules from, 4–141
inserting object modules into, 4–141
obtaining directory listings of, 4–142
replacing modules in, 4–144
suppressing creation of, 4–142
updating, 4–144
with duplicate module names, 4–141
Object module patch program
See PAT
/OBJECT option
COMPILE keyboard command, 4–39
DIBOL keyboard command, 4–70
EXECUTE keyboard command, 4–109
FORTRAN keyboard command, 4–121
LIBRARY keyboard command, 4–142
MACRO keyboard command, 4–160
OCTAL operating mode (IND), 5–44
/OCTAL option
DIRECTORY keyboard command, 4–87
<OCTAL>
IND special symbol, 5–20
ODT
description of, 1–9
linking with your program, 4–148
On-line debugging technique
See ODT

/ONDEBUG option
COMPILE keyboard command, 4–40
DIBOL keyboard command, 4–71
EXECUTE keyboard command, 4–110
FORTRAN keyboard command, 4–122
.ONERR directive (IND), 5–64
/ONLY option
DUMP keyboard command, 4–94
.OPEN directive (IND), 5–55
.OPENA directive (IND), 5–55
.OPENR directive (IND), 5–55
Operating modes (IND), 5–40
default settings, 5–40
disabling, 5–66
enabling, 5–39
global, 5–40
local, 5–40
testing to see if enabled, 5–60
/ORDER option
DIRECTORY keyboard command, 4–88
/OUTPUT option
SQUEEZE keyboard command, 4–216
/OUTPUT option
DIFFERENCES keyboard command, 4–76
DIRECTORY keyboard command, 4–89
DUMP keyboard command, 4–94
EDIT keyboard command, 4–102
Owner name
writing, 4–136
/OWNER option
COPY keyboard command, 4–51
DIRECTORY keyboard command, 4–89

P-sects
See Program sections
P1 through P9
IND local string symbols, 5–8
/PACKED option
COPY keyboard command, 4–51
/PAGE option
COMPILE keyboard command, 4–40
DIBOL keyboard command, 4–71
EXECUTE keyboard command, 4–110
PAGELENGTH:n
SETUP mode, 2–10
.PARSE directive (IND), 5–56
PAT
description of, 1–6
Patch programs
See PAT, SIPP, and SLP
/PATTERN option
FORMAT keyboard command, 4–114
/PAUSE option
FRUN keyboard command, 4–126
SRUN keyboard command, 4–218
Peripheral devices
specifying
See Physical device names
Peripheral interchange program
See PIP
Permanent devices names
See Physical device names
PF1 key
use of with single-line editor, 4–10
Physical device names
for peripheral devices, 3–3
Physical device names (table), 3–3
PIP
description of, 1–7
Position (P) command (EDT), 6–24
/POSITION option
COPY keyboard command, 4–51
DELETE keyboard command, 4–66
DIRECTORY keyboard command, 4–89
/PREDELETE option
COPY keyboard command, 4–52
PREFIX operating mode (IND), 5–44
PRINT keyboard command, 4–165
/BEFORE option, 4–166
/COPIES option, 4–166
/DATE option, 4–166
/DELETE option, 4–166
/FLAGPAGE option, 4–166
overriding SET SP FLAG=n, 4–167
restriction with QUEMAN/P option, 4–167
/INFORMATION option, 4–167
/LOG option, 4–168
/NAME option, 4–168
/NEWFILES option, 4–168
/NOFLAGPAGE option, 4–167
/NOLOG option, 4–168
/PROMPT option, 4–168
/QUERY option, 4–169
/SINCE option, 4–169
specifying on more than one command line, 4–168
/WAIT option, 4–169
when running QUEUE, 4–165
when running SPOOL, 4–165
Printer modes
See SETUP
/PRINTER option

DIFFERENCES keyboard command, 4–76
DIRECTORY keyboard command, 4–89
DUMP keyboard command, 4–94
HELP keyboard command, 4–130
Printing files
and specifying a job name, 4–168
and then deleting, 4–166
excluding banner pages when, 4–167
on the line printer, 4–165
with a single-disk system, 4–169
with banner pages, 4–166
Priority level
assigning for a system job, 4–218
Program sections
changing the size of during link, 4–152
extending at link time, 4–149
specifying a starting address boundary for, 4–148
Programs
debugging, 4–105
developing, 2–1
developing (figure), 2–3
examining, 4–111
running, 4–230
suppressing execution of, 4–111
/PROMPT option
EXECUTE keyboard command, 4–110
LIBRARY keyboard command, 4–143
LINK keyboard command, 4–152
PRINT keyboard command, 4–168
PROTECT keyboard command, 4–171
/BEFORE option, 4–172
/DATE option, 4–172
/EXCLUDE option, 4–172
/INFORMATION option, 4–172
/LOG option, 4–173
/NEWFILES option, 4–173
/NOLOG option, 4–173
/QUERY option, 4–173
/SETDATE option, 4–173
/SINCE option, 4–174
/SYSTEM option, 4–174
/WAIT option, 4–174
Protected files
deleting, 4–63
obtaining directory listings of, 4–89
Protected status
assigning to files, 4–52
removing from files, 4–52
Protecting files
while renaming, 4–180
Protecting files from deletion, 4–171
on a single-disk system, 4–174
/PROTECTION option
COPY keyboard command, 4–52
DIRECTORY keyboard command, 4–89
RENAME keyboard command, 4–180
Protection status
of a file
determining, 4–225
.PURGE directive (IND), 5–58

/Q
IND option, 5–8
/QUERY option
COPY keyboard command, 4–53
DELETE keyboard command, 4–66
FORMAT keyboard command, 4–116
INITIALIZE keyboard command, 4–134
PRINT keyboard command, 4–169
PROTECT keyboard command, 4–173
RENAME keyboard command, 4–180
SQUEEZE keyboard command, 4–217
TYPE keyboard command, 4–225
UNPROTECT keyboard command, 4–231
Queue
deleting a job from, 4–64
listing the contents of the, 4–213
Queue package
description of, 1–8
QUIET operating mode (IND), 5–44
/QUIET option
DIFFERENCES keyboard command, 4–76

R keyboard command, 4–175
/RAD50 option
DUMP keyboard command, 4–94
<RAD50>
IND special symbol, 5–20
Radix
setting octal default, in control files, 5–44
Random-access devices
See Devices
RD50/RD51 drives
disable writing to, 4–134
enable writing to, 4–134
verifying output to, 4–133
Read (R) command (EDT), 6–15
.READ directive (IND), 5–58
/RECORD option
COMPILE keyboard command, 4–40

EXECUTE keyboard command, 4–111
PORTRAN keyboard command, 4–122
REENTER keyboard command, 4–176
with EDIT, 6–5
.REL file
See Relocatable image file
Relocatable image file, 3–2
Relocation base
setting
See B keyboard command
REMOVE keyboard command, 4–177
/REMOVE option
LIBRARY keyboard command, 4–143
RENAME keyboard command, 4–178
/BEFORE option, 4–179
/DATE option, 4–179
/INFORMATION option, 4–179
/LOG option, 4–179
/NEWFILES option, 4–180
/NOLOG option, 4–180
/NOPROTECTION option, 4–180
/NOREPLACE option, 4–181
/PROTECTION option, 4–180
/QUERY option, 4–180
/REPLACE option, 4–181
/SETDATE option, 4–181
/SINCE option, 4–181
/SYSTEM option, 4–181
/WAIT option, 4–182
Renaming files
on a single-disk system, 4–182
REPEAT
SETUP mode, 7–8
/REPLACE option
COPY keyboard command, 4–53
INITIALIZE keyboard command, 4–134
LIBRARY keyboard command, 4–144
RENAME keyboard command, 4–181
RESET
SETUP control, 7–13
RESET keyboard command, 4–183
Resident Monitor
See RMON
RESORC
description of, 1–7
Resource program
See RESORC
/RESTORE option
BACKUP keyboard command, 4–29
INITIALIZE keyboard command, 4–136
RESUME keyboard command, 4–184
/RETAIN option
COPY keyboard command, 4–53
Retry attempts
changing number to be performed, 4–191
/RETURN directive (IND), 5–59
/REVERSE option
DIRECTORY keyboard command, 4–90
Ring buffers
resetting, 4–183
RMON
definition of, 1–3
/ROUND option
LINK keyboard command, 4–152
RSTS/E format files
deleting, 4–64
obtaining a directory of, 4–86
RT–11 directory structured devices
See Devices
RUN keyboard command, 4–185
/RUN option
EXECUTE keyboard command, 4–111
LINK keyboard command, 4–152
RX01 drives
write-enabling, 4–194
write-protecting, 4–194
RX02 diskettes
See Double-density diskettes
RX02 drives
write-enabling, 4–194
write-protecting, 4–194

.SAV files
See also Memory image file
executing, 4–175, 4–185
loading into memory, 4–127
running, 4–152
SAVE
SETUP control, 7–13
Save (S) command (EDIT), 6–31
Save image patch program
See SIPP
SAVE keyboard command, 4–187
Screen cursor
setting type of, 7–3
turning on and off, 7–5
Screen scrolling
JUMP or SMOOTH, 7–6
Search commands (EDIT), 6–22
/SEGMENTS option
INITIALIZE keyboard command, 4–136
Sequential access devices
See Devices

SET CR CODE = n command, 4–190
SET CR CRLF command, 4–190
SET CR HANG command, 4–190
SET CR IMAGE command, 4–190
SET CR NOCRLF command, 4–190
SET CR NOHANG command, 4–190
SET CR NOIMAGE command, 4–190
SET CR NOTRIM command, 4–190
SET CR TRIM command, 4–190
SET dd CSR2 = n command, 4–191
SET dd CSR = n command, 4–191
SET dd NOSUCCES command, 4–192
SET dd RETRY = n command, 4–191
SET dd SUCCES command, 4–192
SET dd VEC2 = n command, 4–193
SET dd VECTOR = n command, 4–192
SET DU CSR3 = n command, 4–193
SET DU CSR4 = n command, 4–193
SET DU VEC3 = n command, 4–193
SET DU VEC4 = n command, 4–193
SET DUn PARI = x command, 4–193
SET DUn PORT = x command, 4–193
SET DUn UNIT = x command, 4–193
SET DW NOWCHECK command, 4–194
SET DW NOWRITE command, 4–194
SET DW WCHECK command, 4–193
SET DW WRITE command, 4–194
SET DXn NOWRITE command, 4–194
SET DXn WRITE command, 4–194
SET DYn NOWRITE command, 4–194
SET DYn WRITE command, 4–194
SET EDIT EDIT command, 4–194
SET EDIT K52 command, 4–194
SET EDIT KED command, 4–194
SET EDIT keyboard command, 4–89
SET EDIT TECO command, 4–194
SET EL LOG command, 4–194
SET EL NOLOG command, 4–194
SET EL PURGE, 4–195
SET ERROR ERROR command, 4–195
SET ERROR FATAL command, 4–195
SET ERROR NONE command, 4–195
SET ERROR SEVERE command, 4–195
SET ERROR WARNING command, 4–196
SET EXIT NOSWAP command, 4–196
SET EXIT SWAP command, 4–196
SET keyboard command, 4–189
SET KMON IND command, 4–196
SET KMON NOIND command, 4–196
SET LD CLEAN command, 4–196
SET LDn NOWRITE command, 4–196
SET LDn WRITE command, 4–196
SET LP CR command, 4–196
SET LP CSR = n command, 4–197
SET LP CTRL command, 4–197
SET LP FORM command, 4–197
SET LP FORM0 command, 4–197
SET LP HANG command, 4–197
SET LP LC command, 4–198
SET LP LENGTH = n command, 4–198
SET LP NOCR, 4–197
SET LP NOCTRL command, 4–197
SET LP NOFORM command, 4–197
SET LP NOFORM0 command, 4–197
SET LP NOHANG command, 4–198
SET LP NOLC command, 4–198
SET LP NOTAB command, 4–198
SET LP SKIP = n command, 4–198
SET LP TAB command, 4–198
SET LP VECTOR = n command, 4–198
SET LP WIDTH = n command, 4–198
SET LS CR command, 4–198
SET LS CSR = n command, 4–199
SET LS CTRL command, 4–199
SET LS FORM command, 4–199
SET LS FORM0 command, 4–199
SET LS HANG command, 4–199
SET LS LC command, 4–200
SET LS LENGTH = n command, 4–200
SET LS NOCR, 4–199
SET LS NOCTRL command, 4–199
SET LS NOFORM command, 4–199
SET LS NOFORM0 command, 4–199
SET LS NOHANG command, 4–199
SET LS NOLC command, 4–200
SET LS NOTAB command, 4–200
SET LS SKIP = n command, 4–200
SET LS SPEED = n command, 4–200
SET LS TAB command, 4–200
SET LS VECTOR = n command, 4–200
SET LS WIDTH = n command, 4–200
SET MM DEFALT = 9 command, 4–200
SET MM DENSE command, 4–200
SET MM NOODDPAR command, 4–202
SET MM ODDPAR command, 4–201
SET MT DEFALT = 9, 4–201
SET MT DENSE command, 4–201
SET MT NOODDPAR command, 4–201
SET MT ODDPAR command, 4–201
SET options in effect
  displaying, 4–208
SET SL ASK command, 4–201
SET SL LEARN command, 4–201
SET SL NOLEARN command, 4–201
SET SL NOTTYIN command, 4–202
SET SL OFF command, 4–202
SET SL ON command, 4–201

SET SL SYSGEN command, 4–202
SET SL TTYIN command, 4–202
SET SL VTxx command, 4–202
SET SL WIDTH=n command, 4–202
SET SP FLAG=n
  overridden by PRINT/FLAGPAGE, 4–167
SET SP FLAG=s command, 4–203
SET SP FORM0 command, 4–203
SET SP KILL command, 4–203
SET SP NEXT command, 4–203
SET SP NOFORM0, 4–203
SET SP NOWAIT command, 4–203
SET SP NOWIDE command, 4–203
SET SP WAIT command, 4–203
SET SP WIDE command, 4–203
SET TERM
  See SET TT
SET TT CONSOL=n command, 4–203
SET TT CRLF command, 4–203
SET TT FB command, 4–203
SET TT FORM command, 4–204
SET TT HOLD command, 4–204
SET TT NOCRLF command, 4–203
SET TT NOFB command, 4–204
SET TT NOFORM command, 4–204
SET TT NOHOLD command, 4–204
SET TT NOPAGE command, 4–204
SET TT NOQUIET command, 4–204
SET TT NOSCOPE command, 4–205
SET TT NOTAB command, 4–205
SET TT PAGE command, 4–204
SET TT QUIET, 4–21
SET TT QUIET command, 4–204
SET TT SCOPE command, 4–205
SET TT TAB command, 4–205
SET TT WIDTH=n command, 4–205
SET USR NOSWAP command, 4–206
SET USR SWAP command, 4–205
SET VM BASE=nnnnnn command, 4–206
SET WILD EXPLICIT command, 4–206
SET WILD IMPLICIT, 4–206
SET XC SPEED=n command, 4–206
.SETD directive (IND), 5–59
/SETDATE option
  COPY keyboard command, 4–54
  PROTECT keyboard command, 4–173
  RENAME keyboard command, 4–181
  UNPROTECT keyboard command, 4–231
.SETF directive (IND), 5–62
.SETL directive (IND), 5–59
.SETN directive (IND), 5–60
.SETO directive (IND), 5–59
.SETS directive (IND), 5–61
.SETT directive (IND), 5–62
Setting hardware characteristics
  See SETUP
.SETTOP programmed request
  limiting amount of memory allocated by, 4–151
SETUP, 7–1 to 7–15
  clock modes, 7–11 to 7–12
    BCD, 7–11
    CLOCK SHOW, 7–12
    DATE[:ddmmm:yyj, 7–12
    DST, 7–12
    12HOUR, 7–11
    24HOUR, 7–11
    NOBCD, 7–11
    NODST, 7–12
    TIME[:hh:mm:ss], 7–12
  controls, 7–13
    DEFAULT, 7–13
    HELP, 7–13
    RESET, 7–13
    SAVE, 7–13
    SHOW, 7–13
  description of, 1–8
  device restrictions, 7–1
  indicated defaults, 7–2
  introduction to, 7–1
  restrictions for, 7–3
  summary of modes and controls, 7–14
  terminal and printer modes, 7–7 to 7–11
    BOLD, 7–7
    CLEAR, 7–7
    DENSE, 7–8
    DOWN, 7–8
    DRAFT, 7–8
    GRAPHIC, 7–8
    HORIZONTAL, 7–8
    LANG:code, 7–9
    LETTER, 7–9
    LISTING, 7–9
    MEMO, 7–10
    NOBOLD, 7–7
    NOTABS, 7–10
    NOUNDERLINE, 7–10
    NOWRAP, 7–11
    PAGELENGTH:n, 7–10
    TABS:n, 7–10
    TEXT, 7–8
    UNDERLINE, 7–10
    UP, 7–10
    VERTICAL, 7–11

WRAP, 7–11
terminal modes, 7–8 to 7–7
ANSWERBACK=message, 7–3
BLOCK, 7–3
CAPS, 7–4
CLICK, 7–4
COLOR on COLOR, 7–4
132COLUMNS, 7–7
80COLUMNS, 7–7
CURSOR, 7–5
DARK, 7–5
GENERIC100, 7–5
INTERLACE, 7–5
JUMP, 7–6
LIGHT, 7–5
MARGINBELL, 7–5
NEWLINE, 7–6
NOCLICK, 7–4
NOCURSOR, 7–5
NOMARGINBELL, 7–5
NONEWLINE, 7–6
NOREPEAT, 7–6
REPEAT, 7–6
SHIFT, 7–4
SMOOTH, 7–6
UNDERSCORE, 7–5
VT100, 7–6
VT52, 7–7
terminals
valid terminals for, 7–2
valid abbreviations for, 7–2
SETUP commands
description of, 7–1
valid destinations for, 7–1
SETUP control commands
example of syntax, 7–2
SETUP mode commands
command syntax, 7–1
example of syntax, 7–2
<SEVERE>
IND special symbol, 5–21
SHIFT
SETUP mode, 7–4
SHOW
SETUP control, 7–13
SHOW ALL keyboard command, 4–208
SHOW CONFIGURATION keyboard command, 4–208
example, 4–210
SHOW DEVICES[:dd] keyboard command, 4–210
SHOW ERRORS ALL keyboard command, 4–211
SHOW ERRORS FROM:date keyboard command, 4–211
SHOW ERRORS keyboard command, 4–211
SHOW ERRORS/FILE:filespec keyboard command, 4–211
SHOW ERRORS/OUTPUT:filespec keyboard command, 4–212
SHOW ERRORS/PRINTER keyboard command, 4–212
SHOW ERRORS/SUMMARY keyboard command, 4–212
SHOW ERRORS/TERMINAL keyboard command, 4–212
SHOW ERRORS/TO:date keyboard command, 4–211
SHOW JOBS keyboard command, 4–212
example, 4–212
SHOW keyboard command, 4–207
SHOW MEMORY keyboard command, 4–212
example, 4–213
/SHOW option
COMPILE keyboard command, 4–40
EXECUTE keyboard command, 4–111
FORTRAN keyboard command, 4–122
MACRO keyboard command, 4–161
SHOW QUEUE keyboard command, 4–213
example, 4–213
restriction for, 4–213
SHOW SUBSET keyboard command
example, 4–214
SHOW SUBSET keyboard command, 4–214
SHOW TERMINALS keyboard command, 4–214
example, 4–215
@ sign
restriction with DCL, 4–16
/SINCE option
COPY keyboard command, 4–54
DELETE keyboard command, 4–67
DIRECTORY keyboard command, 4–90
PRINT keyboard command, 4–169
PROTECT keyboard command, 4–174
RENAME keyboard command, 4–181
TYPE keyboard command, 4–225
UNPROTECT keyboard command, 4–233
Single-job monitor
See SJ monitor
Single-line editor

deleting characters with, 4–13, 4–14
deleting lines with, 4–13
executing a command line edited with, 4–14, 4–15
function keys (table), 4–11
function of, 4–9
GOLD key, 4–10
help key, 4–10
learning to use, 4–201
loading and enabling, 4–201
matching system generation characteristics for, 4–202
moving cursor with, 4–11
PF1 key, 4–10
redisplay line before last, 4–15
redisplaying current line with, 4–15
reproducing last command line with, 4–12
restoring deleted characters with, 4–13
restoring deleted command line, 4–13
switching characters with, 4–14
turning off, 4–10
turning on, 4–10
unloading and disabling, 4–202
/SINGLEDENSITY option
FORMAT keyboard command, 4–116
SIPP
description of, 1–9
/SIPP option
DIFFERENCES keyboard command, 4–76
SI monitor
advantages of, 1–3
features of, 1–3
SL
See Single-line editor
SLMEN.SYS
smaller SL handler, 4–10
/SLOWLY option
COPY keyboard command, 4–54
LINK keyboard command, 4–153
SLP
description of, 1–9
/SLP option
DIFFERENCES keyboard command, 4–76
SMOOTH
SETUP mode, 7–6
SOFT RESET
performing a, 7–7
/SORT option
DIRECTORY keyboard command, 4–90
Source file comparison program
See SRCCOM
Source files
comparing (example), 4–77
comparing to create SLP input file, 4–76
Source language patch program
See SLP
<SPACE>
UNO special symbol, 5–21
/SPACES option
DIFFERENCES keyboard command, 4–76
Special function keys, 3–6
SPOOL
132-column flag pages, 4–209
80-column flag pages, 4–209
description of, 1–8
form feed at block 0, 4–208
no form feed at block 0, 4–208
removing spooled files, 4–208
resume sending output, 4–208
setting number flag pages, 4–209
stop and delete/send next file, 4–208
stop then/en delete file, 4–208
SQUEEZE keyboard command, 4–216
/OUTPUT option, 4–216
SQUEEZE keyboard command
/NOQUERY option, 4–217
/QUERY option, 4–217
/WAIT option, 4–217
Squeeze operation
on a single-disk system, 4–217
SRCCOM
description of, 1–8
SRUN keyboard command, 4–218
/BUFFER option, 4–218
default file type, 4–218
/LEVEL option, 4–218
/NAME option, 4–218
/PAUSE option, 4–218
/TERMINAL option, 4–219
/STACK option
LINK keyboard command, 4–153
Stack pointer
modifying the address of, 4–153
Stack size
changing, 4–153
START keyboard command, 4–220
/START option
COPY keyboard command, 4–54
CREATE keyboard command, 4–59
DIFFERENCES keyboard command, 4–77

DIRECTORY keyboard command, 4–96
DUMP keyboard command, 4–94
Start-up indirect command files, 3–2
See Indirect command files
Start-up messages, 3–1
STARTF.COM, 3–2
Starting RT–11
See Bootstrapping the system, 3–1
STARTS.COM, 3–2
STARTX.COM, 3–2
/STATISTICS option
COMPILE keyboard command, 4–40
EXECUTE keyboard command, 4–111
FORTRAN keyboard command, 4–122
.STOP directive (IND), 5–63
String symbols (IND)
breaking into substrings, 5–19
concatenating, 5–19
defining, 5–19, 5–61
testing for alphanumeric or RAD50, 5–64
<STRLEN>
IND special symbol, 5–21
Subroutines
calling within a control file, 5–48
returning from, in control files, 5–69
SUBSTITUTION operating mode (IND), 5–45
<SUCCES>
IND special symbol, 5–21
SUFFIX mode (IND), 5–45
/SUMMARY option
DIRECTORY keyboard command, 4–96
SUSPEND keyboard command, 4–221
/SWAP option
COMPILE keyboard command, 4–41
EXECUTE keyboard command, 4–111
FORTRAN keyboard command, 4–122
Swapping program into memory, 4–196
preventing, 4–196
<SYDISK>
IND special symbol, 5–22
Symbol definitions file
creating, 4–153
Symbol substitution (IND)
enabling, 5–45
Symbol table overflow
preventing, 4–153
Symbol tables (IND)
deleting definitions from, 5–46
displaying contents of, 5–37
Symbols
in control files, 5–16, 5–17
/SYMBOLTABLE option
LINK keyboard command, 4–153
<SYMTYP>
IND special symbol, 5–21
.SYS files
copying, 4–43, 4–54
deleting, 4–63, 4–67
protecting from deletion, 4–174
renaming, 4–178, 4–181
SYFLIB.OBJ, 4–139
SYSMAC.SML, 4–139
System communication area
clearing locations in, 4–183
System device
squeezing, 4–216
System files
See .SYS files
removing protection from, 4–232
System generation options
displaying, 4–208
System jobs
assigning a logical job name to, 4–218
assigning priority levels for, 4–218
communicating with
See CTRL/X
debugging, 4–218
running, 4–212
suspending, 4–221
/SYSTEM option
COPY keyboard command, 4–54
DELETE keyboard command, 4–67
PROTECT keyboard command, 4–174
RENAME keyboard command, 4–181
UNPROTECT keyboard command, 4–232
System resources
displaying, 4–207
<SYSTEM>
IND special symbol, 5–21
<SYUNIT>
IND special symbol, 5–21

/T
IND option, 5–8
/T option
GT keyboard command, 4–129
TAB key
with EDIT, 6–3
/TABLES option
COMPILE keyboard command, 4–41
DIBOL keyboard command, 4–71
EXECUTE keyboard command, 4–111
TABSn

SETUP mode, 7–10
/TECO option
EDIT keyboard command, 4–102
TECO text editor
calling, 4–102
setting as default editor, 4–194
Terminal modes
See SETUP
/TERMINAL option
DIFFERENCES keyboard command, 4–77
DIRECTORY keyboard command, 4–91
DUMP keyboard command, 4–94
FRUN keyboard command, 4–126
HELP keyboard command, 4–130
SRUN keyboard command, 4–219
Terminal output
resuming
See CTRL/Q
suppressing
See CTRL/O
suspending
See CTRL/S
Terminals
displaying assignments of, 4–206
displaying files on, 4–229
displaying status of, 4–214
setting width of, 7–7
Terminating program execution
See CTRL/C
.TEST directive (IND), 5–64
.TESTDEVICE directive (IND), 5–65
.TESTFILE directive (IND), 5–67
TEXT
SETUP mode, 7–8
Text editors
calling, 4–99
for hard copy terminals, 1–6
for video terminals, 1–6
function of, 4–99
RT–11, 1–6
setting default, 4–99
types of, 4–99
Time
displaying, 4–222
setting, 4–222
TIME keyboard command, 4–222
<TIME>
IND special symbol, 5–22
TIME[:hh.mm:ss]
SETUP mode, 7–12
Timeout count
IND, 5–25, 5–27, 5–31, 5–45
TIMEOUT operating mode (IND), 5–45
<TIMOUT>
IND special symbol, 5–26
/TOP option
LINK keyboard command, 4–153
/TOPS option
COPY keyboard command, 4–55
DIRECTORY keyboard command, 4–91
TRACE operating mode (IND), 5–46
Transfer address
specifying, 4–153
/TRANSFER option
LINK keyboard command, 4–153
Transparent spooler package
See SPOOL
/TRIM option
DIFFERENCES keyboard command, 4–79
<TRUE>
IND special symbol, 5–20
TYPE keyboard command, 4–223
/BEFORE option, 4–224
/COPIES option, 4–224
/DATE option, 4–224
/DELETE option, 4–224
/INFORMATION option, 4–224
/LOG option, 4–225
/NEWFILES option, 4–225
/NOLOG option, 4–225
/QUERY option, 4–225
/SINCE option, 4–225
/WAIT option, 4–226
Type-ahead, 3–10
TYPEAHEAD operating mode (IND), 5–46

UCL
appending information to, 4–16
command abbreviations, 4–16
definition syntax, 4–15
deleting commands from, 4–16
description of, 4–15
displaying command definitions, 4–16
redefining commands, 4–16
UNDERLINE
SETUP mode, 7–10
UNDERSCORE
SETUP mode, 7–3
/UNITS option
COMPILE keyboard command, 4–41
EXECUTE keyboard command, 4–112
FORTRAN keyboard command, 4–128

UNLOAD keyboard command, 4–227
UNPROTECT keyboard command, 4–229
  /BEFORE option, 4–230
  /DATE option, 4–230
  /EXCLUDE option, 4–230
  /INFORMATION option, 4–230
  /LOG option, 4–231
  /NEWFILES option, 4–231
  /NOLOG option, 4–231
  /QUERY option, 4–231
  /SETDATE option, 4–231
  /SINCE option, 4–232
  /SYSTEM option, 4–232
  /WAIT option, 4–232
Unprotected files
  obtaining directory listings of, 4–89
Unprotecting files
  while renaming, 4–180
Unsave (U) command (EDIT), 6–32
  arguments (table), 6–33
UP
  SETUP mode, 7–10
/UPDATE option
  LIBRARY keyboard command, 4–144
User command linkage
  See UCL
User program memory area
  purging, 4–181
User Service Routine
  See USR
USR
  definition of, 1–3
  enabling swapping of, 4–205
  preventing from swapping over FORTRAN programs, 4–123
  preventing swapping of, 4–205
  swapping over FORTRAN programs, 4–121, 4–122
Utility programs
  list of, 1–6

Vectors
  changing in device handlers, 4–192, 4–193, 4–198, 4–200
/VECTORS option
  COMPILE keyboard command, 4–41
  EXECUTE keyboard command, 4–112
  FORTRAN keyboard command, 4–123
Verify (V) command (EDIT), 6–26
/VERIFY option
  COPY keyboard command, 4–55
  FORMAT keyboard command
    devices supported by, 4–113
Verifying a volume
  process of, 4–113
VERTICAL
  SETUP mode, 7–11
Virtual terminal communications package
  See VTCOM
.VOL directive (END), 5–67
Volume
  copying using SQUEEZE command, 4–216
Volume ID
  testing from within a control file, 5–67
Volume identification
  writing, 4–136
/VOLUMEID option
  DIRECTORY keyboard command, 4–91
  INITIALIZE keyboard command, 4–136
Volumes
  binary comparison of, 4–75
  copying
    See COPY keyboard command
  directory of
    clearing, 4–132
  initialized
    restoring an, 4–135
  listing unused areas on, 4–86
  verifying, 4–113
VT100
  SETUP mode, 7–6
VT52
  SETUP mode, 7–7
VT52 codes
  setting terminal for, 7–7
VTCOM
  description of, 1–8
  setting baud rate, 4–206

/WAIT option
  BOOT keyboard command, 4–33
  COPY keyboard command, 4–56
  DELETE keyboard command, 4–67
  DIRECTORY keyboard command, 4–91
  FORMAT keyboard command, 4–117
  INITIALIZE keyboard command, 4–137
  PRINT keyboard command, 4–169
  PROTECT keyboard command, 4–174
  RENAME keyboard command, 4–182
  SQUEEZE keyboard command, 4–217
  TYPE keyboard command, 4–226

UNPROTECT keyboard command, 4–232
<WARNING>
IND special symbol, 5–21
/WARNINGS option
COMPILE keyboard command, 4–41
DIBOL keyboard command, 4–71
EXECUTE keyboard command, 4–112
FORTRAN keyboard command, 4–123
/WIDE option
EXECUTE keyboard command, 4–112
LINK keyboard command, 4–164
% wildcard
See Wildcards, 4–7
* wildcard
See Wildcards, 4–6
Wildcards
commands that support (table), 4–7
default usage of, 4–8
embedded, 4–7
enabling use of implicit, 4–206
suppressing use of implicit, 4–206
use of, 4–6
using with DIFFERENCES keyboard command, 4–8
/WORDS option
DUMP keyboard command option, 4–95
WRAP
SETUP mode, 7–11
Write (W) command (EDT), 6–16
arguments (table), 6–17
/WRITE option
MOUNT keyboard command, 4–164
Write-enabling RX01 drives, 4–194
Write-enabling RX02 drives, 4–194
Write-protecting RX02 drives, 4–194
Write-protecting RX01 drives, 4–194

XM monitor
features of, 1–4
minimum requirements, 1–5
/XM option
LINK keyboard command, 4–164

HOW TO ORDER ADDITIONAL DOCUMENTATION

From	Call	Write
Chicago	312–640–5612 8:15 A.M. to 5:00 P.M. CT	Digital Equipment Corporation Accessories & Supplies Center 1050 East Remington Road Schaumburg, IL 60195
San Francisco	408–734–4915 8:15 A.M. to 5:00 P.M. PT	Digital Equipment Corporation Accessories & Supplies Center 632 Caribbean Drive Sunnyvale, CA 94086
Alaska, Hawaii	603–884–6660 8:30 A.M. to 5:00 P.M. ET or 408–734–4915 8:15 A.M. to 5:00 P.M. PT	
New Hampshire	603–884–6660 8:30 A.M. to 6:00 P.M. ET	Digital Equipment Corporation Accessories & Supplies Center P.O. Box CS2008 Nashua, NH 03061
Rest of U.S.A., Puerto Rico*	1–800–258–1710 8:30 A.M. to 9:00 P.M. ET	

*Prepaid orders from Puerto Rico must be placed with the local DIGITAL subsidiary (call 809–754–7575)

Canada		
British Columbia	1–800–267–6146 8:00 A.M. to 5:00 P.M. ET	Digital Equipment of Canada Ltd 940 Belfast Road Ottawa, Ontario K1G 4C2 Attn: A&SG Business Manager
Ottawa–Hull	613–234–7726 8:00 A.M. to 5:00 P.M. ET	
Elsewhere	112–800–267–6146 8:00 A.M. to 5:00 P.M. ET	
Elsewhere		Digital Equipment Corporation A&SG Business Manager*

*c/o DIGITAL's local subsidiary or approved distributor

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ___ Assembly language programmer
- ___ Higher-level language programmer
- ___ Occasional programmer (experienced)
- ___ User with little programming experience
- ___ Student programmer
- ___ Other (please specify) _____

Name _____ Date _____

Organization _____ Telephone _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear -- Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.32 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG/ML PUBLICATIONS, MLO5-5/E45
DIGITAL EQUIPMENT CORPORATION
146 MAIN STREET
MAYNARD, MA 01754-2571

Do Not Tear -- Fold Here

Cut Along Dotted Line