Volume 1

Micro/RSX User's Guide

Order No. AA-Y539B-TC

Micro/RSX Version 3.0

Digital Equipment Corporation Maynard, Massachusetts

First Printing, December 1983
Revised, June 1985

DEC	EduSystem	UNIBUS
DEC/CMS	IAS	VAX
DEC/MMS	MASSBUS	VAXcluster
DECnet	MicroPDP-11	VMS
DECsystem-10	Micro/RSX	VT
DECSYSTEM-20	PDP	
DECUS	PDT	
DECwriter	RSTS	digital
DIBOL	RSX	

ZK-2544

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

Contents Volume 1

Chapter 7 Running Tasks

Chapter 8 Preparing a User Batch Job

Chapter 9 The Indirect Command Processor

Figures

Tables

Preface

Manual Objectives

The *Micro/RSX User's Guide* provides information needed to do work on a Micro/RSX system. Important system concepts are introduced and explained, including how to use the DIGITAL Command Language (DCL) to communicate with the system. DCL is based on English words and is designed for ease in issuing commands to the system. The manual also gives instruction in using the EDT editor and many other useful system facilities.

Intended Audience

This manual is intended for any user of a Micro/RSX system. The manual is a reference manual with many tutorial elements, but new users should read the *Introduction to Micro/RSX* first.

Associated Manuals

If you are entirely new to computers, or to DCL, or to the RSX family of operating systems, you should read the *Introduction to Micro/RSX* and follow its instructions before proceeding to the *User's Guide.*

If you will be managing the Micro/RSX system, you should see the *Micro/RSX System Manager's Guide.*

Other manuals related to Micro/RSX are described in the book *Programming on Micro/RSX*. You should read this book even if you are not planning to do any programming, in order to learn more about how the system works and what other information is available.

Structure of This Manual

The *Micro/RSX User's Guide* is in two volumes. In general, Volume 1 should cover any questions that come up in your day-to-day use of the computer system. Volume 2 includes reference information on all parts of the operating system and is considerably more detailed. In many cases, Volume 2 duplicates information that appears in simpler form in Volume 1. The manual is organized around the major operating system functions.

Chapters 1 through 10 are in Volume 1 of the *Users's Guide*.

Chapter 1	Welcome to Micro/RSX, is a general introduction to Micro/RSX.
Chapter 2	How to Use DCL, introduces the basic concepts of DCL and includes many hints on using DCL.
Chapter 3	Using Your Terminal, covers terminal operations, including logging in and logging out, keyboard terminology and functions, setting and displaying important terminal attributes, broadcasting messages, and using the system HELP commands.
Chapter 4	EDT, documents EDT, the standard DIGITAL editor.
Chapter 5	Files on Micro/RSX Systems, introduces file handling operations, including commands that create and delete files, list directories, and rename, copy, type, and print files. Wildcards and defaults are explained.
Chapter 6	Devices and Volumes, covers common peripheral devices and introduces the system I/O (input/output) terminology. The chapter describes commands affecting devices and software volumes.
Chapter 7	Running Tasks, explains how tasks run in the system. In addition, the chapter describes commands to abort tasks and to display information about tasks in the system.
Chapter 8	Batch Processing, explains how to prepare and submit a batch job. With batch processing, you can schedule less urgent work, or work that uses lots of computer resources, for some later time when the system is not being heavily used. You need not be present to do batch processing.

Chapter 9	Indirect Command Processor, explains how to prepare and execute indirect command files. With Indirect, you can expedite the entry of a complex sequence and preserve it in an indirect command file; you can execute the sequence of commands any time you like.
Chapter 10	Quick Reference, is the alphabetical listing of all DCL commands and their formats, including comments to remind you of specific points about many commands and command elements. Each entry in Chapter 10 includes a cross-reference to the full command description elsewhere.

Chapters 11 through 16 are in Volume 2 of the *User's Guide.*

Chapter 11	More About Using Your Terminal, provides additional information on terminal operations, including logging in and logging out, keyboard terminology and functions, setting and displaying terminal attributes, broadcasting messages, using the system HELP commands, and writing a help file.
Chapter 12	More About Files on Micro/RSX Systems, goes into more detail about commands that list directories and that create, delete, rename, copy, type, and print files. Wildcards and defaults are explained.
Chapter 13	More About Devices and Volumes, discusses peripheral devices and explains the system I/O terminology. The chapter explains the relationship between hardware devices and the software file system, and describes how to prepare scratch disks and magnetic tapes for use on the system. Commands affecting devices and software volumes are also described.
Chapter 14	LINK and LIBRARY Commands, is for programmers. It documents the program development facilities included on the Micro/RSX Base Kit. The chapter briefly explains the RSX Task Builder and the LIBRARY command, which is used to maintain libraries.
Chapter 15	More About Running Tasks, explains how tasks run in the system, and how they are named, installed, fixed in memory, and controlled while running. In addition, the chapter describes commands that abort tasks, place tasks in the clock queue, and display information about tasks in the system.

Chapter 16	Common Error Messages, explains error messages common to several DCL commands. Other command-specific error messages are explained in the full command descriptions.

Conventions Used in This Manual

A number of conventions are used in the command descriptions in this manual:

Convention	Meaning
. . .	The vertical ellipsis shows where elements of command input have been omitted because they are irrelevant to the point being discussed.
[/qualifier]	Any command field enclosed in brackets is optional. If the brackets include syntactical elements, such as dots (.) or slashes (/), those elements are required for the field. If the field appears in lowercase, you are to substitute a legal command element if you include the field.
[g,m]	This signifies a User Identification Code (UIC). The g is a group number and the m is a member number. Where a UIC is required, only one set of brackets is shown, as [g,m]. Where the UIC is optional, two sets of brackets are shown, as [[g,m]]. The UIC identifies a user and is used mainly for controlling access to files and privileged system function.
[directory]	This signifies a directory, which is the location of files. Most directories have 1- to 9-character names, but some are in the same [g,m] form as the UIC. Where a directory name is required, only one set of brackets is shown, as [directory]. Where the directory is optional, two sets of brackets are shown, as [[directory]].
UPPERCASE	Any command field in uppercase indicates the legal form of the command. If you type it in that form, it will work as described. Most DCL commands also have abbreviations.

Convention	Meaning
lowercase	Any command field in lowercase is to be substituted for. Usually the lowercase word identifies the kind of substitution expected, such as filespec, which indicates that you should fill in a file specification.
/qualifier	Any command element preceded by a slash (/) is a DCL qualifier. Command qualifiers alter the action of a command they are attached to. Parameter qualifiers modify the action of the command as it affects that parameter.
parameter	Required command fields are generally called parameters. The most common parameters are file specifications. Parameters are preceded by blanks or DCL prompts.
:argument	Some parameters and qualifiers can be altered by the inclusion of arguments preceded by a colon. An argument can be either numerical (COPIES:3) or alphabetical (NAME:QIX). The equals sign (=) can be substituted for the colon to introduce arguments. COPIES=3 and COPIES:3 are the same.
filespec	A full file specification includes device, directory, file name, file type, and version number, as in the following example: `DB1:[COTTONTAL]HIPPITY.HOP;2` Full file specificationss are rarely needed. If you do not give a version number, the highest numbered version will be used. If you do not give a directory, the default directory will be used. Some system functions default to particular file types. See Chapter 5 for more information on file specifications. See also the individual command descriptions. Some commands accept a filespec with a DECnet node name. See Chapter 2 for more information.
red ink	All user input in examples is printed in red ink to distinguish it from system output. That is, what you type is shown in red.

Convention	Meaning
RET	A rectangular symbol with a 2- to 6-character abbreviation indicates that you are to press the corresponding key on your terminal. For example, RET indicates that you are to press the RETURN key, and DEL means that you are to press the DELETE key.
{A | B}	A number of options between braces and separated by vertical bars means that you are to choose one from the options listed.
CTRL/a	The rectangular symbol CTRL/a means that you are to press the key marked CTRL while pressing another key. Thus, CTRL/Z indicates that you are to press the CTRL key and the Z key together in this fashion. CTRL/Z is echoed on your terminal as ^Z. Not all control characters echo. See Chapter 3 for more information.
GOLD aaa	In Chapter 4, EDT, the word GOLD followed by other key names means press the keys in that order.

See Chapter 2 for more information on DCL conventions.

Summary of Technical Changes

New Commands and Qualifiers

The following table lists the new commands and qualifiers to DCL.

ASSIGN[/qualifier[s]]
  /GROUP:[g]
  /FINAL

New qualifiers.

COPY[/qualifier[s]]
  /ALLOCATION:n[.]

New qualifier.

DEASSIGN[/qualifier[s]]
  /GROUP[:g]

New qualifier.

DELETE/DIRECTORY

New command. Deletes a directory on a Files–11 volume and removes its name from the volume's Master File Directory (MFD).

```
INITIALIZE[/qualifier[s]]
        /BAD_BLOCKS:arg
                    NOAUTOMATIC
                    (AUTOMATIC,MANUAL)
                    (OVERRIDE,MANUAL)
        /DENSITY:arg
                HIGH
                LOW
```

New qualifiers.

```
INSTALL[/qualifier[s]]
      /[NO]RESIDENT_HEADER
      /[NO]WRITE_BACK
```

New qualifiers.

```
LINK[/qualifier[s]]
    /CODE:(arg[s])
          CLI
          FAST_MAP
    /[NO]PRINT
```

New qualifiers.

```
MOUNT[/qualifier[s]]
      /PROCESSOR:arg
                 UNIQUE
      /OVERRIDE:arg
                IDENTIFICATION
```

New arguments to existing qualifiers.

SET HOST

New command. Connects your terminal to another system.

SET PASSWORD

New command. Allows nonprivileged users to change their passwords.

```
SET PROTECTION[/qualifier]
     /[NO]DEFAULT
```

New qualifier.

```
SET TERMINAL[/qualifier[s]]
            /INQUIRE
            /DTC01
            /LA210
            /LN03
            /LQP02
            /LQP03
            /PRO_SERIES
            /[NO]HOSTSYNC
            /PRINTER_PORT
            /[NO]TTSYNC
```

New qualifiers.

```
SHOW ASSIGNMENTS[/qualifier[s]]
                 /GROUP[:g]
                 /TERMINAL:ttnn:
```

New qualifiers.

```
SHOW HOST
```

Displays the name of the processor to which your terminal is currently connected.

```
SHOW PROTECTION
```

New command. Displays your personal default file protection code.

```
SHOW TASKS[:taskname]/qualifier[s]
                      /DEVICE:ddnn:
```

New qualifier.

```
SHOW TERMINAL[:ttnn][/qualifier]
                    /[NO]DTC01
                    /[NO]LA210
                    /[NO]LN03
                    /[NO]LQP02
                    /[NO]LQP03
                    /[NO]PRO_SERIES
                    /[NO]HOSTSYNC
                    /PRINTER_PORT
                    /[NO]TTSYNC
```

New qualifiers.

New Device Support

The following table lists the new devices supported by Micro/RSX:

New terminal types:

DTC01	LA210
LN03	LQP02
LQP03	PRO_Series

New disk device:

RA60	RA80
RA81	RC25
RD52	RD53

New magnetic tape devices:

TK25	TK50

Chapter 1

Welcome to Micro/RSX

Micro/RSX is an *operating system*. The operating system is the fundamental *software* for a computer. Operating systems make it possible for all other programs to run on a computer.

The Micro/RSX Operating System is the latest member of the RSX family of operating systems. The MicroPDP–11 is a member of the PDP–11 family of computers. RSX operating systems run on all PDP–11 computers. VAX–11 RSX provides the RSX environment on VAX/VMS systems. This means that the same program can run under RSX on a wide variety of DIGITAL minicomputers and microcomputers, large and small. RSX systems allow several users and their programs, or *tasks* as they are called on RSX systems, to use the computer both independently and cooperatively.

Micro/RSX was designed to provide the capabilities of RSX systems on the MicroPDP–11 computer. In general, you'll find that Micro/RSX is very close to the RSX–11M–PLUS system in its behavior.

Micro/RSX is a *real-time* system. Real-time systems respond rapidly to input from users and to input from tasks.

Micro/RSX systems are *multiuser* systems. More than one user can have access to the system at any time without interfering with each other.

This combination, a real-time, multiuser system, allows the most rapid possible response for a variety of users. One important real-time feature is the ability to assign priority to different tasks. A task with a high priority can move a lower priority task out of the way temporarily in order to respond to the more urgent need.

All RSX systems feature *DCL*, the DIGITAL Command Language. DCL is based on English words and is designed for easy use. As you become more experienced, you can use DCL abbreviations to save yourself typing effort. DCL is used on many DIGITAL operating systems, including VAX/VMS. See Chapter 2 for more information on DCL.

1.1 What's It For?

You wouldn't have either a MicroPDP-11 or a Micro/RSX system unless you had some application for which you need a computer. Typically, commercial and industrial data-processing applications fall into three categories: real-time control, applications processing, and general purpose time-sharing. Micro/RSX can handle all three types.

1.1.1 The Real-Time Control Environment

In the real-time control environment, the principal function of the operating system is to handle rapid data movement with little human interaction.

Typical examples of such environments are steel rolling mills, oil refineries, and communications switching centers. When certain conditions are met—a thickness, a temperature, a delay—the system must respond rapidly—closing a valve, slowing a motor, throwing a switch.

The operating system, naturally, does not know about steel rails or long distance calls or gasoline. The principal function of the system in a real-time control environment is to receive, verify, reply to, and move data messages rapidly and without error.

1.1.2 The Applications Environment

In the applications environment, Micro/RSX is used for running tasks. These tasks can be office automation, communications systems, education programs, games, you name it. Sometimes these applications are available as tasks that you simply run with the DCL RUN command, but often terminal users have no direct contact with the operating system at all. All they see is the application. Terminal users communicate directly with the task rather than with the operating system. They may not even log in. Terminal users enter data, for processing by the applications task. The task itself opens and closes files, updating and altering information as it is entered.

In the applications environment, the greatest part of the system's resources is given over to continuous, high-volume data handling. Again, rapid, error-free handling of data messages is the principal function of the system, but instead of controlling a process, the messages update a data base under the control of an applications task.

1.1.3 The General Purpose Timesharing Environment

The general purpose time sharing environment is often a mixture of the other two environments with the addition of program development and testing as important activities. Terminal use is heavy in this kind of environment. There may be several users at one time, but usually they are thinking, looking up commands, or otherwise between keystrokes most of the time. Micro/RSX can get a lot accomplished between keystrokes.

In the general purpose time sharing environment, the system's interactive facilities are heavily used. These include DCL, EDT, and other tools. Many of the program development tools of the RSX family are part of the Micro/RSX Advanced Programmer's Kit, which is available separately. Also available separately are a number of programming languages and other products for programmers. Micro/RSX systems are often used to develop programs to be run on another system. These programs might be for real-time control or applications tasks. On the other hand, the programs may perform special computations, such as modeling, statistical analysis, or forecasting, that are intended to be run on the same system they were developed on.

Micro/RSX can be used in any combination of these three environments. If you want to understand your Micro/RSX system fully, you'll want to understand any applications that use the system in your installation.

1.1.4 The Purpose of the Operating System

The purpose of the operating system is to make the computer hardware easier to use.

The Executive controls the operating system. The Executive is a set of routines that coordinate all activities in the system, including supervision of input and output, allocation of resources, task execution, and communications.

The operating system manages the hardware and software resources of your system. This management requires that the operating system do four things:

1. **Keep track** of all resources
2. **Enforce a policy** on who gets what resources, when, and to what extent
3. **Allocate the resources** according to system policies
4. **Reclaim the resources** when they are no longer needed

There are four basic resources under the control of the operating system:

1. *Memory,* the system's workspace, where active tasks, active data, and the Executive itself are located
2. *Central Processing Unit,* or **CPU**, the part of the computer that executes instructions or computes
3. *Peripheral devices,* including terminals and disks, the line printer, if any, and any other hardware attached to the system
4. *Stored information,* the file system, the organization of files into directories and directories into volumes

Each task has different resource requirements. Involved scientific and statistical calculations, often called "number-crunchers," need a great deal of CPU time and memory, but make few demands on the system's devices or file system. Conversely, printing a long file can tie up a line printer for hours while using little memory and only a few seconds of CPU time.

In general, because there is an operating system, individual users don't need to concern themselves about such matters.

System managers need to be aware of what is happening on their systems and how to control it. See the *Micro/RSX System Manager's Guide* for more information on system management.

1.2 How the User's Guide is Organized

Because the Micro/RSX system can be used in so many different ways, it includes a wide choice of commands, many of which have only specialized uses. To help you tell the difference between commonly used commands and qualifiers and rarely used specialized commands, the *Micro/RSX User's Guide* is divided into two volumes. In general, Volume 1 describes the commonly used commands and procedures. The help files describe all commands, common and uncommon, and Chapter 10, Quick Reference, in Volume 1 also lists them all.

You will probably find that you use most of the common commands and a few of the uncommon ones. The wide choice of commands, although it may be confusing at first, gives you (and your system manager) a great deal of control over how the operating system does things. Often, you may not understand the purpose of a command or qualifier until you run into a situation where you need it.

Most users will be able to do most of their work successfully using Volume 1 only.

You should read over Chapters 1 through 6 in Volume 1 of the *Micro/RSX User's Guide* so that you will be familiar with the material if you ever need it. If you have questions about the material, you'll probably find the answers in the parallel, but more detailed, chapters in Volume 2. In any case, look at the tables of contents for the *Micro/RSX User's Guide* volumes and read anything that appeals to you.

If you are interested in programming, or if you are interested in what lies beneath the surface of Micro/RSX, you should read the separate book *Programming on Micro/RSX*. Among other things, this book describes the other Micro/RSX and RSX documentation that is available separately and as part of the Micro/RSX Advanced Programmer's Kit.

The Micro/RSX Advanced Programmer's Kit includes both documentation and some additional software.

Chapter 2

How to Use DCL

DCL is the DIGITAL Command Language. DCL gives users an extensive set of commands for dealing with the operating system. DCL commands provide for interactive program development, device and file manipulation, and interactive program execution and control.

If you are new to computers, or if you wish a quick review of DCL and the operating system, you should read the *Introduction to Micro/RSX*. This book includes a guided tour of the operating system, including an interactive terminal session that introduces all of the most commonly used DCL commands.

DCL commands are full words that describe the action to be taken, not abbreviations or mnemonics. Thus, if you wish to set a terminal to lowercase, you type SET TERMINAL/LOWERCASE. Because DCL commands and command elements are full words, they are self-documenting.

You are not required to use the full form of DCL commands at the terminal, however. Usually, you need type only the command elements required to form a unique command. For instance, the abbreviation SET TERM/LOW will also set a terminal to lowercase, but SET TERM/LO will not work because there is also a command SET TERMINAL/LOCAL. See Section 2.3 for more information on how DCL commands are parsed.

This chapter introduces the rules of DCL. The examples are intended to illustrate these rules, not to illustrate the full capabilities of the command. For more detail, see the individual command descriptions.

2.1 Micro/RSX and DCL: Basic Concepts

Commands in DCL are English or English-like words and follow well-defined syntax rules. Full commands are self-documenting. DCL is designed for consistency and ease of use.

DCL is the command language used on many DIGITAL operating systems. In particular, Micro/RSX DCL is designed for compatibility with future releases of RSX–11M–PLUS and VAX/VMS DCL.

2.2 DCL Command Descriptions

Command descriptions are found throughout both volumes of the *Micro/RSX User's Guides,* where chapters are organized according to the major functions of the operating system.

Command descriptions can have five parts:

1. *A brief statement of the function* of the command.
2. *A full description of the format* of the command, including prompts, defaults, acceptable values for arguments, and the effects of each qualifier.
3. *Examples* of the command in use. These examples often illustrate less obvious aspects of the use of the command.
4. *Notes,* including warnings about side effects, counteracting commands, recommendations for further reading, and so forth. Some command descriptions do not have notes.
5. *Error messages,* including an explanation and a suggested user action for each error. Only error messages specific to the command are included in the command descriptions. General error messages, such as those relating to syntax, are described in Chapter 16.

In addition, the formats of all Micro/RSX DCL commands are presented in the alphabetical listing in Chapter 10.

Nonprivileged commands are those commands needed by all users in everyday use of the system. Many privileged commands are also included in DCL. Privileged commands are those commands that affect system operations. For instance, the SHOW TIME command is a nonprivileged command, but the SET TIME command is privileged.

2.2.1 System Programmer Commands

Micro/RSX systems accept a few commands that are not in DCL format. They do not prompt. They are often not English words. In general, these commands are for the use of system programmers, not average users. You'll find these commands described in the *Micro/RSX System Manager's Guide* and in the Micro/RSX Advanced Programmer's Kit, which is available separately.

2.3 The DCL Command Line

A command consists of a command name (usually a verb) specifying the action the system is to take. Most commands also include one or more parameters and qualifiers to further define the action of the command. Qualifiers are preceded by a slash (/) and parameters are preceded by a space or prompt. Both qualifiers and parameters can take arguments. Arguments are preceded by a colon (:), or an equals sign (=). This book documents only the colon.

Here is an example:

```
$ DIRECTORY/OUTPUT:UNCLE.DAT SOLO.* RET
```

In this case, the elements of the command line are as follows:

- DIRECTORY is the command name.
- OUTPUT, which is preceded by a slash (/), is a qualifier. This qualifier indicates that you want the output from the DIRECTORY command to go somewhere other than your terminal screen, which is the default.
- UNCLE.DAT, which is preceded by a colon (:), is the argument to the /OUTPUT qualifier. This argument is the name of the file where you want the output from the command to go.
- SOLO.*, which is preceded by a blank, is a parameter. This parameter is the name of the file(s) of which you are taking a directory listing. Most DCL parameters are file specifications, as is this one. The asterisk (*) indicates a wildcard file type. See Section 5.4 for information on wildcards.

Note that after you type in a command, you must pass it to the operating system by pressing the RETURN key. See Section 2.3.11 for a discussion of command lines that are too long to fit on one line of your terminal.

Some commands require parameters or arguments as part of the command line. If you fail to supply a required command element, DCL prompts you with one or two words indicating the general nature of the required element. If you do not understand the prompt, type a question mark (?) for help. (In some cases, an omission causes an error message rather than a prompt.)

2.3.1 Prompting

The prompts teach you the form of a command by requesting that you supply required command elements.

For example, the RENAME command works as follows:

```
$ RENAME [RET]
Old file name?  BROWNS.STL [RET]
New file name?  ORIOLES.BLT [RET]
```

The one-line format for RENAME is as follows:

```
$ RENAME BROWNS.STL ORIOLES.BLT [RET]
```

The formats can be mixed. DCL prompts for whatever you leave out. For example:

```
$ RENAME BROWNS.STL [RET]
New file name? ORIOLES.BLT [RET]
```

There are no defaults for prompts. You must supply a response to any prompt. If you do not wish to continue with the command, press CTRL/Z.

2.3.2 Qualifiers

Qualifiers modify the action of the command and are generally optional. Qualifiers always start with a slash (/).

Qualifiers are either command qualifiers or parameter qualifiers. Most qualifiers are command qualifiers. In this manual, command qualifiers are always shown as modifying the command name, as in this example:

```
$ TYPE/TODAY *.HLP [RET]
```

However, most command qualifiers can appear anywhere in the command line. They are also called "floating qualifiers." The following examples illustrate how command qualifiers can float:

```
$ TYPE *.DAT/TODAY [RET]
```

or

```
$ TYPE [RET]
File(s)? *.DAT/TODAY [RET]
```

or

```
$ TYPE [RET]
File(s)? /TODAY [RET]
File(s)? *.DAT [RET]
```

You can mix formats and get exactly the same results, as in the following examples:

```
$ TYPE/TODAY [RET]
File(s)? *.DAT/EXCLUDE:ILLYA.DAT;* [RET]
```

or

```
$ TYPE/TODAY [RET]
File(s)? *.DAT/ [RET]
Qualifier? EXCLUDE:ILLYA.DAT;* [RET]
```

or

```
$ TYPE/TODAY [RET]
File(s)? / [RET]
Qualifier? EXCLUDE:ILLYA.DAT;* [RET]
File(s)? *.DAT [RET]
```

Note that you are prompted for a qualifier when a slash with no qualifier attached appears on the command line. When you supply the qualifier, do not type the slash again.

Almost all command qualifiers can float. In a few cases, however, the command qualifier must appear directly after the command it modifies. Whenever two similar commands are described separately, such as ASSIGN and ASSIGN/REDIRECT or CREATE and CREATE/DIRECTORY, the distinguishing qualifier cannot float. Other qualifiers to such commands can still float, however, as in this example:

```
$ CREATE/DIRECTORY [RET]
Device and UFD? [JENA]/ALLOCATION:5 [RET]
```

which is the same as the following example:

```
$ CREATE/DIRECTORY/ALLOCATION:5 [JENA] [RET]
```

Parameter qualifiers, sometimes called file specification qualifiers, cannot float. Usually a parameter qualifier must be attached to a file specification, because most DCL parameters are file specifications.

Many qualifiers can be negated by prefixing NO or—(minus) to the qualifier name. Thus, the command

```
$ DELETE/LOG *.TXT;* RET
```

deletes all files with the type .TXT and prints a list of the deletions on your terminal, while the command

```
$ DELETE/NOLOG *.TXT;* RET
```

or

```
$ DELETE/-LOG *.TXT;* RET
```

simply deletes all the files without printing a list. (This action is in fact the default behavior, but the example is given to illustrate the use of the negative form of qualifiers.)

2.3.3 HELP

HELP is available from the system for all DCL commands and also for many other aspects of the system. You can get help through the HELP command or by typing a question mark (?) in response to any DCL prompt.

For instance, if you need help on the TYPE command, type the following:

```
$ HELP TYPE RET
 TYPE[/qualifier[s]] filespec[s]
        /DATE:dd-mmm-yy
        /SINCE:dd-mmm-yy
        /THROUGH:dd-mmm-yy
        /SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
        /TODAY
        /EXCLUDE:filespec
The TYPE command displays the contents of text files on your
terminal.
To obtain help on the above qualifiers, type the following:
    HELP TYPE qualifier
$
```

The HELP text consists of the command syntax, showing that TYPE accepts one or more file specifications and one or more qualifiers, followed by a brief explanation.

You can also type a HELP command naming the qualifier to get more information on that qualifier, as in this example:

```
$ HELP TYPE TODAY [RET]

 TYPE/TODAY filespec[s]

 The /TODAY qualifier specifies that you wish the TYPE
 command to type only files created today.
```

If you want help while being prompted by the TYPE command, use the following procedure:

```
$ TYPE [RET]
File(s)? ? [RET]

 TYPE[/qualifier[s]] filespec[s]
        /DATE:dd-mmm-yy
        /SINCE:dd-mmm-yy
        /THROUGH:dd-mmm-yy
        /SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
        /TODAY
        /EXCLUDE:filespec

The TYPE command displays the contents of text files on your
terminal.

To obtain help on the above qualifiers, type the following:

    HELP TYPE qualifier

File(s)?
```

The same help text is printed on your terminal, but the prompt returns, meaning the TYPE command is still waiting for you to list the files you want typed.

You can also get help on a specific subtopic while being prompted by a command by responding to the prompt with a question mark (?) followed by the subtopic. For example, after getting help on SET, you can also get help on a specific function of SET:

```
$ SET [RET]
Function? ? [RET]

 The SET command establishes or changes the following:

 [DAY]TIME    DEFAULT     DEVICE         ERROR_LOG    GROUPFLAGS
 HOST         LIBRARY     [NO]PARTITION  PRIORITY     PROTECTION
 PROTECTION/DEFAULT       QUEUE          SYSTEM       TERMINAL

 For information on the above commands, type HELP SET commandname.
```

To get help on a specific topic, enter a question mark followed by the topic as follows:

```
Function? ? DEFAULT [RET]
 SET DEFAULT [ddnn:][directory]]
 The SET DEFAULT command sets your default directory or device,
 or both.
Function?
```

You can also get help by typing a question mark in response to the dollar sign prompt ($).

If you should decide after reading the help text that you have chosen the wrong command, enter a CTRL/Z in response to the prompt to end the execution of the command. (A CTRL/Z pressed before entering the command always cancels the command.)

There may also be help files providing information on special aspects of your installation. In addition, you can create local help files for your own use. See Chapter 3 for more information on the HELP command. For information on how to write your own help files, see Chapter 11.

2.3.4 Abbreviations

It is rarely necessary for you to type either the complete command name or the complete qualifier name. You only need to type the characters needed to distinguish the command or qualifier from all others.

For example:

- TYPE can be abbreviated T because it is the only command beginning with that character.
- INITIALIZE can be abbreviated INI, but not IN.
- INSTALL can be abbreviated INS, but not IN.

Three letters will usually be enough; four letters will always be enough. You can often omit other parts of commands as well. You should experiment to find how short you can abbreviate various commands. For instance, the following command:

```
$ SET TERMINAL/VT100 [RET]
```

is the documented format for the command that sets a terminal as a VT100. The same command to VAX/VMS DCL does the same thing. However, the Micro/RSX version of DCL permits you to type the following:

```
$ SET VT100 [RET]
```

to achieve the same result. This second form does not work on VAX/VMS systems.

These briefer forms should be used interactively only, and not used when you are making a permanent record or creating an indirect command file (or batch job).

For your convenience, some frequently used commands have special brief forms as follows:

Command	**Brief Form**	**Command**	**Brief Form**
ABORT	A	HELP	H
BROADCAST	B	HELP	?
COPY	C	LOGOUT	LO
DIRECTORY	D	LINK	L
DEALLOCATE	DEAL	MACRO	M
DEASSIGN	DEAS	PRINT	P
EDIT	E	RUN	R
FORTRAN	F	SHOW	S
		TYPE	T

To save time and typing, use these brief forms to replace the command names when you are entering commands.

Note

As new commands are added in future releases, abbreviations may change.

2.3.5 Numbers and Dates

DCL recognizes both octal and decimal numbers. You usually do not have to identify a number as octal or decimal, as DCL takes care of it. In rare cases, the command description directs you to add a decimal point to identify a decimal number.

DCL recognizes dates in two forms:

```
dd-mmm-yy as in 21-JUN-85
```

or

```
mm/dd/yy as in 06/21/85
```

System displays are always in the first format.

2.3.6 Multiple Parameters

Some commands permit you to enter a list of parameters instead of just one. If you are entering a list of parameters, each parameter must be set off by commas. For example,

```
$ PRINT JANE.TXT [RET]
```

causes a single file to be printed, while

```
$ PRINT JANE.TXT, CHRIS.TXT, MULP.TXT [RET]
```

causes three files to be printed. You have the option of including spaces on either side of the comma in lists.

If you end the list with a comma, DCL prompts you for further parameters. For instance:

```
$ PRINT JANE.TXT, CHRIS.TXT, [RET]
File(s)?  MULP.TXT [RET]
```

Some commands for program development accept a list of arguments to a single qualifier or parameter. In such cases, the list of arguments must be enclosed in parentheses, with the elements set off by commas. For example:

```
$ LINK/CODE:(PIC,FPP) HIYA [RET]
```

If you need to enter only a single argument, you do not need the parentheses. For example:

```
$ LINK/CODE:PIC HIYA [RET]
```

2.3.7 Underscore Character

The underscore character (_) is used to make DCL commands more readable where two words are needed to name a single command element, such as PRINT/FLAG_PAGE. However, you need not type the underscore to enter the command. PRINT/FLAGPAGE is the same as PRINT/FLAG_PAGE. However, PRINT/FLAG PAGE will not work. You cannot include a space in a command field.

2.3.8 Colon and Equal Sign

The command descriptions in this manual show arguments set off by a colon (:), such as follows:

```
$ PRINT/COPIES:2 IZZY.TXT [RET]
```

You can usually replace such colons with an equal sign (=), as in this example:

```
$ PRINT/COPIES=2 IZZY.TXT [RET]
```

Colons in device names, such as DU1:, cannot be replaced by equal signs.

2.3.9 Quoting Strings

If you wish to include an exact string in a DCL command, put the string in quotation marks. For instance, the message

```
$ BROADCAST/ALL Rock and roll will never die [RET]
```

is broadcast as

```
11-MAY-85 13:55         From PRINCE::WRITERS (TT64) to ALL
ROCK AND ROLL WILL NEVER DIE
```

while

```
$ BROADCAST/ALL "Rock and roll will never die" [RET]
```

is broadcast as

```
11-MAY-85 13:56         From PRINCE::WRITERS (TT64) to ALL
"Rock and roll will never die"
```

You need quotation marks when passing commands to tasks using the /PARAMETERS qualifier to MOUNT or the /COMMAND qualifier to RUN or INSTALL.

2.3.10 DECnet and DCL

Your system may include the optional DECnet networking software. If so, you'll need the following information.

DECnet is a DIGITAL product that enables two or more systems to "talk" to each other. These systems are linked together physically to form a *network*. The purpose of a network is to allow the users on different systems to share information and resources. See the *Introduction to DECnet* for more information about this product.

Each system in a network is called a *node*. The system that you originally log in to is your *local node*. All other systems in a network are called *remote nodes*.

Some DCL commands accept DECnet node names as part of a file specification. They are the following:

APPEND	COPY	CREATE	DELETE
DIRECTORY	RENAME	TYPE	

Several other DCL commands accept node names in the file specifications. However, DECnet modifies the basic syntax of these commands. These commands are: PRINT, SUBMIT, SET PROTECTION, and SET FILE. Be sure to check the *RSX DECnet Guide to User Utilities* before attempting to use these commands.

If the node you select is part of your network, you can simply add the node name to the file specification in the appropriate DCL command. Note that your terminal remains connected to your local node when you issue these commands to a remote node.

In the following example, you issue a command to type on your terminal the file ROMAN.TXT, which is located on remote node PRINCE and device DB2:, in directory [PETER].

```
$ TYPE PRINCE::DB2:[PETER]ROMAN.TXT RET
```

All the usual rules about file and volume protection are maintained, of course. See Section 5.6.

If the remote node you specify has a different style of file specification from Micro/RSX, you must enclose the file specification in quotation marks, as shown:

```
$ TYPE NEMO::"DISK$USERDISK:[DALTON]DESPERADO_READERS.DIS" RET
```

In addition to transmitting commands between nodes, DECnet also allows you to connect your terminal to a remote node. See the SET HOST command, described in the following section.

2.3.10.1 SET HOST

After you have logged in to a system, you can use the SET HOST command to connect your terminal to a different system.

Both your current system and the remote system must run DECnet software. In addition, you need to have an account on the remote system. Otherwise, you will not be able to log in to the remote system after you issue the SET HOST command.

Format

```
SET HOST   nodename
```

Parameters

nodename

Specifies the name of the remote system that you want to connect your terminal to.

A node is one system within a network of systems. The system that you originally log in to is called a local node; all other systems in a network are called remote nodes.

Once you have connected to the remote node, that operating system responds with a prompt. After you log in, use commands that the remote operating system accepts.

Type the LOGOUT command to log out of a remote node. After typing this command, you are located on your local node.

You can only use SET HOST to connect to one remote node at a time. For example, suppose you want information located at two different remote nodes, KING and JUNE. You cannot connect first to remote node KING, and then execute SET HOST again to connect to remote node JUNE. You must first log out of KING, which relocates you on your local node, then use SET HOST to connect to JUNE.

See the *RSX DECnet Guide to User Utilities* for a full explanation of this command.

Example

```
$ SET HOST TOOTSI [RET]
Connected to "TOOTSI", System type = Micro/RSX
System ID:  RSX TIMESHARING
$ LOGIN [RET]
Account or Name: Erin [RET]
Password:
     .
     .
     .
$
```

This SET HOST command connects your terminal to the remote node TOOTSI. The remote system identifies itself, then prompts you. You log in to the remote system, using the name and password of your account on that system.

2.3.10.2 SHOW HOST

The SHOW HOST command displays the name of the processor to which your terminal currently is connected. The display also shows you the name and version number of the operating system running on the processor.

The SHOW HOST command is most useful after you have connected your terminal to a remote system with the SET HOST command. However, SHOW HOST works whether or not your system runs DECnet software. Without DECnet on your system, this command simply displays information about your local operating system.

Format

SHOW HOST

Example

```
$ SHOW HOST [RET]
HOST=TOOTSI Micro/RSX V3.0 BL24
```

This example indicates the display from SHOW HOST. The name of your current processor is TOOTSI, which is running Version 3.0 of the Micro/RSX operating system.

2.3.11 Command Line Continuation

Sometimes a command will not fit all on one line. In that case you can continue the line by using a hyphen (-). When you end a command line with a hyphen and a RETURN, the DCL continuation prompt (->) indicates that you can continue entering the command line. If you are continuing a line from a prompt, such as **Task?**, that prompt is the indication that the line is being continued.

This feature permits you to enter command lines including more characters than your terminal has room for on one line.

DCL commands are limited to 80 characters in all. When you type a continuation line, count the hyphen and the two times you press the RETURN key as three of the 80 characters. Of course, each blank, as well as each punctuation mark, counts as one character.

Here is an example of line continuation:

```
$ PRINT/COPIES:2/FLAG_PAGE/AFTER:(04-JUN-85) OZY.TXT,- [RET]
->IZZY.TXT,FIZZY.TXT [RET]
```

The command is not entered until DCL encounters a line ending with a RETURN that is not preceded by a hyphen. In the example, the first RETURN does not enter the command. Only the second RETURN, with no hyphen, enters the command. The RETURN can be on a line by itself.

2.3.12 Comments in Command Lines

You may want to include comments in command lines if you are keeping a permanent record, such as a batch job or command file, or simply an interactive session on a hardcopy terminal. You can include a comment in a DCL command line by preceding it with an exclamation point (!).

If the comment ends the command line, you need only a single exclamation point, as in this example:

```
$ PRINT OZY.TXT    !Poem by Shelley [RET]
```

If the comment is within the command line, you need two exclamation points to set it off, as in this example:

```
$ PRINT!Parody of Shelley!   IZZY.TXT [RET]
```

These comments are ignored and not interpreted in any way by DCL.

Comments can be placed at any natural break in the command line: between qualifiers, between parameters, even as part of a response to a prompt. Another example:

```
$ PRINT/FORMS:2!Letter-quality printer! IZZY.TXT [RET]
```

2.3.13 Errors

You can correct typing errors or change the line completely by using the DELETE key or CTRL/U, provided you have not terminated the line.

You can get rid of any DCL command by pressing a CTRL/Z, provided you have not terminated the command.

If the system detects an error in the command line input, it returns the appropriate error message at the issuing terminal.

Here are some examples of incorrect commands and the error messages they produce:

```
$ PRIJT IZZY.TXT [RET]
DCL  --  Illegal command

$ PRINT/PURPLE IZZY.TXT [RET]
PRINT  --  Illegal or contradictory qualifier
PRINT/PURPLE IZZY.TXT
      ^

$ PRINT/COPIES:TWELVE IZZY.TXT [RET]
PRINT  --  Numeral expected
PRINT/COPIES:TWELVE IZZY.TXT
             ^
```

In the first case, the error was detected by DCL, as indicated by the first part of the error message. There is no DCL PRIJT command. The entire command was rejected.

In the second case, the command was entered correctly, but the qualifier was incorrect. The first part of the message shows that the error was detected within the PRINT command itself. The command is reprinted and a circumflex (^) points to the error.

In the third case, the command and the qualifier were correct, but the argument was in error. The message explains the error and the circumflex points to the error.

Sometimes the circumflex does not point directly at the error, but at the point at which the command started to go wrong, which may be several characters before or after the actual error. Typing mistakes are by far the most common cause of errors. Retyping the command is often all you need to do to eliminate the error. Other common causes of errors are omitting a space or other delimiter in a command line, specifying invalid devices or nonexistent files, issuing privileged commands from a nonprivileged terminal, and failing to type a sufficient number of characters to distinguish the command or command element.

The command descriptions include the most common errors produced by the commands and suggestions for correcting the errors. All the DCL error messages are listed and explained in Chapter 16.

2.4 Using the Queue Manager: PRINT and BATCH Jobs

The Micro/RSX Queue Manager provides facilities for printing files on line printers or other output devices. Files can be printed under user control or under the control of a system task or applications task. See the description of the PRINT command in Chapter 5. The description includes information on how to display and alter jobs in the print queues. A full description of Queue Manager commands for the user is in Chapter 12.

In addition, the Queue Manager provides a batch processing facility. Batch jobs allow you to use the system without requiring you to be present. This means that jobs that take a long time to run, or otherwise tie up system facilities, can be run when there are fewer demands on the system, such as at night or on weekends.

How to write and submit a batch job is explained in Chapter 8, Batch Processing. The chapter includes information on how to display and alter jobs in the batch queues.

The *Micro/RSX System Manager's Guide* includes a complete description of the process of setting up the Queue Manager and other commands affecting the Queue Manager.

2.5 Indirect Command Files in DCL

In addition to batch processing, Micro/RSX provides the Indirect Command Processor (Indirect) as a means of automatically passing commands to the operating system. If you have a series of commands to be executed in the same or similar fashion every time, you can include these commands in a file to be run by Indirect.

Indirect accepts not only DCL commands, but also directives. You use these commands and directives to program indirect command files that control your use of the system. Indirect is described in full in Chapter 9.

Chapter 3
Using Your Terminal

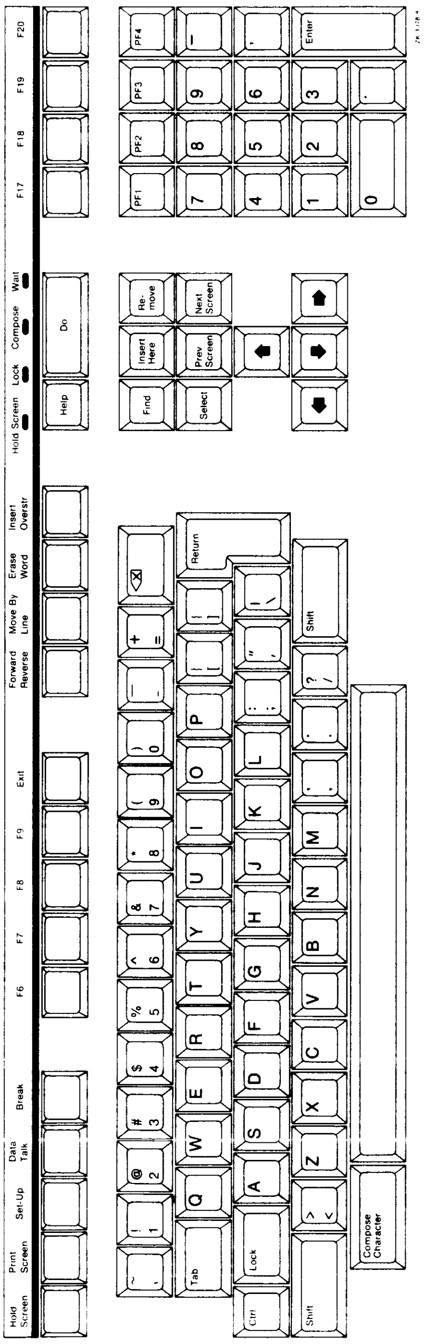
The terminal is your main channel of communication with the computer system. This chapter gives you basic information on how to use a terminal to communicate with the Micro/RSX Operating System. For a quick review of basic operations at a terminal, see the *Introduction to Micro/RSX*. For complete details on terminal operations, see the *Micro/RSX User's Guide*, Chapter 11.

The discussions in this chapter generally assume you have one of two DIGITAL terminals: the VT100- or VT200-series video terminals. If you are using another type of terminal, check with your system manager to make sure the information in this chapter applies. In any case, you should read the manual shipped with your terminal to get full use of your terminal on the system. The VT100- and VT200-series terminals have many features that can be set either by users or by programs.

A typical terminal keyboard is shown in Figure 3–1. A keyboard is a set of alphanumeric keys, similar to the set on a typewriter. The keypad shown in the illustration is a set of special keys. Where pertinent, the functions of these keys will be explained in this chapter. All terminals will have some form of a keyboard; some terminals will not have the keypad.

Each terminal on a Micro/RSX system has a number. Your terminal number is displayed when you log in. Your terminal number is also used by the system to identify tasks run from your terminal.

Figure 3–1: VT200-Series Terminal Keyboard and Keypad



You can always use the pseudo device name TI: to refer to the terminal you are currently using. You do not need a specific number. In most cases, when a system task requires you to name an output file, you can specify TI: and the output will be printed on your terminal.

3.1 Logging In and Logging Out

Logging in informs the system that you are using a terminal. Logging out informs the system that you are through using the terminal.

Most terminals have an accessible on-off switch. This switch supplies power to the terminal, but it has no bearing on whether the terminal is logged in. If you turn the power switch off, a logged-in terminal remains logged in.

You can test whether a terminal is turned on and available by pressing RETURN. The RETURN key causes the dollar sign prompt ($) to appear. If the prompt does not appear, the system may need to be rebootstrapped. See the *Micro/RSX System Manager's Guide* for more information.

The SHOW DEFAULT command tells you whether the terminal is logged in or not. If it is not logged in, the command produces an error message. You can then log in. If the terminal is logged in, the command displays the current default device and directory as well as other information.

It is not good practice to take over a terminal while another person is using the terminal, as you may interfere with the other user's work. Check with SHOW DEVICES to see if the other user has devices allocated or volumes mounted and with SHOW TASKS/ACTIVE to see if the other user is executing a task. If the other user has no significant activity under way, use your own judgment as to whether you should log the other user out, by typing the following:

```
$ LOGOUT [RET]
```

Then, after you log out the terminal, log yourself in.

If you are not sure how to log in, type HELP for help in logging in.

All Micro/RSX systems are multiuser protection systems. This means that the system includes features such as LOGIN and LOGOUT and private devices that enable several users to use the system without interfering with each other's work.

3.1.1 LOGIN

LOGIN grants access from a terminal to the system. LOGIN also establishes certain characteristics of your terminal session. HELLO is a synonym for LOGIN.

Format

```
LOG[IN]
Account or name:  userid
Password:  password

LOGIN  userid
PASSWORD:  password

LOGIN  userid/password
```

Parameters

userid

Identifies the user logging in. It is easiest to log in by name. However, this command also accepts four forms of User Identification Code (UIC):

```
[g,m]
g,m
[g/m]
g/m
```

Each user has a unique UIC, which the system manager assigns when setting up your account. The g is your group number, and the m is your member number.

If you log in using a comma in the UIC—for example, [303,17]—or your name, a file called LB:[1,2]LOGIN.TXT normally prints on your terminal. Your system manager puts information about the system and other announcements in this file.

You can suppress LOGIN.TXT messages by using a slash (/) instead of a comma (,) between the group and member numbers of your UIC when you log in.

Your system manager can set up your account so that messages will not print on your terminal when you log in or log out. If your system manager selects this option, you will not receive LOGIN.TXT messages, regardless of how you log in.

password

Your password can have up to 39 characters. These can be letters, numbers, periods (.), dollar signs ($), exclamation points (!), quotation marks ('), or hyphens (-). When you enter your password in response to the **Password:** prompt, your password will not print on the screen. However, if you type in your password on the same line as your name, the password does print. Your system manager establishes your password as part of setting up your account. You may change your password after you log in by using the SET PASSWORD command. See Chapter 11.

Examples

```
$ LOGIN RET
Account or name : KAFKA RET
Password:          RET

Micro/RSX  V3.0  BL24   [1,54] System THEFLU
15-MAY-85  10:28

Good Morning

April 20, 1985

           System Manager

** System will be down tonight from 21:00 to 24:00 **

         ** BACKUP TIME! **

Please purge your files!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

March 12, 1985         Sam Spade

Softball Practice Monday afternoon

$
```

In this example, full login text is automatically printed on the terminal because the user logged in with a name (KAFKA). The full login text is also printed if you log in with a UIC entered with a comma [g,m]. See next example.

The dollar sign prompt ($) signifies the completion of logging in.

```
$ LOGIN 303/5 RET
Password:       RET
Micro/RSX  V3.0  BL24   [1,54] System THEFLU
15-APR-85  10:28
```

```
Good Morning

March 20, 1985          System Manager:

** System will be down tonight from 21:00 to 24:00 **

            ** BACKUP TIME! **

Please purge your files!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

$
```

This example shows the first login of the day for this user. The UIC was entered with a slash (303/5). The printing of the system LOGIN.TXT file in full was therefore suppressed. Only the first message (defined as being a priority message by the system manager) was printed. The priority message is printed only on the first login of the day. This ensures that users who habitually log in with the slash do not miss anything. See next example.

```
$ LOGIN 303/5 RET
Password:        RET

Micro/RSX  V3.0  BL24    [1,54] System THEFLU
15-APR-85   11:32

Good Morning

$
```

This example shows a subsequent login on the same day by the same user. Note that no system messages are displayed.

```
$ LOGIN RET
Account or name: 301/370 RET
Password:        RET

Micro/RSX  V3.0  BL24    [1,54] System THEFLU
15-APR-85   19:36

Good Evening
$ @LOGIN.CMD
$ SET TERMINAL/INQUIRE
$ SET PROTECTION:(S:RWED,O:RWED,G:R,W:)/DEFAULT
$ ASSIGN DU1: RS:
$ @ <EOF>
$
```

This user has prepared a LOGIN.CMD file in his directory to set up the terminal, establish a default protection code, and make a logical assignment. The operating system runs the LOGIN.CMD file each time the user logs in. The commands shown are issued by the indirect command file LOGIN.CMD; the @ <EOF> marks the end of this command file. See Chapter 9, Indirect Command Processor, and the *Introduction to Micro/RSX* for more information on indirect command files.

Your system manager may have established your account so that you do not receive login messages, regardless of how you log in. If this is the case, none of the information that usually appears between the **Password:** prompt and your LOGIN.CMD file prints on your terminal. Therefore, you do not see the system identification or messages from the system manager's LOGIN.TXT file.

Notes

HELLO is identical to LOGIN.

LOGOUT, or BYE, counteracts LOGIN.

When you log in, the system's Account File Maintenance Program (ACNT) establishes many characteristics of the terminal session you are beginning. These include your default device (where your files are located), whether your account is privileged or not, and so forth.

Your system may have a file called LB:[1,2]SYSLOGIN.CMD. This file usually contains system-level commands that your system manager wants the operating system to execute each time a user logs in.

You can create a file called LOGIN.CMD. You keep this file in your directory, and the operating system executes it each time you log in. A LOGIN.CMD file usually includes commands for setting up your terminal and your account for regular use, although this file may contain any commands you want. See the examples. See also Chapter 9 and the *Introduction to Micro/RSX* for more information on indirect command files.

It is wise to log in occasionally with either your name or a comma in your UIC. This way you can be sure you have not missed any important messages.

Error Messages

MCR—Not logged in

DCL—Not logged in

Explanation: The terminal is not logged in.

User Action: Log in using LOGIN.

LOG—Account file open failure

Explanation: The system could not access the account file for some reason.

User Action: Try again. Wait and try again. If the message recurs, see your system manager.

LOG—Invalid account

Explanation: The name or UIC, or the password, given in the command is not recorded in the account file.

User Action: Retype the command using correct data.

LOG—Logins are disabled

Explanation: The system is being shut down, or a privileged user has issued SET NOLOGINS for some other reason.

User Action: You cannot log in. Try again later. Often, when logins are enabled again, the operator sends a message to all terminals.

LOG—Other user logged on

Explanation: Another user is logged in. Only one user at a time can log in on a terminal.

User Action: Issue SHOW USERS to find out who is logged in. Issue SHOW TASKS ACTIVE to find out what tasks are active on the terminal. At your discretion, issue LOGOUT and then log yourself in.

LOG—Terminal allocated to other user

Explanation: The terminal has been allocated (made private) by another user. You cannot log in on an allocated terminal.

User Action: Go to another logged in terminal. Issue SHOW DEVICES TT: and find out which terminal has your terminal allocated.

3.1.2 LOGOUT

LOGOUT logs the user off the system.

LOGOUT also aborts any active nonprivileged tasks running from the terminal, as well as dismounting any private volumes and deallocating any private devices allocated from the terminal.

Format

LO[GOUT][/HOLD]

Command Qualifier

/[NO]HOLD

The /HOLD qualifier is for use on remote or DECnet host terminals. If you include the qualifier, the terminal is logged out but the line is not disconnected. This means you can log back in without reconnecting the line. The default is /NOHOLD, meaning that when you log out, the line is also disconnected.

Examples

```
$ LOGOUT RET
Connect time:      0 hrs 44 mins  0 secs
CPU time used:     0 hrs  5 mins 36 secs
Task total:      90
Have a good afternoon
08-MAY-85 16:58 TT10: Logged off PRINCE
```

In this example, a user logs out of a Micro/RSX system. The information on system use only prints on systems that have Resource Accounting running. (PRINCE is a DECnet node name. DECnet is a DIGITAL product that allows the users of different computer systems to share information and resources. A node is the name of one of these systems.)

Your system manager may have set up your account so that you do not receive system information when you log out. In this case, the operating system simply prints a dollar sign prompt ($) to indicate that you have successfully logged out.

```
$ LOGOUT [RET]
DMO -- TT1: Dismounted from DU1:
Connect time:     0 hrs 32 mins  0 secs
CPU time used:    0 hrs  0 mins 12 secs
Task total:      21
Have a good morning
08-MAY-85 11:17 TT10:  Logged off ROMPER
```

In this example, the user had a volume mounted on device DU1:. The LOGOUT command dismounted the volume, as the message states. If the device was allocated, the LOGOUT command will deallocate it, but will not issue a message.

```
$ LOGOUT [RET]
13:29:36  Task "TT10  " Terminated
          Aborted via directive or CLI
$
Have a good afternoon
19-JUN-85  13:13  TT10:  Logged off
```

In this example, the user had a task running from the terminal at the time the LOGOUT command was issued. The command caused the task to be aborted. The system did not include Resource Accounting, so no system-use statistics were printed.

Notes

BYE is identical to LOGOUT.

LOGIN, or HELLO, counteracts LOGOUT.

You need type only LO to log out.

3.2 The Keyboard

Most of the keys on the main keyboard are self-explanatory. They function much the same as they do on a typewriter. Only the most important differences from typewriter operation are noted here.

You should remember that a computer terminal is not a typewriter. A computer terminal has two functions: transmitting input to the computer and receiving output from the computer. Usually, when you press a key, the letter appears immediately on your terminal. When you pressed the key, you sent a character to some system task or program. One of the actions the system normally takes upon receiving a character is to send it back to your terminal as an echo. This usually happens so fast that it appears that you are typing on your terminal, but you are not. You are sending input to the system and it is sending output to your terminal. A terminal is an input/output device.

Occasionally, when heavy demands are being made on the system, there may be a noticeable lag between the input and the echo.

You should also remember that keys may behave differently when sending output to some task other than DCL. For instance, all the keypad keys, plus the ENTER, BACK SPACE, and LINE FEED keys, have special meanings when you are running the EDT editor. Special applications software at your installation may also use special meanings for certain keys. This chapter covers key use for DCL (also called monitor level).

3.2.1 RETURN and Command Line Length

The RETURN key has two functions. It is used in the conventional way to supply a line feed and a carriage return, but the key is also used to signal the system that you have finished typing a command. Commands can be quite varied in length; the RETURN key defines the length of the command. The symbol RET is used in this manual to signify that you press the RETURN key. Thus,

```
$ PRINT IZZY.TXT  RET
```

is a complete command, and so is

```
$ PRINT/COPIES:2 IZZY.TXT  RET
```

If you must enter a command that is longer than a single line on your terminal, you can override the effect of the RETURN key by ending the line with a hyphen before pressing the RETURN key. This works as follows:

```
$ PRINT/COPIES:2/FORM:1/DELETE IZZY.TXT, OXY.TXT,- RET
->MADISON.AVE, FUZZY.TXT;12, GUNGA.DIN;2  RET
```

The hyphen tells the system to postpone execution of the command until it receives a RETURN not preceded by a hyphen. In the example given, the user had more files to be printed together than could be specified on a single line. The hyphen called for the special continuation prompt (->) rather than execution of the command. The RETURN without the preceding hyphen then called for execution of the full command. For more information on command lines continuation, see Chapter 2.

Note

In most examples and command formats in this manual, the RETURN at the end of a line containing a command or command element is assumed and is not shown.

3.2.2 Line Terminators

To enter a command, terminate the command line with the RETURN key. To cancel a command you do not want to enter, terminate the command line with CTRL/Z or CTRL/C.

The ENTER key on the keypad is identical with RETURN at DCL monitor level. (The two keys function differently in EDT, however.)

When you terminate a command line with the RETURN key, as soon as the command is processed, the dollar sign prompt ($) reappears, ready for another command. For example:

```
$ SHOW TIME [RET]
09:59:21 13-MAR-85
$
```

3.2.3 DELETE Versus BACK SPACE

You should avoid using the BACK SPACE key. This key is included on terminals for compatibility with other systems but is not used in DCL. The BACK SPACE key does not produce an echo, but it can have confusing results when used in a file or when entering a command. When you make a typing mistake, use the DELETE key (on the VT100-series terminals) or <X] key (on the VT200-series terminals) to correct it, not BACK SPACE. (The BACK SPACE key may be used for special functions within tasks, as it is in EDT.)

The DELETE key erases the character immediately to the left of the cursor.

3.2.4 CAPS LOCK and SHIFT

On most video terminals, the CAPS LOCK key (on the VT100-series terminals) and the LOCK key (on the VT200-series terminals) causes all letters to be typed in uppercase. This key affects only letters, not numbers or symbols. You can engage CAPS LOCK and type 'PDP-11' without disengaging it, even though the apostrophes, numbers, and the hyphen are lowercase.

The SHIFT keys set your terminal to uppercase for the keys not affected by CAPS LOCK, as well as for individual alphabetic characters.

Different terminal models vary widely in their treatment of CAPS LOCK or the equivalent key. You should experiment on your terminal.

3.2.5 NO SCROLL or HOLD SCREEN

The NO SCROLL key on the VT100-series terminals and the HOLD SCREEN on the VT200-series terminals control the flow of information on your terminal screen. These keys are useful when you want to read text on your terminal screen at a rate slower than the terminal presents it. Press NO SCROLL or HOLD SCREEN to stop information from moving upward and off your screen; you press the key again to resume the flow of information.

Note that you do not lose any information when you use the NO SCROLL or HOLD SCREEN keys.

If your terminal appears to be doing nothing, you may have inadvertently pressed NO SCROLL or HOLD SCREEN. If your terminal does not have either a NO SCROLL or HOLD SCREEN key, see the discussion of CTRL/S and CTRL/Q in Chapter 11.

3.2.6 Other Keys

See Chapter 11, More About Using Your Terminal, for more complete information on keys, character names, and other terminal specifics.

3.3 The Keypad

The VT100-series terminals have a second set of keys to the right of the main keyboard, arranged much like a calculator's keypad. These keys make up the alternate keypad.

The keys of the alternate keypad are available for special functions. When you use EDT, the DIGITAL standard editor, you can use these keys to easily edit text. In addition, some functions of the operating system use these keys. They may be used in many other ways, depending on the other applications your operating system is running.

The following list describes only the uses of the alternate keypad while you are in DCL. See Chapter 4 for a description of how you use these keys to execute EDT editing commands.

Keypad Key	Function
NUMBERS	Work in the same way as the numbers on the regular keyboard.
PERIOD	Works in the same way as the period on the regular keyboard. It can be used for including the decimal point in numbers.
ENTER	Works the same as the RETURN key on the main keyboard.
PF KEYS	Reserved for special functions. They are not used at monitor level.
ARROW KEYS	Used for cursor control by EDT, but are otherwise available for special functions.

The VT200-series terminals have both the alternate keypad and, between it and the keyboard, another keypad with editing functions printed on the keys. The middle set of keys make up the editing keypad.

The six labeled keys on the editing keypad are available for special functions. They are defined by EDT as well as by many applications.

The keys of both keypads can have odd effects if pressed accidentally. These keys each send an ESCAPE character plus one or more other characters to the operating system. The ESCAPE is a nonprinting character, but the associated printing characters may be echoed on the terminal causing an error, such as

```
RSX -- Illegal function
```

No harm is done by this.

3.4 The CTRL Key and Control Characters

The key marked CTRL is called the control key. It is used much like the shift key on a regular typewriter. If you hold down the CTRL key and then press another key, you are sending a command to the operating system. The CTRL key alone has no effect.

The combination is designated by CTRL/a, where a is the chosen letter key.

When the combination is echoes on your terminal, the CTRL key is represented by the circumflex (^). Press a CTRL/U and you see a ^U on your terminal.

Micro/RSX recognizes 11 letters combined with the CTRL key. Only the most important are discussed here. See Chapter 11, for information on the others.

With the exception of CTRL/Z, the control characters are commands directed to the operating system and not to any task you may be running from your terminal. This means that they usually work without interfering with such tasks. For instance, CTRL/O is used to skip over terminal output. It skips over any output, whether sent by DCL, the Task Builder, an editor, or any other system function. It only skips output. It does nothing else.

Note

Although you must press two keys to enter a control character, the system considers a control character as a single character.

3.4.1 CTRL/C

On most Micro/RSX systems, CTRL/C aborts a task running from your terminal. At any time that you want to stop a task from running, simply press CTRL/C. This does no harm to the task itself. You should remember, however, that the task has stopped whatever it was doing and therefore may have left some file or job in an incomplete state. In general, this will not be a problem, but you should be aware of the situation.

More sophisticated users can alter CTRL/C's behavior. See Chapter 11, for more information.

3.4.2 CTRL/O

This control character is used to skip over unwanted output to your terminal. It is analogous to the fast-forward control on a tape recorder.

If you type a single CTRL/O in response to undesired output, the output stops appearing on your terminal, but the system continues to generate the output. It can do this much faster when it does not have to print it on the terminal. If you type another CTRL/O, the output picks up at the point the operating system has reached, not the point at which you typed the last CTRL/O. If you do not type another CTRL/O, the system continues to generate the output until it reaches the end, whereupon a prompt appears and you may continue in response to the prompt.

If you are printing multiple files on your terminal, the CTRL/O will be canceled at the end of each file. For example, if you have typed

```
$ TYPE *.LST RET
```

the system will start with the first file of the type .LST in your directory. If you suppress this output with a CTRL/O, the system will start printing again as soon as it reaches the beginning of the next file of the type .LST.

3.4.3 CTRL/Z

A CTRL/Z in response to a prompt or at the end of a command line cancels the command.

CTRL/Z is also used as a command within many system tasks.

If your terminal does not respond, a CTRL/Z will often clear it.

A CTRL/Z in response to a request for input from an indirect command file cancels the execution of the remainder of the file.

Fundamentally, CTRL/Z signals the system that you have finished typing input.

3.4.4 CTRL/U

CTRL/U deletes the current line on your terminal. It is as if the line had never been typed. The system responds with a carriage return and a line feed so that the line can be retyped.

If you use CTRL/U with a CREATE command to delete a line, the line you have deleted will appear as a blank line within the file.

In three instances, a ^U may appear on your terminal without your having entered a CTRL/U.

1. The system automatically sends a CTRL/U to any terminal that has been awaiting input past a 4-minute time-out limit. This means that any line that is not terminated within the 4-minute limit is wiped out, as indicated by the CTRL/U that appears at the terminal. If you have typed in a line and then deleted it with the DELETE key, the system still interprets this as a line awaiting input and sends the CTRL/U when the time-out limit passes.
2. If you try to type more than 80 characters on a line without using a hyphen to continue the line, the system automatically cancels the command and indicates this by sending a ^U to the terminal. See Chapter 2, How To Use DCL, for more information on line continuation.
3. Finally, if there is no available pool space when you type in a command, the system sends a ^U to your terminal informing you that your command has been canceled. Pool is the section of the computer memory where Micro/RSX keeps track of what it is doing. When the system is very busy, the system may run out of available pool space and refuse to accept your command. If this happens, the condition will probably be corrected shortly. Try typing the command again. If this does not work after some reasonable interval, you may want to consider rebootstrapping the system. See the *Micro/RSX System Manager's Guide* for more information.

3.4.5 CTRL/G

CTRL/G inserts a beep (buzz, bell) in a file or message. CTRL/G cannot be the first character in a line. See Chapter 4, EDT, for more information on including this signal in your files.

3.5 HELP

HELP displays information about DCL commands and other information about the operating system. Users can easily add to the information shown by HELP.

HELP is the only command that you can issue without logging in.

Format

HELP[/OUTPUT:filespec][/qualifier] [%] [parameter1] [...parameter9]

Command Qualifiers

/OUTPUT:filespec
/LOCAL
/GROUP
/DCL
/FILE:filespec
/filename

Parameters

No parameters are required for the HELP command and there are no prompts. You can get help on any given DCL command or subtopic, however, by typing a question mark (?) in response to the prompts from DCL. You can also get help by typing a question mark in response to the dollar sign prompt ($). Examples follow at the end of this section.

The HELP command without qualifiers or parameters displays the list of commands for which help is available. This display also contains information on getting more help.

The displays for all forms of the HELP command are stored in help files, which are text files in help file format. See Chapter 11, for information on writing help files. Users can write help files for their entire system, for a group of users, or for themselves alone.

parameter1

If you include a parameter in the HELP command, you will jump over the initial HELP display to a display concerning the parameter you have included. Examples follow at the end of this section.

Usually this parameter will be a command name. When seeking help on a command, you should always start with a single parameter. The display always includes directions for getting any available further help.

parameter9

There can be up to nine levels of help. In any case, the number of parameters you enter determines the display shown; you always jump over intervening displays.

Command Qualifiers

/OUTPUT:filespec

Permits you to name an output file where the requested help text is to be saved. Use this feature to make your own library of help files that you have found useful. You must be logged in to use this qualifier.

The following qualifiers are mutually exclusive. These qualifiers have the effect of naming a file where the requested help text is located. No matter which qualifier you use, you can still include up to nine parameters.

/LOCAL
%

Specifies that the help text is in the file HELP.HLP in the default directory on the default volume. You can specify a local help file in two ways: HELP/LOCAL or HELP%. You must be logged in to use this qualifier.

/GROUP

Specifies that the help text is in the file HELP.HLP in the directory [current group,1] on the default volume. All users with the same group number have access to this file by typing HELP/GROUP. Examples follow at the end of this section. You must be logged in to use this qualifier.

/DCL

Specifies that you want help on a DCL command. For this qualifier, the help text begins in the file LB:[1,2]DCL.HLP. This is the default for Micro/RSX systems.

/FILE:filespec

Specifies any file where help text is located. If you do not give a complete file specification, the defaults are LB:[1,2]filename.HLP. You must be logged in to use this qualifier.

/filename

Specifies that the help text begins with LB:[1,2]filename.HLP. When using this qualifier, you cannot specify the device, directory, or extension, only the file name. You must be logged in to use this qualifier.

Examples

```
$ HELP [RET]
For help on logging in, type:  HELP LOGIN

Help is available for DCL commands, utilities, and system features.
You can receive help by typing either HELP or ? in response to the
DCL prompt, followed by the desired topic. For example, for help on
the COPY command type either HELP COPY or ? COPY. To simplify HELP
text, the HELP files indicate only the first of these two formats.
Also note that you can type HELP requests in either upper or lower case.

Help is available for the following DCL commands:

ABORT    ALLOCATE    ANALYZE  APPEND      ASSIGN
BACKUP   BROADCAST   BYE      CANCEL      CONTINUE
CONVERT  COPY        CREATE   DEALLOCATE  DEASSIGN
DEBUG    DEFINE      DELETE   DIFFERENCES DIRECTORY
DISMOUNT EDIT        FIX      HELLO       HELP
HOLD     INITIALIZE  INSTALL  LOGIN       LIBRARY
LINK     LOGOUT      MOUNT    PRINT       PURGE
RELEASE  RENAME      REQUEST  RUN         SET
SHOW     START       STOP     SUBMIT      TYPE
UNFIX    UNLOCK

For information on a command, type HELP commandname.
Additional help on a particular qualifier is often available
by typing:

   HELP commandname qualifier.

For information on utilities and system features, type HELP MORE.
For the short forms of some commands, type HELP BRIEF.

In addition, your Micro/RSX system may include other software, such
as BASIC or FORTRAN and HELP will be available for those features.

For information on utilities, system tasks, and other system
information, type HELP MORE. For the short forms of some commands,
type HELP BRIEF. For help on logging in, type HELP LOGIN.
```

This example shows the initial display from HELP. The user typed HELP without parameters or qualifiers at a DCL terminal.

```
$ TYPE [RET]
File(s)? ? [RET]

TYPE[qualifier[s]] filespec[s]
      /DATE:dd-mmm-yy
      /SINCE:dd-mmm-yy
      /THROUGH:dd-mmm-yy
      /SINCE:dd-mm-yy/THROUGH:dd-mmm-yy
      /TODAY
      /EXCLUDE:filespec
```

```
The TYPE command displays the contents of text files on your terminal.
File(s)?
```

This example shows the display that results from typing in a question mark (?) in response to a prompt from DCL. The same display results from typing HELP TYPE.

```
$ SHOW RET
Function? ? RET
 SHOW thing
The SHOW command can be used to show something. The following
things can be shown with this command:
 ACCOUNTING     [DAY]TIME    LIBRARY       QUEUE     TERMINAL
 ASSIGNMENTS    DEFAULT      PARTITIONS    SYSTEM    UIC
 CLOCK_QUEUE    DEVICES      PROCESSOR     TASKS     USERS
 COMMON         ERROR_LOG    PROTECTION
 Abbreviation: S
Function ? ? TIME RET
 SHOW DAYTIME
 SHOW TIME
The SHOW TIME command displays the current time and date. The time
is in 24-hour format and the date is formatted as dd-mmm-yy.
Function?
```

This example shows the display that results from typing a question mark (?) and a parameter in response to a prompt from the SHOW command. The same display results from typing HELP SHOW TIME.

3.6 BROADCAST

BROADCAST displays a specified message at one or more terminals.

Format

BROADCAST
To? [ttnn:]
Message? message

BROADCAST/qualifier
Message? message

BROADCAST username message

BROADCAST @filespec

Command Qualifiers

/ALL
/LOGGED_IN

Parameters

ttnn:

Specifies the terminal to which you want the message to be sent. Terminals need not be logged in to receive messages, but they must be turned on (powered).

If you do not specify a destination for the message, the message is sent to pseudo device CO:, the operator's console.

message

The message must fit on a single line, but the final character can go in column 80. The message can include any printing character. Lowercase letters are converted to uppercase unless enclosed in quotation marks (").

username

Specifies the user to whom you want the message to be sent. The message goes to all terminals the user is logged in on. The username must be the same as that shown by the SHOW USERS command. If SHOW USERS does not display any names, you cannot send a message by name.

If you have two users on your system with the same last names, you can separate the usernames by preceding their last name with their first initial immediately followed by a period (for example, P.ROBUST and M.ROBUST).

@filespec

Specifies the name of an indirect command file. All users can send multiple messages or multiple copies of the same message using this method. See the example. The file should contain messages in the following format:

```
TTnn:message
```

Privileged users can also use the privileged qualifiers in the following formats to send messages to multiple terminals:

```
ALL:message
LOGGED_IN:message
```

The indirect command file cannot include any Indirect directives or labels, only destinations and messages. The destination cannot be preceded by tabs or blanks.

Command Qualifiers

/ALL

This privileged qualifier sends the same message to all powered terminals, excluding slaved terminals.

/LOGGED_IN

This privileged qualifier sends the same message to all logged-in terminals.

Examples

```
$ BROADCAST [RET]
To?  TT2: [RET]
Message?  Meet me in the alley behind the barn.
```

The message is printed on TT2:.

```
30-MAY-85               From FEDERICO (TT1:)    To: TT6:
MEET ME IN THE ALLEY BEHIND THE BARN.
$
```

As the message arrives, the terminal's audio signal (beep, buzz, or bell) sounds. Notice that the message is printed in uppercase characters on the receiving terminal.

```
$ BROADCAST TT2:  "This is a one-liner." [RET]
```

This is the one-line form of the command. Because the message was included within quotation marks ("), it appears on the terminal in exactly the same form as it was sent:

```
30-MAY-85           From FELLINI (TT1:)     TO TT2:

"This is a one-liner."

$
$ BROADCAST/LOGGED_IN [RET]
Message? Everybody take the rest of the day off. [RET]
```

The message is printed on all logged-in terminals. This is a privileged command.

```
$ BRO @DAILY RET
```

The file DAILY.CMD contains the following messages:

TT1:"Where is the Ditko contract?"
TT3:"Meet me for lunch at 11:30"
TT7:"The quick brown fox jumped over the lazy dog"
TT5:"Get back to work"
TT4:"I lost the Pearson account. Have you seen it?"

The messages are sent to the designated terminals. Privileged users can also include messages preceded by ALL: and LOGGED_IN:.

Notes

B is the short form of BROADCAST.

Only terminals can receive messages. You can send a message to yourself to test the BROADCAST command, using the form BROADCAST TI:.

If you want your message to include lowercase characters when printed, enclose the message in quotation marks (").

All messages include one beep (buzz or bell, depending on the terminal). If you want your message to include extra beeps, simply press CTRL/G to add them.

If the message cannot be broadcast within 10 seconds, the system displays the following message at the initiating terminal:

```
BRO---Terminal is busy---TTnn:
```

If a user specifies multiple destinations, the system returns an error message for each busy terminal.

The BROADCAST command uses the write-breakthrough feature of the terminal driver. This means the message breaks through any kind of I/O at the destination terminal. If you are editing, for instance, the message may appear in the middle of your text, but in fact it has no effect on the text you are editing. You can issue a SET TERMINAL/NOBROADCAST command if you do not want to receive broadcasts. See Chapter 11.

Error Messages

BRO—Command input error

Explanation: The BROADCAST task did not receive the command line.

User Action: Often, this message results from a missing indirect command file. Locate the file, or check for proper syntax and enter the command again.

BRO—Command syntax error

Explanation: The command syntax was not correct.

User Action: Check for proper syntax and enter the command again.

BRO—Illegal device specified

Explanation: The destination device was not a terminal.

User Action: Check for proper device and enter the command again.

BRO—Privileged command

Explanation: Nonprivileged users cannot send messages to all connected or logged-in terminals.

User Action: Use an indirect command file to send messages to multiple terminals from a nonprivileged terminal.

BRO—User not receiving messages
BRO—TTnn:message

Explanation: The message was sent to a terminal set to NOBROADCAST.

User Action: Wait and try again.

3.7 SET and SHOW TERMINAL

SET TERMINAL and SHOW TERMINAL are complementary commands. SET TERMINAL establishes terminal characteristics, and SHOW TERMINAL displays information about terminal characteristics.

Only a few of these characteristics are described here. See Volume 2, Chapter 11, for much more information. Also, for detailed information about these characteristics, see the discussion of the full duplex terminal driver in the *Micro/RSX I/O Drivers Reference Manual,* available separately and included in the Micro/RSX Advanced Programmer's Kit.

3.7.1 SET TERMINAL

SET TERMINAL establishes various attributes of your terminal.

This command description covers the /INQUIRE qualifier, which sets the appropriate attributes of your terminal for you based on the type of terminal you are using. In addition, this section describes one other frequently used qualifier for setting terminal characteristics, /[NO]BROADCAST.

See Volume 2, Chapter 11, or the help files, for the complete list of attributes you can set, particularly if you are a programmer or privileged user.

Format

SET

Function? TERMINAL

Terminal Attribute? /attribute[s]

SET TERMINAL/attribute[s]

Attributes

/[NO]BROADCAST
/INQUIRE

Attributes

/[NO]BROADCAST

Establishes whether you want to receive broadcast messages on your terminal. The default, /BROADCAST, is to receive broadcasts.

The /NOBROADCAST qualifier limits the messages that appear on your terminal. This command prevents messages from appearing on your terminal that another user sends with the BROADCAST command. The operating system notifies senders that you are not receiving messages. In addition, this command restricts messages from the system shutdown task when it is shutting down the operating system. Your terminal does not display any messages about the impending shutdown of the system except those issued in the last five minutes. However, your terminal still receives messages from other tasks.

/INQUIRE

Tells the operating system to set all appropriate attributes for that type of terminal. The /INQUIRE qualifier is perhaps the most useful of all qualifiers to SET TERMINAL. It is good practice to include SET

TERMINAL/INQUIRE in your LOGIN.CMD file if you do not always log in to the system with the same terminal.

These attributes are described in detail in Chapter 11, Volume 2, and include all the attributes listed there under Terminal Setup. In addition, SET TERMINAL/INQUIRE sets the terminal width at 80 columns.

SET TERMINAL/INQUIRE sends a query to the terminal to find out what model it is and then issues the SET TERMINAL command for that terminal model. The following DIGITAL terminals are set explicitly: VT52, VT62, the VT100-series (VT100, VT101, and VT105 are all set VT100; VT102 is set VT102), the VT200-series, the DECmate II and the Rainbow 100-series (these two types are set VT102), the Professional 300-series, LA34, LA38, LA100, and LA120. All other terminals are set /HARDCOPY, WIDTH:80, and considered "unknown" models.

Note

You can find how all attributes are set for your terminal with an unadorned SHOW TERMINAL command. See the next section.

3.7.2 SHOW TERMINAL

SHOW TERMINAL displays information about your terminal. The display from this command includes all the attributes of your terminal. Most of these are set automatically and will be of no interest to you. For a full description of the meaning of all attributes, see Volume 2, Chapter 11.

Format

SHOW
Function? TERMINAL[:TTnn:]

SHOW TERMINAL[:TTnn:]

Qualifiers

/[NO]PRIVILEGED	/[NO]BROADCAST
/HT	/TT
/RT	/VT
/TI:	

Command Option

TTnn:

Identifies the terminal about which you want to display information. The default is your terminal, TI:.

An unadorned SHOW TERMINAL command, such as the following:

```
$ SHOW TERMINAL [RET]
```

or

```
$ SHOW TERMINAL:TT6: [RET]
```

displays all the attributes set for your terminal or the terminal you name. See the examples.

SHOW TERMINAL/ALL displays information about all terminals on the system.

Qualifiers

Each SHOW TERMINAL qualifier directly relates to a SET TERMINAL qualifier. The meaning of each qualifier, including /[NO]PRIVILEGED and /[NO]BROADCAST, is discussed under SET TERMINAL in Volume 2, Chapter 11.

/HT
/RT

/TI:
/TT

/VT

These qualifiers display information about particular types of terminals on the system. The /HT and /RT qualifiers display a list of DECnet host terminals. The /TI: qualifier displays information about your terminal and is the same as SHOW TERMINAL without any qualifier. The /TT qualifier displays a list of real terminals on the system.

The /VT qualifier displays a list of virtual terminals, which are used in batch processing.

Examples

```
$ SHOW [RET]
Function? TERMINAL [RET]
TT3:    [MCHEARTY]                09-APR-85  1     B. MCHEARTY
      CLI   = DCL     BUF  =  80.      HFILL = 0       SPEED=(9600,9600)
      LINES = 24.     TERM =  VT220    OWNER = SELF    BRO     NOABAUD
      LOWER   NOPRIV  NOHOLD  NOSLAVE  NOESC   NOCRT   NOFORM  NOREMOTE
      ECHO    NOVFILL NOHHT   NOFDX    WRAP    NORPA   NOEBC   TYPEAHEAD
      CTRLC   AVO     ANSI    DEC      NOEDIT  NOREGIS NOSOFT  NOBLKMOD
      SERIAL  NOHSYNC NOPASTHRU        TTSYNC
```

This example displays all the attributes for the user's own terminal. The command is the equivalent of SHOW TERMINAL:TI:. The attributes include the terminal number, the default directory, plus all the attributes that can be set with SET TERMINAL. See Volume 2, Chapter 11 for a full explanation of all these attributes.

```
$ SHOW TERMINAL:TT5: [RET]
TT5:     [THEFROG]                09-APR-85        1      K. THEFROG
      CLI   = DCL     BUF  =  80.      HFILL = 0       SPEED=(9600,9600)
      LINES = 24.     TERM =  VT100    OWNER = none    BRO       NOABAUD
      LOWER   PRIV    NOHOLD  NOSLAVE  NOESC   CRT     NOFORM    NOREMOTE
      ECHO    NOVFILL NOHHT   NOFDX    WRAP    NORPA   NOEBC     TYPEAHEAD
      CTRLC   AVO     ANSI    DEC      NOEDIT  NOREGIS NOSOFT    NOBLKMOD
      SERIAL  NOHSYNC NOPASTHRU        TTSYNC
```

This example displays all the attributes for another terminal.

```
$ SHO TERM/PRIVILEGED [RET]
PRIV=TT7:
    .
    .
    .
$ SHO TERM/NOPRIVILEGED [RET]
NOPRIV=TT0:
    .
    .
    .
$ SHO TERM/PAGE_LENGTH [RET]
LINES=TT7:24
```

This example displays the page size (screen size) of the terminal from which the command was issued. A nonprivileged user cannot display the page size of another terminal with this command. Use the unadorned SHOW TERMINAL and name the terminal about which you want the information.

```
$ SHO TERM/REMOTE [RET]
REMOTE=TT1:
$ SHO TERM/LOCAL [RET]
NOREMOTE=TT12:
    .
    .
    .
$ SHO TERM/NOLOCAL [RET]
REMOTE=TT1:
$ SHO TERM/NOREMOTE [RET]
NOREMOTE=TT12:
    .
    .
    .
```

This command shows which terminal numbers are assigned to telephone lines so that callers with terminals and modems can log in from elsewhere. Remote terminals access the system through dial-up lines while local terminals are hard-wired to the computer.

This attribute has nothing to do with the LOCAL/REMOTE switch found on some terminals. This attribute has nothing to do with DECnet host terminals either.

The LOCAL/REMOTE switch on a terminal sets a terminal as a typewriter or terminal, respectively.

Chapter 4

EDT

4.1 Introduction to the EDT Editor

EDT is an interactive text editor. You can use EDT to edit many kinds of text files—letters, memos, or complex computer programs. With EDT you can create new files, insert text into them, and edit that text. You can also edit text in existing files.

EDT offers many features to make text editing easier and more efficient. These features include the following:

- Three methods of editing: keypad, line, and nokeypad.
- On-line HELP. You can use HELP any time during your editing session without affecting your work.
- Journal facility. This feature allows you to recover edits lost during a system interruption.
- Multiple buffers. You can place different pieces of text in separate storage areas, allowing you to more easily organize modular programs and documents.
- Start up command files. These files enable you to personalize the characteristics of your editing sessions.
- Key definition facility. You can define keys to perform your most frequent types of edits.

- EDT macros. You can create a series of editing commands to automate your editing work.
- Tabbing facility. This feature enables you to create layered text formats. It is particularly useful for writing structured programs.

The best way to familiarize yourself with EDT is to follow the examples presented in the *Introduction to Micro/RSX*. These examples are intended for those who have never used an interactive editor, although even experienced users who do not know EDT can speed up the learning process by reading and trying them as well. Another way to learn about EDT is to use the help text available from EDT itself and try out the various operations described there.

This chapter adds to the information on EDT presented in the *Introduction to Micro/RSX*. It tells you how you can extend and customize EDT in a number of ways, and how you can use all its features. If you want even more information about EDT, see the *EDT Editor Manual,* which is available separately.

The rest of this section presents basic concepts of EDT. The following sections discuss the three types of editing available, including the line-mode commands you use to control EDT's actions and displays. Also described are advanced editing features such as macros, key definitions, startup command files, and structured tabs. Finally, there is a summary of all the EDT commands and a section that describes the EDIT command as well.

Note the following conventions used in this chapter:

- Capital letters indicate the portion of an editing command that you must type. For instance, EXit means that you only need to type EX to execute this command. Likewise, Insert indicates that you need only type the I.
- The expression "press the PASTE keypad function (GOLD 6)" means to press first the GOLD key (PF1) and then the 6 key. The keypad keys of the alternate keypad have numbers and symbols on them. However, the text in this chapter describes editing functions rather than keypad numbers or symbols. The keypad numbers are provided along with a description of a function to help you execute your edit. This convention is used in keypad mode.

4.1.1 Starting an Editing Session

To create a new text file or to edit an existing file, you type the command EDIT followed by a file specification. (After you have familiarized yourself with EDT by reading through to Section 4.6, you may want to refer to Section 4.7 for more detailed information about the EDIT command.) Note that EDT does not destroy the contents of any existing file that you edit; it simply produces a new version, leaving the old version intact. (See Chapter 5 for an explanation of how to name a file.)

The following example shows how you invoke EDT to edit a file that you named MEMO.TXT:

```
$ EDIT MEMO.TXT [RET]
```

If MEMO.TXT is a new file, you will see the message "Input file does not exist" followed by the end of buffer sign ([EOB]) and the asterisk prompt (*). (A *buffer* is a temporary storage area for your text. The [EOB] sign with no text preceding it indicates that there is no text in this file.) If MEMO.TXT is an existing file, a copy of the first line of text appears on the screen followed by the asterisk prompt (*).

The asterisk prompt (*) indicates that EDT is ready to accept a command. You are now ready to select one of EDT's three editing methods.

4.1.2 Choosing an Editing Mode

Once you enter EDT, you can choose among three different editing methods—also called *modes*. These three modes are called line, keypad, and nokeypad (which is briefly discussed only in Sections 4.1.2.3 and 4.6.6).

4.1.2.1 Keypad Mode

After you enter EDT, you can enter keypad mode (or character mode) by typing the CHANGE command after the asterisk prompt (*).

In keypad mode, you enter text by typing at the keyboard. You execute editing commands by pressing one or two keys on the terminal's numeric keypad. Unlike line mode, you can edit individual characters in the text. You move the cursor to the exact point in the file at which you want to edit the text, and you see your editing revisions as they happen.

Using keypad editing commands, you can move the cursor over *entities* of text—such as words, paragraphs, and pages—delete and undelete these entities, change the case of letters, move text around, and locate specific text in the file. In addition, you can enter line-mode commands, and you can define nokeypad commands to be executed when you press certain keys. See Section 4.3.8 for information on how to enter line-mode commands from keypad mode, and Section 4.5.2 for information on defining keys.

The advantages of keypad editing are the speed with which you can enter text and the continual display of the text you have created. You enter text by simply typing it, rather than preceding it with an Insert command, as you must do in either line or nokeypad editing. Entering commands through the keypad is fast and easy, requiring only that you press one or two keys to execute an edit. The continual display of text as you edit it means that you generally have fewer errors (you catch more of them as you enter them) and you spend less time reworking what you have already entered. These advantages make keypad mode the most popular editing style.

The only major disadvantage of keypad editing is that you cannot use it conveniently on hardcopy terminals. Almost all users of video terminals prefer keypad editing.

The format of the command keypad is different for the VT100- and VT200-series terminals. The VT200-series terminals have an additional editing keypad, which does not execute EDT editing commands. Figures 4–1 and 4–2 show the layout of the keypads for these two terminals.

4.1.2.2 Line Mode

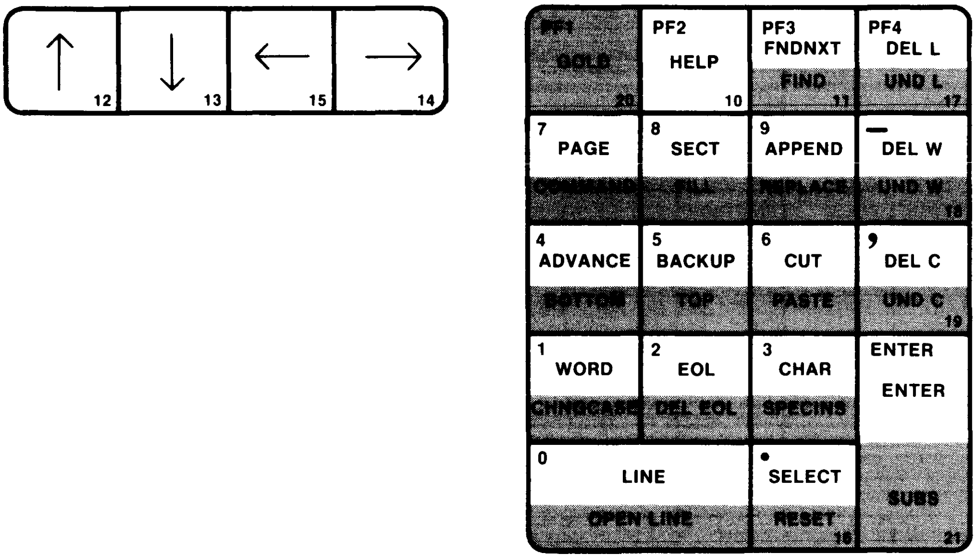
When you enter EDT, you enter line mode by default. You know you are in line mode when you see the asterisk prompt (*).

Line mode commands accomplish three types of editing tasks:

1. Editing lines of text
2. Controlling the display and automatic actions of EDT
3. Performing complex editing functions, when you use a series of these commands to define a *macro*

Line editing refers to an editing style in which the smallest unit of text you can handle is a line. You enter text into a file by typing first the Insert command and then your text. EDT numbers these lines of text for you. You revise the text by typing EDT commands after the asterisk prompt (*). Frequently, you also type line numbers to indicate the *range* of text you want the EDT command to affect. EDT executes your editing command, then—once the edit is complete—displays your revisions one numbered line at a time.

Figure 4-1: VT100-Series Keypad

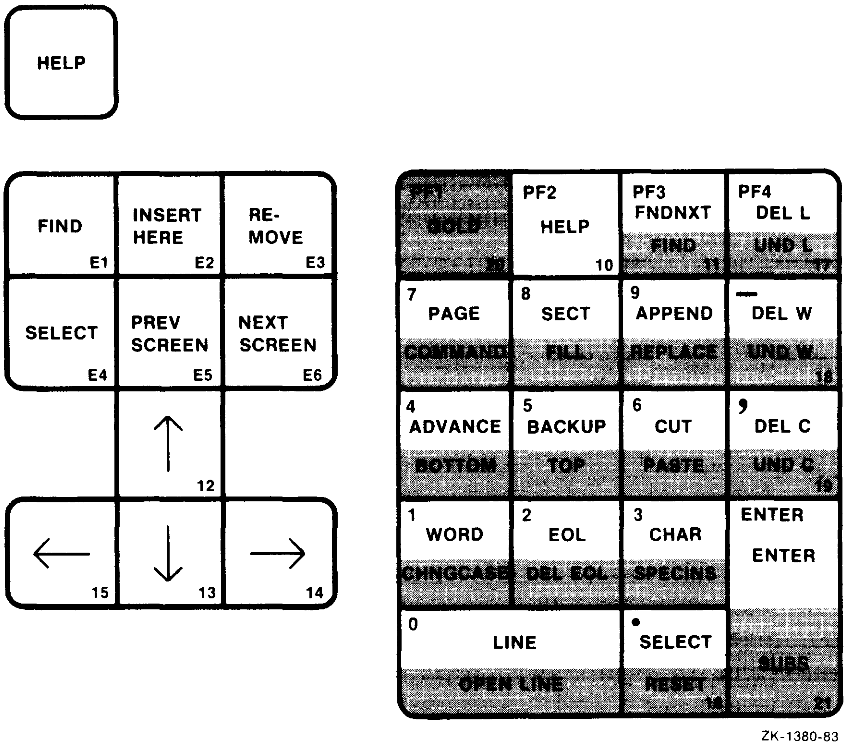


The advantages of line editing are its ease of use and its flexibility. EDT's line-mode commands are simple English words that have obvious functions. There are only a few commands, so you can learn and remember them easily. Line editing is especially handy for dealing with large blocks of text. In addition, you can work on either a video terminal or a hardcopy terminal using the same editor commands.

The disadvantages of line editing are its limited range of functions and the lack of display. For example, key definitions perform useful and often complex operations on files, but they do not work in line mode. In addition, your inability to see what you are editing may increase the number of errors in your text. Because of these disadvantages, most users who have video terminals use keypad mode, with a few line-mode commands entered from keypad mode. See Section 4.3.8 for information on how to enter line-mode commands from keypad mode.

In addition to editing lines of text, you use line-mode commands to control the automatic actions of EDT and to display information about your editing operations. For example, these line-mode commands will change you from one editing mode to another, then display for you the mode you are currently in. They can also set terminal characteristics, such

Figure 4–2: VT200-Series Keypad



as screen width, as well as copy other text files into your current editing session. See Section 4.4, which describes some frequently used line-mode commands for controlling EDT.

You also use line-mode commands to build macros. Macros are groups of line editing commands which perform complex functions that a single command cannot provide—such as deleting a line and inserting a new line all at once. See Section 4.5.1 for more information on defining macros in EDT.

4.1.2.3 Nokeypad Mode

After you enter EDT, you invoke nokeypad mode by typing the following commands after the asterisk prompt (*):

```
*SET NOKEYPAD [RET]
*CHANGE [RET]
```

If your terminal is not a VT52, VT100- or VT200-series, EDT assumes it has no keypad. In this case, if you enter a Change command after invoking EDT, you go into nokeypad editing. If you issue the SEt NOKeypad command on any terminal, any editing that follows is in nokeypad mode.

Nokeypad editing combines features of line and keypad editing. Nokeypad editing is designed for a terminal that has a video screen but no keypad. Like line editing, you do not need a keypad because you type the names of the editing commands on the keyboard. Also, you enter text into the file by typing the Insert command followed by the text you want to enter. However, nokeypad editing resembles keypad editing in two respects: you can edit individual characters in the text, and EDT displays your editing revisions as they happen.

Using nokeypad commands, you can execute such standard editing operations as deleting and undeleting text, moving text around, moving the cursor over entities, substituting one string for another, and inserting structured tabs. (See Section 4.5.4 for information on using structured tabs.) In addition, you use nokeypad commands to construct special key definitions to customize your keypad editing environment. You can also enter line-mode commands by using the EXTend command followed by the line-mode command you want.

The advantages of nokeypad editing are its power in dealing with entities of text and its use in constructing key definitions for keypad editing. Unlike line and keypad editing, nokeypad editing can work not only on lines, words, and characters, but on defined entities such as pages or paragraphs as well. This gives you great flexibility in dealing with a complex editing task, since you can refer to pieces of text more precisely. Constructing strings of nokeypad commands to use as key definitions in keypad editing allows you to use this power and flexibility along with the simplicity of keypad editing.

The disadvantages of nokeypad editing are the relatively complicated command formats and the necessity of entering text by using the Insert command. Since many of the nokeypad commands work on entities, you must remember the names of the entities as well as the commands. The command formats often accept repetition counts and directions, further complicating the process of entering commands.

Unless you have a terminal with no keypad, you will find that the nokeypad commands are most often used in constructing key definitions to be used with conventional keypad editing. All the keypad keys, plus many other keyboard keys, can be defined or redefined to perform editing functions of your own choice. See Section 4.5.2 for more information on key definition.

4.1.3 Moving Among Editing Modes

Once you become familiar with the three modes of EDT, you will want to travel between them. Table 4–1 lists the commands you need to know to move between the line and keypad modes.

Table 4–1: Moving Between Editing Modes

From	To	Command
Line Mode	Keypad Mode	*CHANGE
Line Mode	Nokeypad Mode	* SET NOKEYPAD
		* CHANGE
Keypad Mode	Line Mode	CTRL/Z
		or GOLD/7
Keypad Mode	Nokeypad Mode	CTRL/Z
		*SET NOKEYPAD
		*CHANGE
Nokeypad Mode	Line Mode	EX
Nokeypad Mode	Keypad Mode	EX
		*SET KEYPAD
		*CHANGE

The following example demonstrates how to invoke EDT to create a new file named WHY.NOT, enter line mode, enter keypad mode, return to line mode, and finally enter nokeypad mode.

```
$ EDIT WHY.NOT [RET] ❶
Input file does not exist ❷
[EOB]
*CHANGE [RET] ❸
[EOB]
[CTRL/Z] ❹
*SET NOKEYPAD [RET] ❺
*CHANGE [RET]
```

❶ You type the EDIT command to invoke EDT line mode.

❷ "Input file does not exist" appears because you are creating the new file WHY.NOT.

❸ EDT responds with the asterisk prompt (*), indicating that you are in line mode. You type the Change command after the prompt to invoke keypad mode.

❹ You press CTRL/Z to return you to line mode.

❺ The asterisk prompt (*) indicates line mode. After this prompt, you type two commands to invoke nokeypad mode: the SEt NOKeypad command followed by the Change command.

4.1.4 Getting Help from EDT

EDT help describes all the commands available and presents information on many concepts of EDT as well.

Each editing mode provides a method for getting help. If you are editing in line mode, you can type HELP for a list of topics on which help is available, or HELP [topic] for information on a particular topic. For instance, you can get information on how to include another file within the file you are editing by typing:

```
*HELP INCLUDE [RET]
```

The following command gets information on the definition of EDT line-mode macros:

```
*HELP DEFINE MACRO [RET]
```

In keypad editing, you press the HELP key PF2 to get help on keypad functions. EDT responds with a diagram showing the layout of the keypad commands for your terminal type. You then can get information on a particular command by pressing the key that executes the command.

In keypad editing, you can also press the COMMAND keypad function (GOLD 7) to get help. After executing this function, you respond to the "Command:" prompt by typing HELP [topic], then pressing the keypad ENTER key. For example, after pressing the COMMAND function, you can type the following text:

```
Command: HELP CHANGE
```

This causes EDT to display the information it has on the CHANGE command by scrolling the text of your buffer up the screen and replacing it with the help text. When you finish reading the help text, press RETURN to resume editing.

If you are doing nokeypad editing, you must type the EXTend command followed by HELP [topic], using the format for line-mode HELP to get information. (The EXTend command allows you to execute a line-mode command from nokeypad mode.) For example:

```
EXT HELP CHANGE
```

4.1.5 Ending an Editing Session

Both the EXit and QUIT commands terminate an editing session; however, only EXit saves your edits. Note that you enter these commands from line mode.

When you enter the EXit command, EDT creates an output file containing the edited version of the input file. By default, the output file has the same name and type as the input file. (The version number is incremented by one.) The following example demonstrates how you invoke and then terminate EDT to create a file named OUTPUT.DAT. The output file has the same name as the input file, and the version number increases by one.

```
$ EDIT OUTPUT.DAT RET
    .
    .
    .
*EXIT RET
DUO:[SCHEDULE]OUTPUT.DAT;5  2 lines
```

If you want to override the default, and specify a different output file name, type the EXit command followed by the new file name. In the following example, you invoke EDT to edit a file named OUTPUT.DAT and specify INPUT.DAT as the new file name when you terminate the session.

```
$ EDIT OUTPUT.DAT [RET]
    .
    .
    .
*EXIT INPUT.DAT [RET]
DUO:[SCHEDULE]INPUT.DAT;1  2 lines
```

If you do not want to save your edits when you end an editing session, type the QUIT command. All the edits you made to the file are lost, and no new output file is created.

Table 4–2 shows the three modes of EDT, and the commands you enter to terminate each mode.

Table 4–2: Ending an EDT Session

Mode	Commands to Terminate EDT
Line	* EXit or QUIT
	$
Keypad	CTRL/Z
	* EXit or QUIT
	$
Nokeypad	EX
	* EXit or QUIT
	$

4.1.6 Recovering from a Lost Editing Session

While you are editing or inserting text, EDT is keeping track of every keystroke you enter at your terminal. EDT records this information in a file called a journal file. If you leave an editing session normally, using either the EXit or QUIT commands, EDT deletes the journal file. However, when you experience a system interruption, the journal file is saved so that you can recover your lost edits.

The journal file does not contain a version of your text. Rather, it contains a record of the keystrokes you entered during the session. By combining the journal file with the text that you had at the beginning of your session, you can recover your session to a point just before the interruption.

Note

Sometimes, the last few keystrokes are missing. This is normal. No work from earlier in your session will be omitted.

To recover an editing session, use the same command line that you used to start the session originally, including the same file specification and any qualifiers you used the first time. Finally, add the /RECOVER qualifier. This tells EDT to read the journal file and execute the commands that are recorded in it. EDT reads commands from the journal file and executes them as you watch. See the examples for the EDIT command in Section 4.7.

The following example demonstrates how to recover a file named MAILLIST.TXT after the editing session was interrupted. Note that you had begun the session that was interrupted with the command EDIT MAILLIST.TXT.

```
$ EDIT/RECOVER MAILLIST.TXT [RET]
```

When you work with journal files, you will notice that they have a file type of JOU. The file name is the same as the file you were editing. The journal file for MAILLIST.TXT is MAILLIST.JOU. And, the journal file for HAMMER.LIS is HAMMER.JOU. When you enter the EDIT/RECOVER command, you enter the name of the file with its original file type, not the .JOU file type. Otherwise you will edit the journal file.

4.2 Using EDT Buffers

Buffers are temporary holding areas for text. You can use the buffers in EDT to do the following:

- Move part or all of another file into your editing session
- Create a file from part or all of the text in a buffer
- Divide one or more files into sections

See Section 4.5.5.5 for specific examples of how to use buffers.

4.2.1 EDT'S Permanent Buffers

At the beginning of an editing session, EDT automatically provides a buffer called MAIN. If you are creating a new file, you insert and edit text in the MAIN buffer. If you are editing a file that already exists, EDT puts a copy of the contents of the file into this MAIN buffer.

MAIN exists during your entire editing session. Also, you can delete only the contents of this buffer, and not the buffer itself.

The other buffer that EDT provides automatically is called the PASTE buffer. The CUT keypad function (6) deletes text and places it in the PASTE buffer. You then can use the PASTE keypad function (GOLD6) to copy the text in the PASTE buffer to other locations, either elsewhere in your current buffer or into another buffer. Every time you perform a new CUT operation, EDT clears the PASTE buffer and replaces its contents with the newly deleted text.

Like the MAIN buffer, you can edit or delete the contents of the PASTE buffer (You cannot delete the buffer itself. See Section 4.5.5.1 for information on how to move from the MAIN buffer to an alternate buffer.

4.3 Using Keypad Mode

Keypad editing is available on VT200- and VT100-series terminals. In keypad editing, the contents of a file are displayed on the screen as you edit. You can see the changes you make to a file as they take place.

In keypad editing, you press keys to perform editing functions instead of typing commands (as is done in line and nokeypad editing).

The following sections guide you through the many keypad functions. To understand how each function works, invoke EDT to create a file called TEST.EDT (Section 4.1.1) and enter keypad mode (Section 4.1.2.1). As you read each section, follow the examples and experiment with the various keypad functions in the file you just created.

4.3.1 Using the GOLD Key

You can use the GOLD key for two purposes:

1. To use the alternate of two functions on a keypad key
2. To execute a function a specified number of times

To use an alternate function, press the GOLD key followed by the desired key. (You do not need to hold down the GOLD key.) For example, to use UND C, you press the GOLD key first, and then the UND C key.

If you want to execute a function a specified number of times (for example, to use DEL C twelve times), press the GOLD key, type the number 12 (on the main keyboard), and press DEL C. Try it and see how much time you save. The following steps show how to repeat any keypad function a specified number of times:

1. Press the GOLD key.
2. Type a number.
3. Press the desired function.

4.3.2 Moving the Cursor

There are many ways to move the cursor in keypad mode. Table 4–3 lists the 15 keypad functions that move the cursor.

You can use the ADVANCE and BACKUP keypad functions to set the cursor in a forward or backward direction, respectively. By specifying the direction of the cursor, you affect eight of the keypad functions: LINE, EOL, WORD, CHAR, SECT, PAGE, FIND, FNDNXT.

Table 4–3: Moving the Cursor in Keypad Mode

Function Key	Destination of Cursor
TOP	Beginning of buffer
BOTTOM	End of buffer
LEFT arrow	One character to the left
UP arrow	Up one character
RIGHT arrow	One character to the right
DOWN arrow	Down one character

Table 4-3 (Cont.): Moving the Cursor in Keypad Mode

Function Key	Destination of Cursor
LINE	One line up or down (depending on the current direction of the cursor)
EOL	End of the current or previous line (depending on the current direction of the cursor)
WORD	Beginning of the next or previous word (depending on the current direction of the cursor)
CHAR	One character to the right or left (depending on the current direction of the cursor)
SECT	Across a 16-line section of text (up or down, depending on the current direction of the cursor) or if there are fewer than 16 lines, across the number of existing lines
PAGE	Across a page of text (by default, the text between form feeds)
FIND	Next or previous specified string of text depending on the current direction of the cursor
FNDNXT	Next or previous occurrence of search string depending on the current direction of the cursor
BACKSPACE	Beginning of the current line

The following example demonstrates how to move the cursor to the beginning (or top) and to the end (or bottom) of the buffer using the TOP and BOTTOM keypad functions.

Insert the following three lines of text:

```
Five golden rings,
four calling birds,
three french hens,
```

1. Press the GOLD key followed by TOP to move the cursor to the "F" in the word "Five".
2. Press the GOLD key followed by BOTTOM. Notice the cursor moving to the left bracket of the end-of-buffer sign [EOB].
3. Press the GOLD key followed by TOP again, moving the cursor back to the "F".

Repeat these steps to become familiar with the TOP and BOTTOM keypad functions.

The following example demonstrates how to move the cursor using the four arrow keys: UP, DOWN, LEFT, RIGHT. Insert the following four lines of text:

```
Under a toadstool crept a wee elf,
Out of the rain to shelter himself.
Under a toadstool all in a heap,
Sat a big doormouse, fast asleep.
```

1. Press the TOP keypad function (the GOLD key followed by TOP) to move the cursor to the letter "U" in the word "Under".
2. Press the RIGHT arrow 9 times, then press the DOWN arrow once, moving the cursor to the "e" in the word "the".
3. Press the LEFT arrow 5 times, then press the DOWN arrow once, moving the cursor to the "r" in the word "under".
4. Press the RIGHT arrow 8 times, then press the DOWN arrow once, moving the cursor to the second "o" in the word "doormouse".
5. Press the LEFT arrow 6 times, then press the UP arrow 3 times, moving the cursor to the word "a".

The following example demonstrates these keypad functions:

TOP
LINE
EOL
WORD
CHAR

Enter the following four lines of text:

```
ONE TWO THREE FOUR FIVE
SIX SEVEN EIGHT NINE
TEN ELEVEN TWELVE THIRTEEN
FOURTEEN
```

1. Press TOP to move the cursor to the first letter of the word "ONE".
2. Press LINE once, moving the cursor to the "S" in the word "SIX".
3. Press WORD twice, moving the cursor to the "E" in the word "EIGHT". Then press EOL. The cursor will move to the end of the line.

4. Press WORD three times, moving the cursor to the "T" in the word "TWELVE". Then press EOL. The cursor will move to the end of the line.
5. If you are using a VT100- or VT200-series terminal, press CHAR four times. The cursor will move to the "R" in the word "FOURTEEN".

Note that the PREV SCREEN and NEXT SCREEN keys (on an LK201 keyboard) work like the SECT keypad function, moving the cursor back or ahead by 16 lines.

4.3.3 Inserting Text

You can insert text simply by typing the text on the keyboard. No command is required. Whatever you type becomes part of the file. Your insertion appears on the screen as you type it and the surrounding text moves as necessary to accommodate it. The cursor marks the starting location for the insertion. When you want to begin a new line, press RETURN to move the cursor to the beginning of the next line and continue typing your text.

As you type new text, you may notice errors in surrounding text. You can move the cursor to these errors and correct them at any time, and then move the cursor back and continue to insert text.

4.3.4 Deleting and Undeleting Text

You can delete text by character, word, and line. DELETE and DEL C delete characters. DEL W and LINEFEED delete words or parts of words. DEL L, DEL EOL, and CTRL/U delete lines or parts of lines. The deleted text is stored in one of three buffers so that you can restore it using the UND commands (UND C, UND W, UND L). (A buffer is a temporary holding area for text. See Section 4.2 for more information on buffers.) The character buffer contains the last character deleted; the word buffer contains the last word deleted; and the line buffer contains the last line deleted. Only the most recent deletion can be restored; you can restore this unit of text numerous times to any location.

Figure 4–3 shows the three buffers that EDT fills when you use the delete and undelete keys.

Figure 4–3: Three EDT Buffers Used for Deleting and Undeleting Text

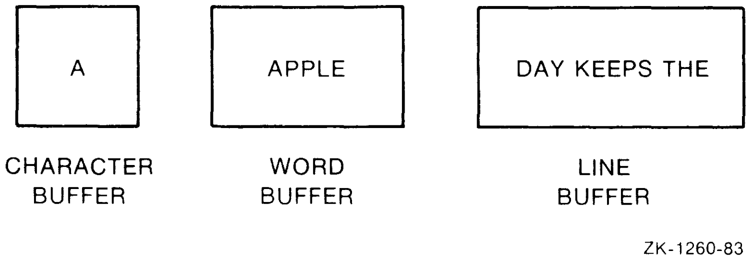


Table 4–4 lists the keypad functions used to delete text:

Table 4–4: Deleting Text in Keypad Mode

Keypad Function	Effect
DELETE	Deletes the preceding character.
DEL C	Deletes the current character.
DEL W	Deletes to the end of the word.
LINE FEED	Deletes to the beginning of the previous word.
DEL L	Deletes to the next line.
CTRL/U	Deletes from the beginning of the line to the cursor.
DEL EOL	Deletes from the cursor to the end or beginning of the line, depending on the current cursor direction.
CUT	Deletes the selected string from the current file and stores it in the PASTE buffer. The selected string is all the text between the cursor location when you pressed the SELECT keypad function and the position to which you moved the cursor. (Note that the REMOVE key on the LK201 keypad works like the CUT keypad function.)

Enter the following three lines of text. Then follow the steps to learn how to use the delete keypad functions.

```
CHARACTER
WORD WORD WORD WORD WORD
LINE LINE LINE LINE LINE LINE LINE LINE
```

1. Move the cursor to the first C in the word "character". Press DEL C. The C will disappear.
2. Press the GOLD key followed by UND C. The C will reappear.
3. Move the cursor to the H. Press DEL C to make it disappear.
4. Then press the GOLD key followed by UND C. The H is restored.
5. Press the GOLD key followed by UND C again, another H will appear.

Notice that the DEL C keypad function deletes the character directly at the cursor and the DELETE key (on the main keyboard) deletes the character immediately to the left of the cursor. You can use the UND C keypad function to restore the most recently deleted character in either case. Press the DELETE key and the DEL C keypad function to see the difference between the two.

1. Move the cursor to the W in "WORD". Press DEL W. The word "WORD" will disappear.
2. Press the GOLD key followed by UND W to restore the word.
3. Press the GOLD key followed by UND W again, another "WORD" will appear.

Notice that the DEL W keypad function deletes to the end of the current word, and the LINE FEED key (on the main keyboard) deletes to the beginning of the preceding word. Press the LINE FEED key to see the difference between the DEL W keypad function and the LINE FEED key. Again, you can use the UND W keypad function to restore the deleted word in either case.

1. Move the cursor to the beginning of the line of LINES. Press the DEL L keypad function. The entire line will disappear.
2. Press the GOLD key followed by UND L to restore the line.
3. Press the GOLD key and UND L several times to create multiple lines.

Use the CTRL/U command to delete text from the cursor to the beginning of the line. Notice that the cursor moves to the beginning of the line. Press UND L to restore the line.

4.3.5 Locating Text

Use the FIND and FNDNXT keypad functions to locate strings of text. When you press the GOLD key followed by FIND you will see the following prompt on the screen:

```
Search for:
```

Type the string of text you are looking for and press one of the following keys:

- ADVANCE
- BACKUP
- ENTER
- DO

If you press ADVANCE, EDT will search in a forward direction for the specified string. If you press BACKUP, EDT will search in a backward direction. If you press ENTER or DO, EDT will search in the direction already set.

Note

The ADVANCE and BACKUP keys set the direction for subsequent EDT commands.

Enter the text below. Press the GOLD key followed by the FIND keypad function. When you are prompted for a search string, enter the word "ticket". Then press BACKUP. The search will start at the end of the text (where the cursor is located) and continue toward the beginning of the text until it finds the word "ticket".

```
Ella will not be able to attend the concert tonight.
She has a sore throat.  Perhaps you could give the ticket
to somebody in your music class.  Ella wants to see the
program when she is feeling better.
```

```
Search for: ticket
```

Now search for the word "program". Press the GOLD key followed by FIND. When you are prompted for the search string, enter the word "program". If you press BACKUP again, you will see the response "String was not found" because the word "program" is located after the word "ticket". Try it. Now press ADVANCE followed by FNDNXT. EDT will find the string.

```
Ella will not be able to attend the concert tonight.
She has a sore throat.  Perhaps you could give the ticket
to somebody in your music class.  Ella wants to see the
program when she is feeling better.

Search for: program
```

Remember the following three items when you are entering a search string:

1. EDT ignores diacritical marks and the case of letters while making searches (unless you enter the SET SEARCH EXACT command).
2. DELETE is the only key you can use to edit an incorrectly typed search string.
3. To cancel a search string, press CTRL/U.

If you want to find the next occurrence of the string you are searching for, use the FNDNXT keypad function. EDT will search in the direction already set. If EDT cannot find the string, it will give you the message "String was not found". You can reverse the direction of the search by pressing either ADVANCE or BACKUP before pressing FNDNXT.

Enter the text below. Press GOLD and FIND. Enter the search string "little" when prompted. Because the cursor is at the end of the text, you must press the BACKUP keypad function to set the search in a backward direction. Now press FNDNXT nine times. EDT will find each occurrence of the search string. When the cursor arrives at the first "little", press the ADVANCE keypad function to reverse the direction of the search. Every time you press FNDNXT, the cursor will move forward to the next occurrence of the word "little".

```
One little, two little, three little chickadees,
four little, five little, six little chickadees,
seven little, eight little, nine little chickadees,
ten little chickadees feeding.

Search for: little
```

4.3.6 Moving Text

You can move text using three different groups of keypad functions:

1. RETURN and OPENLINE
2. DEL L, UND L, DEL W, UND W, DEL C, and UND C
3. CUT and PASTE (or REMOVE and INSERT HERE)

Using the first method, you press RETURN to move text (with the cursor) and OPENLINE to move text (without the cursor) down the screen line by line. Using the second method, you delete a unit of text (character, word, or line) from one location, move the cursor to another location, and undelete the text there. (See Section 4.3.4 for more information about deleting and undeleting text.) You can use the third method, which is described in this section, to move larger units of text.

Use the following three keypad functions to move text:

1. SELECT
2. CUT (or REMOVE)
3. PASTE (or INSERT HERE)

Press SELECT to mark one end of a string of text that you want to delete or move. Then move the cursor either backwards or forwards, to the other end of the string and press CUT. Before you press CUT, you can correct a mistake by pressing the GOLD key followed by RESET to cancel the select range and start over.

You press CUT to delete the selected string of text from the current file and store it in the PASTE buffer. (See Section 4.2 for more information about the PASTE buffer.) The selected text is all the text between the cursor location when you pressed SELECT and the position to which you moved the cursor.

If you press CUT before you press SELECT, you will see the message "No select range active".

You press GOLD followed by PASTE to insert the contents of the PASTE buffer to the left of the cursor position.

To summarize, use the following steps to move text:

1. Place the cursor at the beginning of the text you want to move.
2. Press SELECT to mark the location.

3. Move the cursor to the end of the select range.
4. Press CUT to delete the text from its current position.
5. Move the cursor to the character just beyond where you want the text inserted.
6. Press the GOLD key followed by PASTE.

Note

> If you want to restore the select string to its original location after you perform Step 4 above, press the GOLD key followed by PASTE.

Enter the following text:

```
january february march april
may june july november
december august september october
```

In order to move the string "august september october" after "july", move the cursor to the "a" in "august". Press SELECT. Then move the cursor to the "r" in "october". Press CUT. The selected string will disappear into the PASTE buffer. Now move the cursor after the word "july" and press PASTE. A copy of the selected string in the PASTE buffer will appear on the screen between the words "july" and "november".

If you press PASTE again, you will get another copy of the contents of the PASTE buffer. Try it. Every time you press PASTE, you will get another copy of the string "august september october". Once you perform another SELECT and CUT operation, specifying a different string, the contents of the PASTE buffer will change. You can always get as many copies of the PASTE buffer as you want by pressing PASTE.

If you want to add to the text in the PASTE buffer, retaining the text already there, use APPEND. APPEND deletes the select range from the current buffer and adds it to the end of the PASTE buffer.

For example, if your PASTE buffer contains the text "Wolfgang Amadeus" and you want to add the text "Mozart", follow the steps below:

1. Press SELECT.
2. Type the word " Mozart". (Precede the word with a space.)
3. Press APPEND.

Now the PASTE buffer contains the text:

```
Wolfgang Amadeus Mozart
```

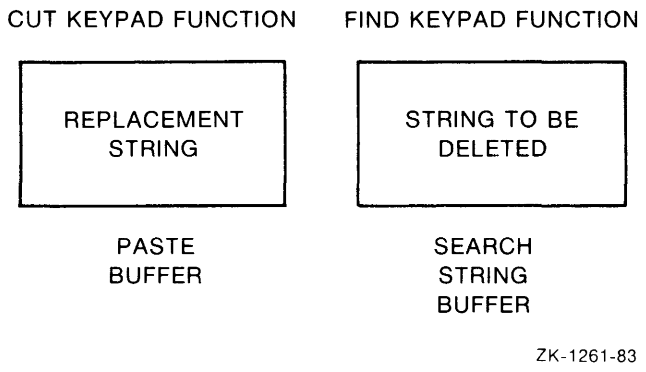
4.3.7 Substituting Text

You can use the SUBS keypad function or the REPLACE keypad function to replace one string of text with another. The following four steps demonstrate how to use SUBS to make substitutions:

1. Press SELECT and enter the replacement text.
2. Press CUT. (The select range will disappear into the PASTE buffer.)
3. Press the GOLD key followed by FIND and enter the search string.
4. Press SUBS to exchange the existing text for the replacement text.

When you make substitutions using SUBS, you are using both the PASTE buffer and the search string buffer. You use CUT to put the replacement string in the PASTE buffer and FIND to put the string you want to find and delete in the search string buffer. Figure 4–4 shows both buffers.

Figure 4–4: Two EDT Buffers Used for Substituting Text



When you use SUBS, EDT performs the substitution first and then moves to the next occurrence of the search string. If EDT cannot find another occurrence of the search string, it prints the message "String was not found". If you do not want to make a particular substitution, press the FNDNXT keypad function. The cursor will move to the next occurrence of the search string. If you want to change that one, press SUBS again. Thus, you can move through the buffer pressing SUBS each time you want to make a replacement and pressing FNDNXT when you want EDT to leave the search string alone.

Note

> You must use the FIND keypad function when you are making substitutions with SUBS because you are replacing text that matches the search string with the contents of the PASTE buffer. SUBS will not work correctly if you do not use FIND.

Enter the following text:

```
Susanne grabbed the red apple and gobbled it down.  Suddenly
her face turned quite red.  Glancing towards the red house
she saw the huge tree bathed in red leaves.
```

Now perform the following steps to substitute the word "green" for the word "red":

1. Press SELECT and enter the word "green".
2. Press CUT. (The word "green" will disappear into the PASTE buffer.)
3. Press GOLD followed by FIND and enter the word "red".
4. Press SUBS to exchange the word "red" for the word "green".
5. Press SUBS three more times.

Your resulting text should look like this:

```
Susanne grabbed the green apple and gobbled it down.  Suddenly
her face turned quite green.  Glancing towards the green house
she saw the huge tree bathed in green leaves.
```

REPLACE deletes the text in the select range and replaces it with the contents of the PASTE buffer.

For example, if your PASTE buffer contains the words "paste paste paste", and your select range contains the words "select select select", press REPLACE to make your select range contain the words "paste paste paste". The following steps demonstrate this example:

1. Press SELECT.
2. Type the words "paste paste paste".
3. Press CUT. Now your PASTE buffer contains the words "paste paste paste".
4. Press SELECT again.

5. Type the words "select select select".
6. Press REPLACE. Now your select range contains the words "paste paste paste".

4.3.8 Five More Keys to Use with the GOLD Key

Five keypad functions associated with the GOLD key are summarized in Table 4–5:

Table 4–5: Five Gold Key Functions in Keypad Mode

Keypad Function	Effect
COMMAND	Enables you to enter a line mode command from keypad mode.
CHNGCASE	Reverses the case of letters in your text. Uppercase letters become lowercase; lowercase letters become uppercase.
FILL	Reorganizes the select range so that the maximum number of whole words can fit within the current line width.
RESET	Changes the following conditions of your editing session: • Cancels an active select range. • Empties the search buffer so that there is no current search string. • Sets EDT's current direction to ADVANCE. • Sets EDT to the default DMOV state. (DMOV is a Nokeypad command that returns your editing session to EDT's default state, where EDT does not alter the case of letters during move operations.)

Table 4-5 (Cont.): Five Gold Key Functions in Keypad Mode

Keypad Function	Effect
SPECINS—special insert	Enables you to insert any character from the DIGITAL Multinational Character Set into your text using the character's decimal equivalent value. For information about the DIGITAL Multinational Character Set, see the *EDT Editor Manual,* which is available separately.

For more information about all the available keypad keys, see the *EDT Editor Manual,* which is available separately.

4.4 How to Use Line Mode

You can use EDT's line editing facility with any interactive terminal—hardcopy or screen. Line editing uses the line as its point of reference. EDT moves through the text line by line, not character by character as in the two other editing modes. Line editing commands are particularly useful for manipulating large blocks of text.

4.4.1 Line Numbers and Ranges

To help you locate and edit text, EDT assigns line numbers. These line numbers are not part of the text and are not kept when you end an editing session. To see the line numbers of an already existing file, enter the TYPE WHOLE command after the asterisk prompt (*). You will notice that the text you enter is indented 12 spaces.

The following example demonstrates how to display your line numbers:

```
*TYPE WHOLE RET
    1          oneoneoneoneoneoneone
    2          twotwotwotwotwo
    3          threethreethree
    4          fourfourfourfour
    5          fivefivefivefive
[EOB]
*
```

Line numbers have the following characteristics:

- They are assigned to every line in every buffer in every editing session, including newly inserted lines and text added with the INCLUDE command.
- They start with 1 and increment by 1.
- They are decimal numbers if newly inserted.
- They are removed by the EXIT command, unless otherwise specified.
- They can be renumbered in increments of 1 or more with the RESEQUENCE command.

When you want a line mode command to affect a specific part of the buffer, you must enter a range. Ranges can specify one or more lines. In addition, the multiple-line ranges can be contiguous or noncontiguous.

Table 4–6 lists and describes the different ranges you can specify when editing in line mode.

Table 4–6: Ranges for Line Mode

Range Type	Description
period (.)	Current line
number	EDT line number
'string'	Next line containing the quoted string
BEGIN	First line of the buffer
END	After the last line in the buffer ([EOB])
LAST	Last line EDT was at in the previous buffer
WHOLE	Entire buffer
BEFORE	All lines in the buffer before the current line
REST	All lines in the buffer starting with the current line and ending with the last line

You can use the symbols and words listed in Table 4–7 with the range types listed in Table 4–6 while editing in line mode.

Table 4-7: Symbols and Words Used in Line Mode

Symbol/Word	Description
, or AND	Used to join noncontiguous ranges in a list; only single lines can be joined in this way
: or THRU	Indicates a group of lines starting with the first range specifier and ending with the second
n	Indicates the number of lines from the current line
# n or FOR n	Indicates the next "n" number of lines
+ "string" or "n"	Indicates that "string" or "n" refers to a line or lines after the current line
– "string" or "n"	Indicates that "string" or "n" refers to a line or lines before the current line
ALL "string" or "n"	Indicates that the command applies to all lines containing "string"

The following examples demonstrate some of the range types, symbols, and words listed above.

Examples

1. `TYPE`

This command displays the current line.

2. `TYPE 35`

This command displays line 35.

3. `TYPE "December"`

This command displays the first line it encounters containing "December."

4. `TYPE BEGIN`

This command displays the first line of the current buffer.

5. `TYPE END`

This command displays the [EOB] mark.

6. `TYPE WHOLE`

This command displays every line in the current buffer.

7. `TYPE 3,6,8`

This command displays lines 3, 6, and 8.

8. `DELETE 4 AND 13`

This command deletes lines 4 and 13.

9. `DELETE 4#3`

This command deletes line 4 and the three lines following line 4.

10. `TYPE . FOR 10`

This command displays the current line and the next ten lines; the number 10 refers to the tenth line after the current line.

11. `TYPE BEGIN +6`

This command displays the seventh line of the current buffer.

12. `DELETE -3 THRU .`

This command deletes the current line and the three lines preceding it; the number –3 refers to the third line before the current line.

13. `TYPE . THRU 1000 ALL "Date:"`

This command displays every line containing the string "Date:" that appears in the group of lines starting with the current line and ending with line number 1000.

4.5 Advanced Editing Features

This section describes some features that EDT provides to help you do more complex tasks. These features include the following:

- **Defining macros**—EDT macros are groups of line-mode commands to be executed at once. You place a series of commands in a buffer and then the buffer name becomes a command to perform that series of commands.
- **Defining keys**—EDT key definition is similar to creating a line-mode macro. You can define a key to perform a series of character-mode commands, both keypad and nokeypad.

- **Startup command files**—You can define keys or macros in a startup command file and the key and macro definitions are automatically ready for use in your editing session. You can also use the startup file to set your own choice of editor characteristics. You can also use startup command files to perform batch-style editing.
- **Structured tabs**—For programmers writing in structured languages, such as Pascal, EDT provides structured tabs for easy editing. This section explains this feature.
- **EDT buffers**—You can use EDT buffers to easily move text from one buffer to another buffer, from a buffer to a file, and from a file to a buffer. This section explains how to manipulate EDT buffers.

You can do fairly complex operations with macros, and even more complex operations with key definitions, both of which are simple to write and use. Finally, you may well accumulate a number of macros and key definitions you want to save and reuse; the startup command file provides a method for doing this.

4.5.1 Defining Macros

You can group line-mode commands and name the group, using the name as a line-mode command to execute the commands. This process is called defining and executing a macro. EDT handles macros by creating a buffer with the same name as the macro and holding the commands in that buffer. When you enter the macro name, EDT pulls the commands out of the buffer with that name and executes them.

To define a macro, use the DEFINE MACRO line-mode command. This command tells EDT to create a buffer with the same name as the macro. Because the macro name and the buffer name must be the same, the rules for macro names are the same as those for buffer names: the names can be 1 to 30 characters in length and may include alphabetic and numeric characters and underscores (_). Next, enter that buffer (with a FIND =buffer, CHANGE =buffer, or =buffer command) and insert text that consists of line-mode commands. You can use any editing mode to insert the line-mode commands. When all the line-mode commands you want to execute are in the buffer, return to the buffer in which you want to use the macro and enter the macro name as a line-mode command.

The following example illustrates this process. The macro searches through the buffer MAIN for the next occurrence of the string "Sales Record". When an occurrence is found, the macro backs up one line and inserts several lines of text.

```
$ EDIT EXAM.TXT [RET]
     1   This file contains sales and shipping records
*DEFINE MACRO DISCLAIM [RET]
*=DISCLAIM [RET]
*INSERT
  FIND ''Sales Record''
  -1
  INSERT
  Sales records include sales
  for which bills have been paid
  and for which bills have been
  outstanding less than 90 days.
  ^Z
^Z
*=main [RET]
```

Note that the first CTRL/Z in the example closes the INSERT in the macro and the second CTRL/Z closes the INSERT that puts the text in the buffer. Now, each time you type the newly created line-mode command DISCLAIM, EDT searches for the string "Sales Record", backs up one line, and inserts the text given.

One advantage of using macros is that you can define large numbers of them and access them with ease. The number of buffers you can maintain (and therefore the number of macros you can access) is limited only by available memory. Another advantage is that you can use macros from any editing mode; the keypad command COMMAND and the nokeypad command EXT (EXTend) allow you to use line-mode commands and macros. You can even include macro names in key definitions with EXT, but they must be the last thing in the key definition.

The disadvantage of using macros is that the macros can only include line-mode commands. This means that you have less flexibility and power than you would in character mode. Only a relatively small number of commands are available in line mode.

Once you have defined macros, you often want to save them so that you can reuse them later. The startup command file is provided for this purpose. Section 4.5.3.3 contains more information on this topic.

4.5.2 Defining Key Functions

You can group nokeypad commands into units (called command strings) and define character editing keys which you press to execute the group of commands as one command. To distinguish them from macros these units are called key definitions. Since you can execute line-mode commands from nokeypad mode, key definitions can include any legal line or nokeypad editing command.

You can create key definitions in either line-editing or keypad-editing mode. In line editing you use the DEFINE KEY command. In keypad editing, you use CTRL/K. When you press CTRL/K, EDT asks you to press the key you want to define. When you have done this, EDT prompts you for the string of nokeypad commands. As you are entering this string, you can use the DELETE key to delete characters, or CTRL/U to erase the entire line and cancel the key definitions.

There are two important differences in these two means of defining keys:

1. With the DEFINE KEY command, you include the key definition within quotation marks ("), while with CTRL/K there are no quotation marks around the key definition.
2. With the DEFINE KEY command, you type in the key definitions using only nokeypad commands, while with CTRL/K, you can either type in the nokeypad command, or press the key that performs that nokeypad command, if there is one. That is, if you want to define a key using CTRL/K that would delete three words, you could either type in D3W to the CTRL/K prompt, or you could press the DEL WORD key three times.

There are several control characters that have specific uses and cannot be redefined:

- CTRL/Z exits to line-editing mode.
- CTRL/S and CTRL/Q are characters used by terminals in buffering text (called XOFF and XON).
- CTRL/O skips over terminal output.
- CTRL/C kills the current EDT command.
- CTRL/H is the same as BACK SPACE, which in EDT moves the cursor to the beginning of the line.
- CTRL/I is the same as TAB.

- CTRL/J is the same as LINE FEED, which in EDT deletes to the beginning of the previous word.
- CTRL/L inserts a form feed, which causes a line printer to move up one page when you print the file.
- CTRL/M is the same as RETURN.

You should not redefine these characters. In some cases the redefinitions would not take effect, while in others your keyboard would not work as you expected (the TAB key would not insert a tab, and so forth).

The GOLD key can precede any key for a key definition. You can define GOLD M or GOLD CTRL/B, for instance.

Key functions are strings of nokeypad commands with some additional rules:

- If you define a key with the DEFINE KEY command, you must enclose the string of nokeypad commands in quotation marks (").
- If you end a string of commands with a period (.), the commands are executed as soon as you press the key. In most cases, you should end the command definition with a period. If you do not end the string of commands with a period, EDT does not execute the commands until you enter the key with ENTER. In other words, the period in the definition enters the command string for you.
- If you enclose a command definition in parentheses, you can give repeat counts by pressing GOLD followed by a number from the keyboard to specify the count before pressing the defined key. If you do not enclose the string in parentheses, the repeat count applies only to the first command in the definition, not the entire definition.
- If you precede a string with a question mark (?), EDT prints the string as a prompt when the command is executed, and accepts input from the terminal in response to the prompt. EDT then substitutes the typed input for the question mark in the command string and executes the string. There is an example of this below.
- When you need to insert a CTRL/Z or a RETURN (CTRL/M) in a key definition, you can enter a circumflex (^) followed by either Z or M. In particular, you can perform the Open Line function (which creates a blank line) with the command sequence ^M-C, which inserts a carriage-return/line-feed combination and backs up one character, placing the cursor on the newly created empty line. If you have a key definition that inserts more than one line of text, the text must be between an I

and a ^Z, but the ^M cannot be before the ^Z. Try the following key definition to see how it works:

```
*DEFINE KEY CONTROL A AS "iName^Z^MiAddress^Z^MTown and State^Z^M."
```

Every time you press CTRL/A while in character mode, the following text is inserted:

```
Name
Address
Town and State
```

The advantage of key definitions is that you can perform complex editing functions by using the powerful nokeypad commands. Because you can deal with entities and the editing operations are done at the character level, you can write key definitions that perform functions impossible to do with macros.

The disadvantage of key definitions is that you must use them in character mode. While this is usually not a restriction, you may want to create a command file that uses EDT to edit text; however, you cannot use the key definitions you have created because the command file must use line editing. That is, you cannot do character editing except on a terminal.

Just as with macros, you will probably want to save the most useful key definitions you create. You can do so with a startup command file. See Section 4.5.3.3 for more information on startup command files.

The following string of nokeypad commands does a global query replace operation from the cursor position to the end of the buffer. That is, the definition prompts you for a string to find and another to replace the first. The macro then searches for an occurrence of the first string and prints the line containing the string. If you want that occurrence replaced, you enter Y; if not, you enter N.

There are two ways to define this key. From line mode, use the DEFINE KEY command:

```
*DEFINE KEY CONTROL A AS "EXT S/?'Replace:'/?' With:'/REST/QUERY."
```

From character mode, press CTRL/K. This produces the prompt **Press the key you wish to define**. Press CTRL/A. This is followed by the prompt **Now enter the definition terminated by ENTER**. Type in the definition, without quotation marks, but with the period (.) at the end.

You have defined a key to use the line-mode command SUBSTITUTE with the slash character (/) as a delimiter. The question marks tell EDT to request input for the strings to delete and insert. When you press the key to which this definition is assigned, EDT prints Replace: at the bottom of your terminal. When you enter a string (it must not contain /, the delimiter character) and press ENTER, EDT prompts **With:** to the right of the string you entered. When you enter the string to be inserted and press ENTER, EDT searches for the first string, displays the line in which it finds the string, and asks you if you want to replace the first string with the second. The legal replies are described in Section 4.4.

This key definition prompts you for a procedure name and places your cursor at the procedure heading for that procedure.

```
BR ADV 'PROCEDURE ?'Procedure name: ''.
```

The BR command places the cursor at the beginning of the current buffer. ADV sets the current direction to forward, and the quoted string moves the cursor to the first occurrence of the string. The question mark tells EDT to print the string in the inner quotes as a prompt and accept text to replace the inner quoted string. If, for example, you type the string "Write_Output" in response to the prompt **Procedure name:**, EDT actually executes the following command:

```
BR ADV 'PROCEDURE Write_Output'.
```

4.5.3 Creating a Personal Editing Environment

EDT provides many features for creating a personal editing environment. You can change the appearance of your screen display, the behavior of lines of text as you insert them, and the mode in which you are working. The following sections discuss some of the SET and SHOW commands provided by EDT and how to use these commands and indirect command files to create a personal startup command file. See the *EDT Editor Manual*, which is available separately, for descriptions of all the available SET commands.

4.5.3.1 Using SET Commands

You can use the SET commands to change the way EDT works. Some SET commands affect the display of text on your screen. For example, you can specify the number of lines you want displayed and whether or not the lines have line numbers (in line mode). You can also specify that EDT make an exact search. See the following SET commands:

```
SET LINES number
SET [NO]NUMBERS
SET SEARCH EXACT
```

By default, your screen contains 22 lines. If you want to decrease the screen display to 5 lines, enter the command SET LINES 5. If you are editing at slow data rates, you can increase your editing speed by decreasing the number of lines displayed on your screen.

When you are working in line mode, EDT displays line numbers by default. If you do not want the numbers to appear on the screen or paper, enter the command SET NONUMBERS.

By default, when you press the GOLD key followed by the FIND key, EDT searches for the specified string, disregarding the case of letters. For example, if you enter the search string "course", EDT will find every occurrence of the word (for example, "course", "Course", "COURSE"). But, when you enter SET SEARCH EXACT, EDT will only find occurrences of the word that exactly match the specified string ("course").

You can use the following line mode commands to determine which mode you enter:

```
SET KEYPAD
SET NOKEYPAD
```

In a start up command file, you would use the following SET commands to determine which mode you enter:

```
SET MODE LINE
SET MODE CHANGE
```

For example, to enter nokeypad mode from keypad mode, enter the following command after pressing GOLDCOMMAND:

```
Command: SET NOKEYPAD
```

One SET command, SET QUIET, even controls the sound of your terminal. You can use this command to suppress the terminal bell that signals an error.

4.5.3.2 Using SHOW Commands to See What Is Set

EDT provides the SHOW commands to enable you to see what is set and what is not set. For every SET command there is a SHOW command. If you want to see the number of lines on your screen, enter the SHOW LINES command. If you want to see whether EDT is performing an exact search, enter the SHOW SEARCH command.

For example, if you want to check the number of lines displayed on your screen, enter the following command:

```
*SHOW LINES
22
```

Or, if you want to see whether line numbers will be displayed in line mode, enter the following command:

```
*SHOW NUMBERS
numbers
```

Table 4–8 lists the SET commands discussed so far with their corresponding SHOW commands:

Table 4–8: SET and SHOW Commands For Line Mode

SET Command	SHOW Command
SET KEYPAD	SHOW KEYPAD
SET NOKEYPAD	
SET LINES	SHOW LINES
SET MODE LINE	SHOW MODE
SET MODE CHANGE	
SET NUMBERS	SHOW NUMBERS
SET NONUMBERS	
SET QUIET	SHOW QUIET
SET NOQUIET	
SET SCREEN	SHOW SCREEN
SET TRUNCATE	SHOW TRUNCATE
SET NOTRUNCATE	

Table 4-8 (Cont.): SET and SHOW Commands For Line Mode

SET Command	SHOW Command
SET WRAP	SHOW WRAP
SET NOWRAP	

For a complete list of the SHOW commands provided by EDT see the *EDT Editor Manual,* which is available separately.

4.5.3.3 Startup Command Files

EDT provides the ability to store macros, key definitions, and editor control commands in a file to be read and executed each time you start EDT. This file is called the EDTINI file (for EDT initialization). The more common name is startup command file. If you have a file in your default directory called EDTINI.EDT, EDT reads the file when it starts.

A startup command file can have any file specification. If it is called EDTINI.EDT it is executed every time you invoke EDT. If it has some other name, it must be called explicitly by using the /COMMAND qualifier to the EDIT command either interactively or in an indirect command file. The default file type for EDT startup command files is .EDT. For example:

```
$ EDIT/COMMAND:SCAN TIZZY.TXT RET
```

EDT executes the file called SCAN.EDT and then is ready to edit TIZZY.TXT.

Following is a sample startup command file. Note that you can include comments in the file by preceding them with an exclamation point (!). The comments follow the key definition they explain.

```
DEFINE KEY CONTROL A AS 'BACK C DC ADV C UNDC.'
 ! CTRL/A transposes the two characters to the left of the cursor.
DEFINE KEY CONTROL E AS "EXT CHANGE=?'To buffer: '."
 ! CTRL/E moves you from buffer to buffer.
DEFINE KEY CONTROL V AS "I~?'Mark as: '~^Z."
 ! CTRL/V marks your location in the text so you can find your
 ! way back to it with CTRL/B.
DEFINE KEY CONTROL B AS "BR ADV S/~?'Find mark: '~//."
 ! CTRL/B finds the location marked by CTRL/V.
DEFINE KEY CONTROL D AS "CUT'?'Delete to: ''."
 ! CTRL/D deletes to the target string you type in. Text is
 ! removed to PASTE buffer and can be pasted back.
DEFINE KEY CONTROL G AS "7ASC."
 ! CTRL/G inserts a bell, beep, or buzz in your text. When you
 ! type the file, you hear the sound.
DEFINE KEY CONTROL R AS "REF."
 ! CTRL/R refreshes the screen.
DEFINE KEY CONTROL F AS "EXT WRITE ?'Write file: '."
 ! CTRL/F writes the current buffer out as a file.
DEFINE KEY CONTROL T AS "EXT INCLUDE ?'Read file: '."
 ! CTRL/T reads a file in from outside EDT to the current position.
DEFINE KEY CONTROL X AS "EXT EX."
 ! CTRL/X causes an EXIT from change mode.
DEFINE KEY GOLD 10 AS "EXT HELP ?'Help on topic: '."
 ! GOLD PF2 prompts for line-mode help.
SET MODE CHANGE
 ! This command puts EDT directly into CHANGE mode upon startup.
SET QUIET
 ! This command eliminates the beep that normally comes with
 ! EDT error messages.
SET WRAP 70
 ! This command causes automatic wrapping at column 70. You
 ! must SET WRAP to use the FILL key (GOLD 8) on a Select Range.
```

Try out these key definitions interactively, using the line-mode command DEFINE KEY or the character-mode CTRL/K. Almost all EDT startup files for use on video terminals include the command SET MODE CHANGE.

Several of the keys defined in this example are defined differently by EDT. Press the HELP key while in character mode for a list of keys defined by EDT. You can usually redefine EDT keys that you do not use, but some keys cannot normally be defined, CTRL/K, for instance. If you want to define one of these keys, but also keep the normal EDT function as well, you can DEFINE KEY GOLD CONTROLS K AS "...". Then, if you wish to use the defined key, press GOLD CTRL/K.

You can also use GOLD to redefine most letter keys, as in the command DEFINE KEY GOLD D AS "...".

Always try out key definitions with CTRL/K before putting them in a startup command file. EDT informs you of undefinable keys. If the CTRL/K does define the key, then press it to make sure it does what you meant it to do.

To define keys on the keypad, use the key number, or GOLD and the number. The keypad keys without numbers printed on them have the numbers as listed in Table 4–9:

Table 4–9: Key Definitions for EDT

Keypad Key	Key Number	Keypad Key	Key Number
UP-ARROW	12	.(dot)	16
DOWN-ARROW	13	0	0
RIGHT-ARROW	14	1	1
LEFT-ARROW	15	2	2
PF1	GOLD	3	3
PF2	10	4	4
PF3	11	5	5
PF4	17	6	6
_(underscore)	18	7	7
,(comma)	19	8	8
ENTER	21	9	9

For even more information on key definition, see the *EDT Editor Manual,* which is available separately.

4.5.3.4 Using Indirect Command Files with EDT

You can write an indirect command file to combine the features of macro definition, key definition, and EDT startup command files to help you create a personal editing environment.

Here is an indirect file:

```
.ENABLE SUBSTITUTION
.ASKS FI File
EDIT/COMMAND:INI 'FI'
.ASK PU Purge it
.IFT PUT PURGE 'FI'
```

This command file enables you to use a specially named EDT startup file (INI.EDT) instead of the default (EDTINI.EDT).

The startup file can contain key definitions.

You can also include macro definitions in a startup file. Keep in mind that macro definitions are stored in a separate buffer. You can insert the macro definition as part of your startup file, or you can do as shown in this example:

```
    .
    .
    .
DEFINE MACRO PELF
 ! Creates buffer PELF and enters name in
 ! line-mode command list.
INCLUDE PELF.FIL =PELF
 ! Finds file PELF.FIL, which includes
 ! the macro definition.
FIND =MAIN
 ! Takes you back to MAIN buffer so you won't
 ! start out in PELF.
    .
    .
    .
```

You can do as many of these as you need for special applications of EDT. Many EDT users have several startup files, one for programming, one for text editing, and one for special formats, such as memos.

See Chapter 9 for more information on the Indirect Command Processor.

4.5.4 Structured Tabs

Most structured programming languages use tabs to indicate nested blocks of code and other structures. EDT provides a structured tab facility to assist you in this process. Skip this section if you are not writing structured code.

When you use the structured tab facility, you define the space between tabs with the SET TAB command. You can execute this command in line mode or with a startup command file. (See Section 4.5.3.3 for details on the startup command file.) For instance, SET TAB 4 tells EDT to put four spaces between two characters that have a tab between them. Once you have executed this command, you can press the TAB key to get four spaces. Pressing TAB again positions the cursor eight characters to the right, and so forth. Thus, you can set the tab stops with the SET TAB command.

EDT keeps track of the current level of indentation using its indentation-level counter. You can increment, decrement, and set the counter with the appropriate commands (described in the following paragraphs) When you type a TAB, EDT multiplies the value you gave with the SET TAB command by the current value of the indentation-level counter to determine the number of spaces and tabs to insert. Suppose, for example, you press TAB before a WHILE statement; the WHILE statement starts at character position 4 (assuming you have entered the command SET TAB 4, as described in the previous paragraph). EDT sets the indentation-level counter to 1. You can now increment the counter, making its value 2, and when you press a TAB at the beginning of the next line, EDT inserts eight spaces. When you have introduced all the code that goes under the WHILE statement, you can decrement the counter, after which pressing TAB causes only four spaces to be inserted. Thus, code that is part of the WHILE statement is indented eight spaces, and code following the WHILE is indented only four spaces.

Three commands affect the indentation-level counter. You can increment the level count with CTRL/E in keypad mode or the TI (Tab Increment) command in nokeypad mode. CTRL/D or TD (Tab Decrement) decrements the level count. CTRL/A and TC (Tab Compute) cause EDT to compute the indentation level that corresponds to the cursor position and store it as the current indentation level. To execute this command, your cursor must be positioned to a column that is an even multiple of the SET TAB value; otherwise you get an error message.

You can use the Tab Compute command to help in editing a program that already exists. If you want to type some code that should be indented twelve spaces, for example, you can move the cursor to character position twelve and execute the Tab Compute command. Assuming that you have previously executed the SET TAB command with a value of 4, EDT computes that the current indentation-level counter value should be 3. You can then proceed to type TAB at the beginning of a line to get twelve spaces in front of the line, just as required.

EDT also has a command to move a region of text left or right. CTRL/T or TABJ (Tab ADJust) moves the Select Range to the right if the repeat count is positive, or to the left if the count is negative. EDT inserts or deletes a number of tabs and spaces corresponding to the repeat count times the SET TAB value.

4.5.5 More on EDT Buffers

This section briefly explains the more advanced features of EDT buffers. These features include the following:

- Creating buffers
- Moving between buffers
- Copying text from a file into a buffer
- Copying text from one buffer to another buffer
- Copying text from a buffer to a file
- Using buffers

For more information on buffers, see the *EDT Editor Manual,* which is available separately.

4.5.5.1 How To Create and Move Between Buffers

To create a buffer, type the Find command followed by an equals sign (=) and then the buffer name. A buffer name must be between 1 and 30 alphanumeric characters, but cannot start with a number. The only punctuation character you can use in a buffer name is an underscore (_).

For example, to create a buffer named OSCAR while in line mode, enter the following command line after the asterisk prompt (*):

```
*FIND=OSCAR RET
```

EDT creates the buffer OSCAR and moves you into that buffer. If you press RETURN, you will see the [EOB] symbol followed by an asterisk prompt (*), indicating that both the buffer OSCAR is empty and EDT is ready for your commands to affect this buffer. You now can insert and edit text as you did in the MAIN buffer.

To create a buffer from keypad mode, press the COMMAND keypad function (GOLD 7). When EDT prompts you with "Command:," type Find followed by an equals sign (=) and then the buffer name of your choice. Terminate the command by pressing the keypad ENTER key. EDT creates the specified buffer, then moves you into it. The screen clears except for the [EOB] symbol, indicating that the current buffer is empty. You can now begin entering and editing text just as you did in the MAIN buffer.

To create a buffer in nokeypad mode, type EXT followed by the line-mode command indicated earlier in this section.

To return your previous location in the MAIN buffer, use the command Find=MAIN. Include the period after the buffer name. If you omit the period, you will return to the top of the MAIN buffer. When you leave the alternate buffer, it remains intact until you exit from EDT. Unless you command EDT to save the text of a buffer other than MAIN, that text is destroyed when you leave EDT. (See Section 4.5.5.4 on how to save the contents of a buffer by copying it into a file.)

Note that you can return to a buffer that you previously created. Just use the Find=buffername command, terminating this command with a period if you want to return to the exact location in *buffername* that you had previously left.

The line-mode command SHOW BUFFERS displays a list of the existing buffers and some information about the contents of each. An equals sign (=) appears in the SHOW BUFFERS display next to the name of the current buffer.

4.5.5.2 How To Copy Text from a File Into a Buffer

To copy text from a file located outside of EDT into an EDT buffer, use the INClude command. While in line mode, for example, you type the following command to copy the contents of a file named OUTSIDER.DAT into your MAIN buffer:

```
*INCLUDE OUTSIDER.DAT =MAIN  RET
```

The text of OUTSIDER.DAT is copied above the current line in the MAIN buffer. If you do not specify a buffer name, EDT copies the text above the current line in your current buffer.

Note that OUTSIDER.DAT is located in your own directory. You can copy a file from another directory into an EDT buffer, providing you have access to that file. See Chapter 5 for information about file specification and file protection.

To copy text from a file into buffer while in keypad mode, you execute the COMMAND keypad function (GOLD 7), then type after the "Command:" prompt:

```
Command: INCLUDE OUTSIDER.DAT =MAIN
```

You press the keypad ENTER key to terminate this command.

In nokeypad mode, type EXT followed by the line-mode command indicated earlier in this section.

4.5.5.3 How To Copy Text from One Buffer to Another Buffer

To copy text from one buffer to another while in line mode, you use the COPY command. For example, to copy the contents of a buffer named OSCAR to a buffer named YORICK, type:

```
*COPY =OSCAR TO =YORICK  RET
```

When you complete this operation, the entire contents of the buffer OSCAR is copied to the top of the buffer YORICK, and you are located in the buffer YORICK.

Using the COpy command, you also can copy a portion of one buffer to a specific location in another buffer. For example:

```
*COPY =OSCAR 5:20 TO =MAIN 45  RET
```

This command copies lines 5 through 20 of the buffer OSCAR to the location immediately above line 45 of the MAIN buffer. Your current buffer is MAIN at the end of this operation.

When you are in keypad mode, you also can use the COpy command to copy the entire contents of one buffer into another. First press the COMMAND keypad function (GOLD 7), then type after the "Command:" prompt:

```
Command: COPY =OSCAR TO =MAIN
```

Press the keypad ENTER key to terminate this command.

However, if you are going to copy a portion of one buffer into a specific location in another buffer, use Find command along with the CUT and PASTE keypad functions. For example, to move a paragraph from buffer OSCAR to a particular location in MAIN:

1. Move to buffer OSCAR, using the FIND=OSCAR command.
2. Move your cursor to the desired paragraph.
3. Use the CUT keypad function (6) to select this paragraph from buffer OSCAR.
4. Move back to buffer MAIN, using the FIND=MAIN command. (To return to your previous position in MAIN, type a period (.) after MAIN.)
5. Move your cursor to the position in MAIN that you want the paragraph to appear.
6. Use the PASTE keypad function (GOLD 6) to copy the paragraph into the buffer MAIN. The text is copied immediately in front of the cursor position.

See Section 4.3.6 for a detailed description of how to use the CUT and PASTE keypad functions.

To copy text from one buffer to another while in nokeypad mode, type EXT followed by the line-mode command indicated earlier in this section.

4.5.5.4 How To Copy Text from a Buffer to a File

To copy text from a buffer to a file, use the Write command. You can use this command to copy text to a new file without affecting your editing session.

The Write command followed by a file name copies the entire contents of your current buffer into the specified file. You can also copy only a portion of your current buffer into a file. For example, the following line-mode command puts a copy of lines 23 through the end of the current buffer into a file that you name EMT.DAT.

```
*WRITE EMT.DAT 23:END [RET]
```

In keypad mode, to copy only a portion of your current buffer into a file called EMT.DAT:

1. Use the CUT keypad function (6) to place the selected text for the new file in the PASTE buffer.
2. Execute the COMMAND keypad function (GOLD 7), then type after the "Command:" prompt:

   ```
   Command: WRITE EMT.DAT =PASTE
   ```

   This Write command copies the contents of the PASTE buffer into the new file EMT.DAT.
3. Use the PASTE keypad function (GOLD 6) to copy the text back into your current buffer, if you want to keep it in your current file also.

In nokeypad mode, to copy text from a buffer into a file, type EXT followed by the line-mode command indicated earlier in this section.

4.5.5.5 Ways to Use Buffers

Here are some ways to use EDT buffers:

1. Suppose you have to send a letter to 10 people, whose names and addresses are in a file. The letter is basically the same for each person but requires some minor editing to personalize the contents. You might do the following:
   - Invoke EDT to type the basic letter in the MAIN buffer.
   - Use the Find command to create a separate buffer and move there.
   - Use the INClude command to copy the file with the address list into this separate buffer.

- Use the CUT keypad function to select the address from the list that you want to use in your letter. (See Section 4.3.6 for information on how to move text from one location to another using the CUT and PASTE keypad functions.)
- Move back to your MAIN buffer, which contains your basic letter.
- Use the PASTE keypad function to add the address to the letter.
- Use the Write command to create a file containing this copy of the letter.

You are now ready for the second letter. Delete the old address and, using the preceding procedure, move the next one into place. Edit the letter so that it reflects this new recipient and then use the Write command to copy that version to another file. Repeat the process until all 10 letter are finished.

2. When you write computer programs, you often use elements from other programs in the one you are currently creating. You might do the following:
   - Invoke EDT to type the new program in the MAIN buffer
   - Use the Find command to create a separate buffer and move there
   - Use the INClude command to copy into this buffer the file containing the other program
   - Use the CUT keypad function to select the portion of this other program that you want to include in your new program
   - Move back to your MAIN buffer, which contains your new program
   - Use the PASTE keypad function to move the section of the other program into your new program
3. If you are writing a large computer program with several modules, you may want to create each module in a separate buffer. When you are ready to assemble your modules into one program, you can use either the COpy command or the Find command along with the CUT and PASTE keypad functions to move your text from one buffer into another.
4. If you are in the middle of an EDT session and need to get some information or text from another file, you can create another buffer and use the INClude command to copy this file there. Once you find

the item you need, you can move back to your original buffer and resume editing there.

4.6 EDT Summary

This section summarizes EDT. There are brief descriptions of range specifications, editor control commands, and line and character editing commands. The editing entities on which character editing commands can operate are also included. For a more detailed description of the commands and specifications presented here, see the *EDT Editor Manual,* which is available separately.

Command names are given in full with the abbreviated form printed in UPPERCASE. Square brackets ([]) enclose an optional parameter or qualifier; in these cases, the default value is specified in the text accompanying the command. A vertical bar (|) separates options; two forms with a vertical bar between them are mutually exclusive. You can choose one or the other, or, in some cases, neither. Lowercase words indicate information you supply, such as file specifications, beginning line numbers, range specifications, and so forth.

Remember that any line-mode command can be included in a key definition using EXT and that any line-mode command can be entered from character mode by using GOLD 7 and typing the command in response to the **Command:** prompt.

4.6.1 Range Specifications

Range specifications can designate single lines, contiguous ranges, or noncontiguous ranges. You specify a range as shown in Table 4–11.

Table 4-11: Range Specification

Range Specification	Description
Single line ranges	
	The current line—the cursor position in character editing. **Example:** *TYPE . Types the current line.
number	The line specified by the number. **Example:** *1536 Line pointer moves to line 1536.
'string' | "string"	The next line containing the string (if preceded by a minus sign, the preceding line containing the string). Two quotation marks with nothing between them, as "", default to the last string entered. **Example:** *'False Aralia' Line pointer moves to next line containing "False Aralia".
[range] + [number]	The line specified by adding the number to the range (if you use a minus sign instead of a plus sign, the line specified by subtracting the number from the range). The range defaults to the current line, number defaults to 1. **Examples:** *. + 25 Line pointer moves down 25 lines *+25 Line pointer moves down 25 lines *237–25 Line pointer moves first to 237 and then up 25 lines
BEGIN	The first line in the buffer. **Example:** *FIND BEGIN Line pointer moves to beginning of the buffer.

Table 4-11 (Cont.): Range Specification

Range Specification	Description
END	An empty line following the last line in the buffer. **Example:** *FIND END-1 Line pointer moves to the last line in the buffer.
LAST	The line position in the most recent text buffer before entering the current buffer. **Example:** *FIND LAST EDT switches back to the last position in the previous buffer. If you use this command in a key definition, such as DEFINE KEY GOLD L AS "FIND LAST." and then press GOLDL while in character mode, you can jump back and forth between two buffers with ease.
ORIGINAL number	The line specified by the number in the buffer MAIN before any edits changed line numbers in this editing session. **Example:** *TYPE ORIGINAL 15 Types the line originally numbered 15, regardless of how many lines you've added to the file.
Contiguous line ranges	
[range-1] : [range-2]	The lines between range-1 and range-2. Both ranges must be single lines. If either is omitted the current line is the default. You can use either : or THRU in this expression. **Example:** *15:36 Lists lines 15 through 36 on your terminal.

Table 4-11 (Cont.): Range Specification

Range Specification	Description
[range] # number	The specified number of lines starting with range, which must be a single line; if range is omitted it defaults to the current line (you can also use FOR instead of #). **Example:** *TYPE 13 5 Types line 13 and the next four lines, five lines in all.
BEFORE	All lines in the buffer preceding the current line. **Example:** *DELETE BEFORE Deletes everything in the buffer above the line pointer.
REST	All lines in the buffer after and including the current one. **Example:** *SUBSTITUTE/Molly/Dolly/REST Performs the substitution in the current line and the rest of the buffer.
WHOLE	All lines in the current buffer. **Example:** *TYPE WHOLE Lists the entire buffer.
Noncontiguous line ranges	
[range-1,range-2,...]	All lines specified by range-1, range-2, and so on. Each range must be a single line (you can also use AND instead of,). **Example:** *12, 16, 14, Types lines 12, 16, and 14, in that order. **Example:** *COPY 12,2 TO END Copies line 12 and line 2, in that order, to the end of the buffer.

Table 4-11 (Cont.): Range Specification

Range Specification	Description
[range]ALL 'string'	All lines in the range containing the string. If you do not supply a range the default is all lines in the current buffer containing the string. **Example:** *COPY ALL 'log' to = BOY Copies all lines containing the string log to a buffer named BOY, which becomes the current buffer.
Ranges that include buffer names	
=buffer [range]	The specified range in the specified buffer. The range defaults to the entire buffer, with the cursor placed at the top of the buffer (you can also use BUFFER instead of =). Whenever you name a buffer with the = or BUFFER, that buffer becomes the current buffer. In the output from the SHOW BUFFER command, the current buffer is indicated by the =.

4.6.2 Editor Control Commands

Editor control commands affect the way in which EDT does its work and the way in which it displays the results. The following list shows the minimum abbreviation for the commands in UPPERCASE. You can enter anything between the minimum abbreviation and the entire keyword when you issue a command.

Change [[=buffer] range]

CHANGE places you in character editing mode, either keypad or nokeypad according to the last SET KEYPAD/NOKEYPAD command, or according to the terminal model. The default is keypad editing for VT100- and VT200-series terminals. For other terminals, the default is nokeypad. The range can include a buffer name and a single line specification where the cursor is positioned in character editing mode. You can express this single-line range as the dot (.), meaning "where I last was in this buffer," or as a line number, or as a quoted string.

EXit [filespec][/SEQuence[[:initial:increment]][[/SAve]]]

EXIT ends an editing session and writes the contents of buffer MAIN to the file whose name you specify (the file specification defaults to the input file specification). The /SEQUENCE qualifier requests EDT to include line numbers in the output file. The initial value is the starting number and the increment value is the increment between numbers. The /SAVE qualifier tells EDT to save the journal file (called input-file-name.JOU). Normally, the journal file is deleted after you successfully exit from EDT.

Help topic subtopic

HELP displays information on the requested subject. The values for "topic" and "subtopic" can be either alphanumeric strings or the wildcard symbol (*). HELP with no qualifiers produces a list of all topics for which EDT is prepared to give help.

Quit[/SAve]

QUIT ends an editing session without saving the text in any buffer. The /SAVE qualifier requests EDT to save the journal file (called input-file-name.JOU).

SEt CAse {Upper | Lower | None}

SET CASE tells EDT to flag any characters of the specified type by printing an apostrophe before the character. This command is mainly used on terminals that do not display both upper- and lowercase.

SEt Cursor top:bottom

SET CURSOR defines the points at which EDT scrolls the text on a display terminal in character editing. When you move the cursor outside the range specified by top:bottom (where top and bottom are line numbers on the screen), EDT scrolls the text up or down to accommodate the cursor movement. The default setting for the cursor is 7:14. The screen does not start scrolling until you move the cursor up to line 7 or down to line 14. If you issue the command SET CURSOR 14:14, the screen starts scrolling whenever you start to move off line 14.

SEt ENTity {Word | Sentence | PAGe | PARagraph} 'string'

SET ENTITY allows you to define character strings to be recognized as entity delimiters. WORD and SENTENCE entities end with any one of the string of characters you specify. PAGE and PARAGRAPH end with the entire string. Default delimiters are space, LF, TAB, FF, and RET for WORD; period, question mark, and exclamation point for SENTENCE; FF for PAGE; and RET RET for PARAGRAPH. Although all these entities are defined, there are no EDT commands that affect sentences or paragraphs. There are several commands that affect words: DEL W, UND W, WORD. There is one EDT command that affects a page: PAGE. The SET ENTITY command makes it possible to define keys (or enter nokeypad commands) that affect all these entities, but you have to do it yourself. If you have a text formatting program, you may want to use SET ENTITY to define paragraphs and pages using the same rules as the text formatter.

SEt Keypad | NOKeypad

SET KEYPAD/NOKEYPAD tells EDT whether you wish to enter keypad or nokeypad editing mode when you issue the CHANGE command or the SET MODE CHANGE command from your startup command file. See Section 4.5.3.3 for more on startup command files. The default is keypad editing for VT100- and VT200-series terminals and nokeypad for all others. If your terminal has a keypad and is not one of the DIGITAL models, you can probably include SET KEYPAD in your startup file and it will probably work. If this is your situation, you should check to make sure the key definitions for your terminal match what is shown in the EDT help files. You should also check your terminal manual.

SEt Lines number

SET LINES tells EDT how many lines to display on your screen in character-editing modes. You can speed editing tasks on slow terminals by reducing the number of lines displayed. The default number of lines is 22. If you are editing on a slow terminal, try SET LINES 12. This reduces the time it takes to change the screen when you edit something. If you do change this setting, you should also SET CURSOR 0:8.

SEt Mode {Line | Change}

The SET MODE command is used in the startup file (Section 4.5.3.3) to control the mode in which EDT starts editing. If you put SET MODE CHANGE in your startup file, you go directly into character editing when you start EDT. The default for EDT is line mode.

SEt NUmbers | NONumbers

SET NONUMBERS tells EDT not to display line numbers in line-editing mode. The default is NUMBERS. EDT displays line numbers along with text in line editing.

SEt Quiet | NOQuiet

SET QUIET tells EDT not to ring the terminal bell when an error occurs in character editing. SET NOQUIET is the default. SET QUIET is commonly included in startup files.

SEt SCreen width

SET SCREEN defines the number of characters EDT displays on each line. What happens to the remaining characters on the line is determined by the TRUNCATE/NOTRUNCATE setting (see SET TRUNCATE, below): the extra characters are either wrapped around to the next line, or they are not displayed.

The default screen width is 80 characters. The VT100- and VT200-series terminals can be set to either 80 columns or 132 columns. If you are editing a file that is in 132-column format, you should reset your terminal before entering the EDIT command. See Chapter 3, the description of the SET TERMINAL command, for more information. Once you have done that, then enter the EDIT command and SET LINES 132 to make EDT match up with the file.

Another method of dealing with lines wider than the screen—and EDT lines can be up to 250 characters long—is to use the nokeypad commands SHL (shift left) and SHR (shift right) in a key definition. See *EDT Editor Manual*, which is available separately, for an explanation of these commands. See also EDT help files under HELP CHANGE SUBCOMMAND SHIFT.

SEt SEarch {Begin | End}

SET SEARCH BEGIN/END tells EDT where to leave the cursor after locating a search string. The default is BEGIN. EDT leaves the cursor at the beginning of the search string.

SEt SEarch {BOunded | Unbounded}

SET SEARCH BOUNDED tells EDT to stop searching when it reaches the next page delimiter. This does not work if you do not have any page delimiters in your text. The default is UNBOUNDED; EDT searches the whole buffer in the current direction.

SEt SEarch {General | EXact}

SET SEARCH EXACT tells EDT to accept a match for a search string only if the case matches exactly for each character in the string. GENERAL, which is the default, allows strings to match whatever the case. If you SET CASE EXACT, you can tell the difference between Mongo and mongo. The case setting affects the SUBSTITUTE command and SUBS key as well as searches.

SEt TAb n | NOTab

The SET TAB command, in conjunction with the tab manipulation commands (CTRL/A, CTRL/D, CTRL/E, and CTRL/T in keypad editing; TC, TD, TI, and TADJ in nokeypad editing), allows you to format your text, especially structured programs, very easily and quickly. For a description of how to use structured tabs, see Section 4.5.4.

SEt TErminal{HCPY | VT52 | VT100}
{SCROLL | NOSCROLL}
{EDIT | NOEDIT}
{EIGHTBIT | NOEIGHTBIT}

The SET TERMINAL command overrides the terminal definition that EDT obtains from the operating system. Micro/RSX takes care of all terminal settings for you when you issue the EDIT command. It is rarely necessary to override these settings if you have a DIGITAL terminal. If you have to do so, it may indicate that Micro/RSX does not know your terminal model. Issue a DCL SET TERMINAL/INQUIRE command to make sure your terminal is set properly.

The EDT SET TERMINAL COMMAND offers four choices:

- The **first** choice is between HCPY, VT52, and VT100. HCPY means hardcopy. This is the default for all terminals except VT52s, the VT100-series, and the VT200-series; nokeypad editing is the default. VT52 sets the terminal as a VT52; keypad definitions match the VT52 keypad; keypad editing is the default. VT100 sets the terminal as a VT100- or VT200-series terminal; keypad definitions match the keypads on those terminals; keypad editing is the default.
- The **second** choice is between SCROLL and NOSCROLL. All DIGITAL video terminals default to SCROLL, meaning that they support scrolling regions that EDT can use. Other video terminals default to NOSCROLL; some support scrolling regions and can be set to SCROLL.
- The **third** choice is between EDIT and NOEDIT. VT102 terminals support certain screen editing features; EDIT is the default for VT102 terminals. NOEDIT is the default for all others.
- The **fourth** choice is between EIGHTBIT and NOEIGHTBIT. Terminals that support the DEC Multinational Character Set, such as the VT200-series, default to EIGHTBIT. All others default to NOEIGHTBIT.

SEt {TRuncate | NOTRuncate}

SET NOTRUNCATE tells EDT not to truncate lines that are longer than the current screen width. If you specify NOTRUNCATE, EDT displays the extra characters on subsequent lines. TRUNCATE is the default setting; EDT displays a diamond on the edge of the screen to indicate that the display does not include the entire line.

SEt Verify | NOVerify

SET VERIFY tells EDT to print the text of macros and command files when you execute them. This can be helpful in debugging macros and key definitions, but otherwise you probably can skip it. The default setting is NOVERIFY.

SEt Wrap n | NOWrap

SET WRAP does two things: it causes EDT to insert a RETURN in your file when the word you are typing causes the line length to exceed the value of n. It also tells EDT what to use for the right margin when you execute the FILL command. NOWRAP is the default setting. If you have issued SET WRAP, you can then mark out a Select Range and press FILL (GOLD 8) and all the lines in the Select Range are rearranged to match the value you set.

SHow BUffer

SHOW BUFFER lists the buffers in use and the number of lines of text in each buffer. The current buffer has an equals sign (=) in front of its name. Any buffer for which EDT has not read all the text has an asterisk after the line count, showing that the count for that file may not be correct. EDT sometimes reads in only part of a file when starting up, but it reads in the rest without any explicit action on your part as you work your way down in the file.

SHow CAse

SHOW CASE displays the current case setting—UPPER, LOWER, or NONE. The default is NONE.

SHow Cursor

SHOW CURSOR displays the range the cursor can move in without EDT scrolling the text up or down. The default setting is 7:14.

SHow ENtity {Word | Sentence | PAGe | PARagraph}

SHOW ENTITY displays the delimiter(s) for the specified entity. Default delimiters are LF, TAB, FF, and RET for WORD; period, question mark, and exclamation point for SENTENCE; FF for PAGE; and RET RET for PARAGRAPH.

SHow Key {CONtrol letter | [Gold] number}

SHOW KEY displays the definition of the specified key in character editing, whether you have redefined the key or not.

SHow SCreen

SHOW SCREEN displays the current setting for the width of the screen. The default setting is 80.

SHow SEarch

SHOW SEARCH shows the current search parameters. The default settings are GENERAL, BEGIN, and UNBOUNDED.

SHow TErminal

SHOW TERMINAL displays the type of terminal EDT thinks you are using.

SHow VErsion

SHOW VERSION shows the version of EDT you are using.

4.6.3 Line-Mode Editing Commands

You use line-mode commands for standard line-mode editing, and from character mode using the GOLD 7 key.

The /QUERY qualifier can be used with several commands (COPY, DELETE, MOVE, and SUBSTITUTE). In each case, EDT displays the line it is currently on, and prompts you with a question mark. The legal replies are Y, N, A, and Q. Y (for Yes) tells EDT to perform the operation as requested. N (No) tells EDT not to perform the operation. A (All) allows EDT to perform the indicated operation for the rest of the range. Q (Quit) tells EDT to quit the operation without doing anything further.

CLear buffername

CLEAR deletes the entire contents of a buffer you name, whether you are currently editing in that buffer or not. The buffer no longer exists after you clear it.

COpy range-1 TO [range-2][/Query][/Duplicate:n]

COPY copies the range of lines specified by range-1 to the position specified by range-2 without deleting the original text. Range-2 can be another buffer. The default for range-2 is the current line. If you use the /QUERY qualifier EDT asks you about copying each line of range-1. The responses to the /QUERY command are described at the beginning of this section. The /DUPLICATE qualifier tells EDT to perform the copy a number of times, thereby inserting several copies of the text at range-2.

DEFine Key [CONtrol | Gold] x AS "string"

DEFINE KEY associates a keypad key or a control character with a string of nokeypad commands. If you use CONTROL, x must be a keyboard character other than 0—9, !, %, ', and ". If you use GOLD, or if you do not include GOLD or CONTROL, x must be a number. (For a display of the number associated with each keypad key, enter HELP DEFINE KEY VT100. The VT200-series is the same for the numerical keypad. The numbers for the VT200-series editing keypad are E1 through E6.) The "string" must be a quoted string of nokeypad commands. For more information on constructing these strings, see Section 4.5.2.

DEFine Macro macro-name

DEFINE MACRO assigns a name to a series of commands. Each macro is associated with a buffer in which reside a number of line-mode commands (the macro). When you enter macro-name as a line-mode editing command, these commands are executed. The macro-name can be 1 to 30 characters in length and may include alphabetic and numeric characters and underscores (_). See Section 4.5.1 for more information on defining macros.

Delete [range][/Query]

DELETE tells EDT to delete the lines in the specified range. The range defaults to the current line. If you enter the /QUERY qualifier, EDT asks you about each line before deleting it. The responses are listed at the beginning of this section.

Find range

FIND locates the specified range (which must be a single line) and places the cursor at the beginning of the line.

INClude filespec [range]

INCLUDE inserts the file whose name you specify into the buffer specified by range. The default for range is the current buffer. EDT inserts the file in front of the first line of range.

Insert [range]

INSERT allows you to insert text at the point specified by range, which must be a single line. EDT inserts the text you type in front of the line indicated by range. You terminate the text to be inserted with a CTRL/Z. However, if you wish to insert only one line of text, you can follow the range with a semicolon; any text typed on the same line after the semicolon is inserted at range.

Move [range-1] TO [range-2][/Query]

MOVE moves the text specified by range-1 to the point indicated by range-2. The default for range-1 and range-2 is the current line. Range-2 must be a single line. MOVE deletes the original copy of the text and moves it in front of the point indicated by range-2.

Print filespec [range]

PRINT writes a file called filespec to the line printer, if any. The default for range is the current buffer.

Replace [range]

REPLACE is the equivalent of a DELETE command followed by an INSERT command. EDT deletes the text specified by range and allows you to insert text to replace it. The default for range is the current line. Replacement text is inserted at the point from which the deleted text was taken. You can insert a single line of text by following the range with a semicolon. EDT inserts any text between the semicolon and the RETURN after deleting the text indicated by range.

RESequence [range][/SEquence[[:initial:increment]]]

RESEQUENCE assigns new sequence numbers to the lines in the range. Range must specify a contiguous set of lines, and the default for range is the entire current buffer. If you use the /SEQUENCE qualifier, EDT starts the sequencing with the value of initial and adds the value of increment to get the next line number. If you omit the /SEQUENCE qualifier, initial defaults to the first line number of range and increment defaults to 1.

Substitute /string-1/string-2[/][range][/Brief[[:n]][[/NOType]][[/Query]]]

SUBSTITUTE replaces occurrences of string-1 with string-2. Range specifies where EDT may look for string-1. If you omit range, it defaults to the current line. (You can use any character as a string delimiter provided you use the same delimiter in all places.) EDT displays the line after it makes the substitution. The /BRIEF qualifier tells EDT to display only n characters of the line. The /NOTYPE qualifier tells EDT not to display the line. The /QUERY qualifier tells EDT to ask before doing each substitution. The possible responses are described at the beginning of this section.

Notice that the slash (/) is part of the command. If you wish to use SUBSTITUTE on a string that includes a /, you can replace the / with another delimiter chosen from the shifted number keys on the keyboard. For example, the following command substitutes Micro/RSX (with /) for MicroPDP–11:

```
*SUBSTITUTE$MicroPDP-11$Micro/RSX$WHOLE/QUERY
```

Notice that the $ is not substituted for the / in front of QUERY.

[Substitute] Next [/string-1/string-2[[/]]]

SUBSTITUTE NEXT replaces the next occurrence of string-1 with string-2. The command is especially useful after you have used the SUBSTITUTE command, since string-1 and string-2 default to the strings specified in your last SUBSTITUTE command. When EDT makes a substitution on a line, that line becomes the current line. The same rule about the / holds.

[Type][range][/BRief[[:n]][[/STay]]]

TYPE is the default line-mode command and need not be specified. TYPE displays the specified range of lines. Range can be a single line or multiple lines, and the multiple lines need not be contiguous. If you include the /BRIEF qualifier, EDT displays only the first n characters from each line (n defaults to 10). If you specify the /STAY qualifier, EDT does not move the current line.

Write filespec [range][/SEQuence [[:initial:increment]]]

WRITE writes the text specified by range to a file called filespec. If you use the /SEQUENCE qualifier, EDT includes the line numbers, starting with initial and increasing by increment, in a fixed field in the output file. If you omit the /SEQUENCE qualifier, EDT uses the line numbers it obtained from the input file. In other words, use this command to create a file without leaving EDT.

4.6.4 Editing Entities

Some character editing commands operate on pieces of text that EDT recognizes by special names. These pieces are called entities of text. The names are used in nokeypad commands, but some of the entities are used in keypad commands as well. Table 4–12 summarizes the EDT entities.

Table 4–12: Editing Entities

Entity	Description
C	A single character, which may be any of the alphabetic, numeric, or special keys on the keyboard. Control characters, such as CTRL/M, are considered single characters.
W	A word, which is a string of characters delimited by one of a set of delimiter characters. The default delimiters are space, RET, TAB, and FF, but you can change these with the SET ENTITY command.
BW	The string of characters from the cursor position to the beginning of the word; if the cursor is at the beginning of a word, BW is the previous word.
EW	The string of characters from the cursor to the end of the word, including the character at the cursor position.
L	A single line of text.
BL	The string of characters from the cursor to the beginning of the line, or, the previous line, if the cursor is at the beginning of a line.
EL	The string of characters from the cursor to the end of the line, including the character at the cursor position.
NL	The string of characters from the cursor position to the beginning of the next line.
PAR	A string of characters between paragraph delimiters; the default delimiter is a pair of RETURNs, but you can change these with the SET ENTITY command.
BPAR	A string of characters from the cursor to the beginning of the paragraph, or the previous paragraph, if the cursor is at the beginning of a paragraph.
EPAR	A string of characters from the cursor to the end of the paragraph, not including the paragraph delimiter, but including the character at the cursor position.
SEN	A string of characters between sentence delimiters; the default delimiters are period, question mark, and exclamation point, but you can change these with the SET ENTITY command.

Table 4–12 (Cont.): Editing Entities

Entity	Description
BSEN	A string of characters from the cursor to the beginning of the sentence.
ESEN	A string of characters from the cursor to the end of the sentence, not including the sentence delimiter, but including the character at the cursor position.
PAGE	The text between page delimiters, including the second delimiter; the default delimiter for a page is a form feed (CTRL/L), but you can change this with the SET ENTITY command.
BPAGE	The text from the cursor to the beginning of the page, or the previous page, if the cursor is at the beginning of the page.
EPAGE	The text from the cursor to the end of the page, not including the page delimiter, but including the character at the cursor position.
BR	The text between the cursor and the beginning of the current buffer.
ER	The text between the cursor and the end of the current buffer.
V	The text between the cursor and the same column position on the next line, or the end of the next line, if the next line does not contain enough characters to reach the same column.
'xyz'	A string of characters that is used in search operations; when you use this entity with the D (delete), CUT, or R (replace) functions, the entity includes the text between the cursor and the string.
SR	The text in the Select Range, which is everything between the cursor position and the place in the buffer where the SEL command was last executed; if you have entered no select command (or you have canceled the last one) SR defaults to the search string, or to a single character in the current direction for the CHGC (change case) command.

4.6.5 Keypad Commands

This section provides a brief description of the function of each keypad command, in Table 4–13. For further information on these commands, use the keypad HELP key.

Table 4-13: Keypad Commands

Command	Function
ADVANCE	Sets the current direction to be forward (toward the end of the current buffer). This is the default setting, and it remains in effect until you give the BACKUP command. This is keypad key 4.
APPEND	Deletes the Select Range and stores it at the end of the paste buffer, leaving the previous contents of the paste buffer untouched. This is keypad key 9.
BACKUP	Sets the current direction to be backward (toward the beginning of the current buffer). This setting remains in effect until you give the ADVANCE command. It is quite confusing to be pointing back when you think you are pointing forward. All commands implying a direction are affected. This is keypad key 5.
BLINE	Moves the cursor over a BL in the current direction.
BOTTOM	Places the cursor at the end of the current buffer. This is keypad key GOLD 4.
CHNGCASE	Changes the case of letters in the Select Range (UPPERCASE letters become lowercase and vice versa). If there is no Select Range this command operates on one character in the current direction. This is keypad key GOLD 1.
COMMAND	Prompts you to enter a line-editing or editor-control command. This is GOLD 7 on the keypad.
CUT	Deletes the Select Range from the current buffer and places the text in the paste buffer. This is key 6 on the keypad. On VT200-series terminals, the REMOVE key also performs a cut.
DEL C	Deletes the character the cursor is on and places it in the character buffer. This is the comma (,) on the keypad.
DEL EOL	Deletes the text between the cursor and the next line terminator, including the character at the cursor but not the terminator, and places it in the line buffer. This is keypad key GOLD2. Compare this key with DEL L.

Table 4-13 (Cont.): Keypad Commands

Command	Function
DEL L	Deletes the text between the cursor and the next line terminator, including the terminator and the character at the cursor, and places it in the line buffer. This is PF4 on the keypad. Compare this key with DEL EOL. You can delete the line with one command, undelete it with UND L (GOLD PF4), and then delete it with the other command and notice the difference.
DEL W	Deletes the text between the cursor and the next word terminator, including the character at the cursor but not the word terminator, and places it in the word buffer. This is the underscore (_) on the keypad.
DOWN	Moves the cursor to the same position on the next line down, or to the end horizontal position if the next line contains enough characters, otherwise the end of the next line. This is the DOWN-ARROW.
ENTER	Causes the command to be executed when you enter a line-mode command (using COMMAND); terminates text strings you enter when searching (using FIND); terminates key definitions you enter when redefining keys (using CTRL/K).
EOL	Moves the cursor to the end of the current line. This is key 2 on the keypad.
FIND	Searches in the current direction for the search string. You are prompted for the string, which you must terminate with ENTER, ADVANCE, or BACKUP. (If you use ADVANCE or BACKUP, you may change the default direction.) FIND also stores the search string in the search buffer. This is GOLD PF3 (and also the FIND key on the VT200-series terminals).
FNDNXT	Searches in the current direction for the search string, which is the text stored in the search buffer. Text is entered in the search buffer when you use the FIND command or execute a line-editing or nokeypad command that specifies a search string. FNDNXT is PF3 on the keypad.
GOLD	Causes the next keypad key pressed to perform the alternate function. You can also enter repeat counts for commands by pressing GOLD, then a number, using the keyboard keys. GOLD is PF1.

Table 4–13 (Cont.): Keypad Commands

Command	Function
HELP	Displays a picture of the keypad on the terminal screen and asks what key you would like described, then displays a short description of the function of the key you specify. HELP is PF2.
LEFT	Moves the cursor one character position to the left; if the cursor is at the beginning of a line, LEFT moves it to the end of the previous line. This is the LEFT-ARROW.
OPEN LINE	Inserts a line terminator after the cursor position; this either creates a blank line or moves the text to the right of the cursor to a new line. This is GOLD 0 on the keypad.
PAGE	Moves across a PAGE in the current direction. PAGE is 7 on the keypad.
PASTE	Inserts a copy of the text in the paste buffer at the cursor position, leaving the paste buffer intact. PASTE is GOLD 6 on the keypad.
REPLACE	Deletes the Select Range and replaces it with the contents of the paste buffer. REPLACE is GOLD 9 on the keypad.
RESET	Cancels the current Select Range or cancels the GOLD key if you have pressed it and changed your mind about using it. RESET is GOLD . ("dot") on the keypad.
RIGHT	Moves the cursor one character position to the right; if the cursor is at the end of a line, RIGHT moves the cursor to the beginning of the next line. This is the RIGHT-ARROW.
SECT	Moves the cursor sixteen lines in the current direction, leaving it at the left margin. SECT is 8 on the keypad. On the VT200-series terminals, PREV SCREEN moves back one section and NEXT SCREEN advances one section.
SELECT	Marks a position in the buffer as being one end of a Select Range. The Select Range is everything between this marked position and the current cursor position. You can move the cursor in all the usual ways. SELECT is . (dot) on the keypad. The VT200-series terminals have a second SELECT key on the editing keypad.

Table 4–13 (Cont.): Keypad Commands

Command	Function
SPECINS	Allows you to insert a character into the buffer by specifying its ASCII value. First, press the GOLD key, then the numeric value using the keyboard keys and then press SPECINS. This is GOLD 3 on the keypad.
SUBS	Deletes the search string, replaces it with the contents of the paste buffer, and finds the next occurrence of the search string. This is GOLD ENTER on the keypad. Read the keypad help for information on using this key. Many users use the line-mode SUBSTITUTE command in conjunction with GOLD 7.
TOP	Positions the line the cursor is on at the top of the screen. This is GOLD 5 on the keypad.
UND C	Places the character in the character buffer at the cursor position, leaving the character buffer unchanged. That is, you can undelete the same character repeatedly. This is GOLD , (comma) on the keypad.
UND L	Places the text in the line buffer at the cursor position, leaving the line buffer unchanged. You can undelete the same line repeatedly. This is GOLD PF3.
UND W	Places the text in the word buffer at the cursor position, leaving the word buffer unchanged. You can undelete the same word repeatedly. This is GOLD _ (underscore).
UP	Moves the cursor to the same cursor position in the line above if it exists; if not, to the end of the line above. This is the UP-ARROW.
WORD	Moves the cursor to the next W in the current direction, past the next word delimiter. This is the 1 on the keypad.

In addition to these functions which are accessible from the keypad, several functions are available on the keyboard itself, as shown in Table 4–14. Keypad and keyboard keys that you never use are good candidates for redefinition.

Table 4-14: Keyboard Commands

Command	Function
BACK SPACE	Moves the cursor to the beginning of the line
LINE FEED	Deletes the previous word and puts it in the word buffer
CTRL/A	Sets tab indentation level to current cursor position (cursor position must be an even multiple of the SET TAB value)
CTRL/D	Decreases the tab level one unit
CTRL/E	Increases the tab level one unit
CTRL/F	Fills the selected range to the line width
CTRL/K	Accepts a definition for a keypad key or control character
CTRL/T	Moves selected range a tab stop left for a minus argument, right otherwise
CTRL/U	Deletes text between the cursor and the beginning of the line
CTRL/W	Refreshes the screen
CTRL/Z	Returns to line editing

4.6.6 Nokeypad Commands

Nokeypad commands are important not only for users who use nokeypad editing, but also for writing key definitions. Many powerful functions can be performed by strings of these commands (see Section 4.5.2 for a detailed discussion of key definitions).

Nokeypad commands must be entered exactly. Abbreviations are not allowed. The commands may be strung together without delimiters if you wish, or you may include spaces. You can repeat a string of commands by enclosing the string in parentheses. When a command in the string fails (for instance a search command does not find its object), EDT stops executing commands. All these rules also hold for a command string of nokeypad commands used in key definitions.

Nokeypad commands consist of one or more of the following elements: a nokeypad command, a count of actions or entities, a direction, an entity, and a buffer name. The count either tells EDT how many times to perform an action, or how many entities to handle. The direction is either + for forward (towards the end of the buffer) or – for backward (toward the beginning of the buffer). Entities are summarized in Section 4.6.4. The buffer name is used in CUT, PASTE, and APPEND commands to tell EDT where to store pieces of text.

There are three possible formats for nokeypad commands, each formed from combinations of the above-mentioned elements.

The legal formats are described with the commands using that format.

Syntax for Format 1

```
command
```

Format 1 commands specify EDT's actions. You cannot modify the action of these commands, so there are no qualifiers possible. See Table 4–15.

Table 4–15: Commands for Format 1

Command	Function
ADV, +	Both ADV and the + set the current direction to forward (towards the end of the buffer, which is to the right and down), unless you override it with the minus (–) or BACK in your command string.
BACK, –	Both BACK and – set the current direction to backward (towards the beginning of the buffer, which is to the left and up), until you override it with the plus (+) or ADV.
EX	Exits nokeypad editing and returns to EDT line mode.
EXT	Tells EDT that the rest of the line consists of line-mode commands; when EDT has finished executing the line-mode commands, it returns to character editing. The GOLD 7 key uses this function.
I	Opens the current buffer to allow you to insert text in front of the cursor position. You terminate the insertion with CTRL/Z, or follow the I with a semicolon (;) to insert the rest of the line into the buffer as text. The INSERT key on the VT220 editing keypad also performs this function.
QUIT	Exits from EDT and returns you to the system command level without saving any files.
REF	Refreshes the screen.

Table 4–15 (Cont.): Commands for Format 1

Command	Function
SEL	Selects a range of text when you enter the command at one end of a piece of text; then positions the cursor at the other end. The SELECT keys use this function.
TAB	If you have set a tab size (with the SET TAB editor-control command) and the cursor is at the beginning of a line, this command inserts enough tabs and spaces to set the cursor position to the SET TAB value times the indentation level count. See Section 4.5.4 for more on the level counter. If you have not set a tab size, or the cursor is not at the beginning of a line, this command inserts a tab character.
TC	Computes the current indentation level count by dividing the current cursor position by the value you gave with the SET TAB command. If the cursor position is not divisible by the tab value, EDT issues an error message. See Section 4.5.4 for more on the level counter.
TOP	Places the current line at the top (beginning) of the buffer.

Syntax for Format 2

```
[ + | - ][count] command
```

Format 2 commands have variable count and direction parameters. The default count for these commands is 1 in the current direction. Only two commands using this format allow you to vary the direction, SUBSTITUTE and SUBSTITUTE NEXT. See Table 4–16.

Table 4–16: Commands for Format 2

Command	Function
ASC	Allows you to enter arbitrary ASCII characters into a buffer by entering the ASCII value of a character before the ASC command. The SPECINS key uses this function.
S/s1/s2/	Replaces string s1 with string s2; if you give a count, that many substitutions are performed, and if you give a minus (−) the search goes backward (note that any nonalphabetic character can be used as a delimiter).

Table 4–16 (Cont.): Commands for Format 2

Command	Function
SN	Uses the strings defined in the last S (Substitute) command to replace the next occurrence of string s1 with string s2; if you give a count, that many substitutions are performed, and if you give a minus (–) the search goes backward.
SHL	Tells EDT to shift the image it displays on the screen to the left. The number of characters shifted is eight times the count entered with the SHL command. This command does not work if you have SET NOTRUNCATE.
SHR	Tells EDT to shift the image it displays on the screen to the right; the number of characters shifted is eight times the count entered with the SHR command. This command does not work if you have SET NOTRUNCATE.
TD	Decreases the indentation level count; if the count is zero, this command has no effect. See Section 4.5.4 for more on the level counter.
TI	Increases the indentation level count. See Section 4.5.4 for more on the level counter.
UNDC	Undeletes a character by copying it from the character buffer into the current text buffer in front of the cursor (leaving the character buffer untouched). If you give a repeat count, EDT inserts that many copies of the character in the buffer.
UNDW	Undeletes a word by copying it from the word buffer into the current text buffer in front of the cursor (leaving the word buffer untouched). If you give a repeat count, EDT inserts that many copies of the word in the buffer.
UNDL	Undeletes a line by copying it from the line buffer into the current text buffer in front of the cursor (leaving the line buffer untouched). If you give a repeat count, EDT inserts that many copies of the line in the buffer.
	Inserts control characters into your text buffer when you enter a letter following the circumflex. This does not work inside an insert block, between an I and the CTRL/Z.

Syntax for Format 3

```
[ + | - ] [repeat-count] command [ + | -] [entity-count]
[ + | - ] entity [ = buffer]
```

Format 3 commands specify an action on an entity. These commands permit variable count and direction parameters, and some can operate in a buffer other than the current one. You can place the direction sign in any of the indicated positions. If you include counts in both places, they are multiplied by each other. For example, the command string 3D3C deletes nine characters. See Table 4–17.

Table 4–17: Commands for Format 3

Command	Function
APPEND	Deletes the specified entities and moves them to the end of the specified buffer. If you do not include a buffer specification in the command, EDT uses the paste buffer. Compare with CUT.
CUT	Deletes the specified entities and moves them to the specified buffer. If you do not include a buffer specification, EDT uses the paste buffer. EDT copies over the previous contents of the receiving buffer. Compare with APPEND.
D	Deletes the specified entities.
FILL	Fills the specified text inside the margin defined by the SET WRAP command. All the text that fits inside the bounds set by the SET WRAP command is placed on each line, with the rest dropped to the next line. Text is broken only between words.
PASTE	Copies the contents of the specified buffer in front of the cursor, leaving the specified buffer unchanged. If you omit a buffer specification, EDT uses the paste buffer.
R	Deletes the specified entities and places you in insert mode, allowing you to replace the deleted text. Press CTRL/Z to exit insert mode.
TADJ	Moves the Select Range the specified number of tab stops in the specified direction. See Section 4.5.4.

4.7 EDIT Command

EDIT invokes EDT, the DEC standard Editor. See the *EDT Editor Manual* for more information.

Format

EDIT[/EDT][/qualifier[s]] infile
 or
EDIT[/qualifier[s]] infile

Command Qualifiers

/[NO]COMMAND[:filespec]
/[NO]JOURNAL[:filespec]
/[NO]OUTPUT[:filespec]
/[NO]READ_ONLY
/[NO]RECOVER

Parameter

infile

Specifies the file to be edited. If the file does not exist, EDT creates it.

You must supply a file name, but the type can be null.

Command Qualifiers

/[NO]COMMAND[:filespec]

Controls whether an EDT initialization file is read by EDT before editing begins. These files contain commands that alter the default setup for EDT, such as custom line-mode commands and change-mode key definitions.

The default is /COMMAND:EDTINI[.EDT].

If you use this qualifier and EDTINI.EDT or some other file you name does not exist, EDT issues no error message and continues with the editing session.

If you have a file EDTINI.EDT and do not want to use it, use the /NOCOMMAND qualifier.

/[NO]JOURNAL[:filespec]

Controls whether EDT creates a journal file for the editing session. The default is to create a journal file with a file name the same as that of the input file with the type .JOU. You can specify a different name by including a file specification.

The journal file consists of all editing commands and text entered during the session. If the editing session ends abnormally, such as through a system crash, or your inadvertently typing three CTRL/Zs in succession, the journal file is saved. In such a case, you invoke EDT again, with the same command line as before plus the /RECOVER qualifier. Your editing session is repeated and all your editing is restored. If the editing session ends normally, the journal file is deleted.

If you specify /NOJOURNAL, no journal file is created and no recovery is possible.

/[NO]OUTPUT[:filespec]

If you do not specify this qualifier, the default is to create a file of the same name and type as the input file with a version number one higher than the input file. If the file is new, EDT creates version number 1. You can alter the name of the output file by including a file specification with the /OUTPUT qualifier. Otherwise, the qualifier need not be included.

If you specify /NOOUTPUT, you cannot exit EDT without including a file specification in your EDT EXIT command.

/[NO]READ_ONLY

Specifies whether you want simply to read the file or to edit it. If your command line includes /READ_ONLY, you can use the full facilities of EDT, but you cannot exit without including a file specification in your EDT EXIT command. Normally, you would use the EDT QUIT command if you had specified /READ_ONLY. The /READ_ONLY qualifier is equivalent to a combination of /NOOUTPUT and /NOJOURNAL. You can use /READ_ONLY to look at files to which you have no write access.

The default is /NOREAD_ONLY, which need never be specified.

/[NO]RECOVER

Specifies whether EDT reads commands from a journal file prior to starting the editing session. With a journal file, your editing session can be restored if interrupted by a system crash or other problem. The default is /NORECOVER, which need never be specified.

The /RECOVER qualifier requests EDT to open the input file and then read EDT commands and text from the file with the same file name as the input file and the file type .JOU. The command line with /RECOVER added to it must be identical to the command line that initiated the original failed editing session. This means that if you specified an EDT initialization file, you must specify the same file in the /RECOVER command line. And, if you specified a name for the journal file other than infile.JOU, you must include the /JOURNAL qualifier with the appropriate file specification. If journaling was not enabled on the original command line, you cannot recover the editing session.

Examples

```
$ EDIT [RET]
File? HORNBLOW.TXT [RET]
Input file does not exist
[EOB]
*
```

This example shows the EDIT command in its simplest form. The file HORNBLOW.TXT is created during the editing session.

```
$ EDIT HORNBLOW.TXT [RET]
```

This is the one-line form of the same command.

```
$ EDIT/OUTPUT:UMPIRE.MAC WEAVER.MAC [RET]
   1              .TITLE WEAVER
*
```

In this example, the user takes an existing file as input and edits it to create a new file.

```
$ EDIT/COMMAND:FORMAT RAMBLE.RNO [RET]
Input file does not exist
[EOB]
*
```

In this example, the user prepared an EDT initialization file called FORMAT.EDT. This file contains EDT command definitions designed to be used with a text formatting program. See next example.

```
$ EDIT/COMMAND:FORMAT/RECOVER RAMBLE.RNO [RET]
!  This is FORMAT.EDT. Version 2.4     Ambrose Bierce, Maintainer
          .
          .
          .
```

During the editing session started in the previous example, the system crashed. Once it was back on the air, the user duplicated his original command exactly, naming the same EDT initialization file and then added the /RECOVER qualifier. EDT begins reproducing the entire editing session, beginning with reading in the initialization file, the first line of which is seen in the example. All the editing commands and text entered during the session have been recorded in the file RAMBLE.JOU, which is also read back in by EDT. The result is that the entire editing session is repeated up to within a few keystrokes of the crash.

If you want to test the recovery procedure, start editing a file and then enter three CTRL/Zs as line-mode commands. This causes an unnatural exit from EDT, so the journal file is saved. You can then duplicate the original command line, add the /RECOVER qualifier, and watch yourself edit.

Notes

See the *EDT Editor Manual,* which is available separately, for more information on journaling, initialization files, command and key definitions, and the other editing functions of EDT.

The qualifiers used with EDIT/EDT will not work with other forms of the EDIT command. If you use the /EDT qualifier, it must appear immediately after EDIT on the command line. Other qualifiers can float.

You can use the MCR command to invoke EDT in a different format that some users may find more convenient. See the *EDT Editor Manual* for the alternate command format.

Chapter 5

Files on Micro/RSX Systems

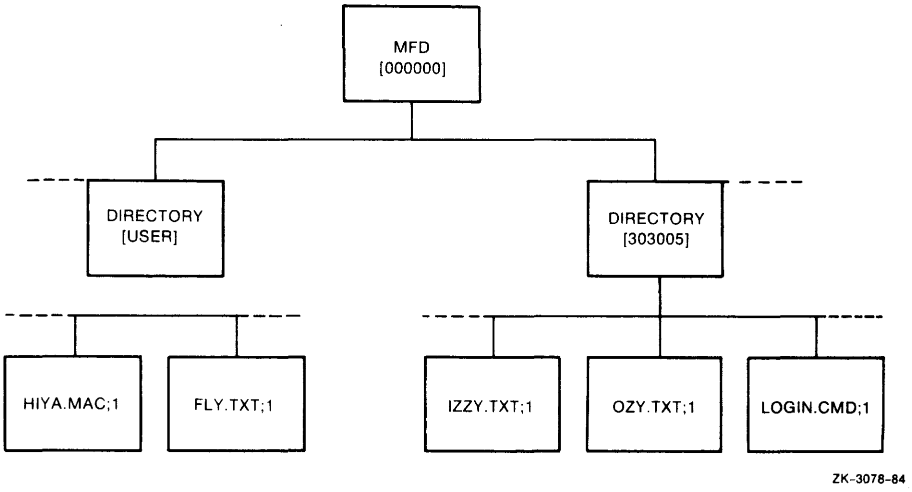
The commands in this chapter are used to create files, list them in directories, remove them from directories, print copies of them on your terminal or the line printer, and alter their contents in various ways. For a quick review of file-handling commands, see the *Introduction to Micro/RSX*. For complete detail on these commands, see the *Micro/RSX User's Guide*, Volume 2, Chapter 12, More About Files on Micro/RSX Systems.

All files are stored on volumes. A volume is a disk, diskette, or tape with data written on it in a form that Micro/RSX can use. Once the volume is mounted, you can access the files on the volume. For more information about volumes and devices, see Chapter 6, Devices and Volumes, or Volume 2, Chapter 13, More About Devices and Volumes.

Within each disk volume, files are organized in directories. For the entire volume, there is a Master File Directory (MFD). The MFD is a file named [000000]000000.DIR. Most files listed in the MFD are User File Directories (UFDs). In general, UFDs are simply called directories. Each directory is a file with a name based on the name of the directory and the type .DIR.

See Figure 5–1 for an illustration of the relationship between volume, MFD, and directory.

Figure 5-1: File Organization on a Volume



5.1 File Ownership and Location

When you log in, you identify yourself with a User Identification Code (UIC). You log in to your default device and directory. The default directory has a name (or number) assigned by the system manager at the time your account is set up. Nonprivileged users can change their default device and directory, but not their UIC. Privileged users can change both. The UIC identifies the user; the directory identifies the location of the files.

If you issue the SHOW DEFAULT command, you can find out your default device (SY:), your default directory, and your UIC. The SHOW UIC command gives you your UIC. Both these commands are described in Section 5.10.

The default device is actually a volume mounted on a device. Most Micro/RSX users have their main accounts on a fixed disk, but you can also have accounts on diskettes as well.

Regardless of whether it is DU0:, DU1:, or some other device, your default device can always be called either by its name or by the name SY:. This is a logical assignment made each time you log in. Information on your default device is kept by ACNT, the Account File Maintenance Program, and picked up from there each time you log in. See the *Micro/RSX System Manager's Guide* for more information on ACNT.

When you create a file, the system places the file name in a directory along with a UIC. The UIC indicates the ownership of the file.

Each directory is a file with a name based on the name of the directory and the type .DIR. Thus, directory [KIZZY] is a file listed in the Master File Directory (MFD) named KIZZY.DIR, and directory [200001] is a file listed in the MFD named 200001.DIR.

Directory entries consist of the names of files and pointers to file headers. The file header holds information about the file's owner and the location of the file on the mass-storage medium. Thus, a full file specification not only identifies the file, but locates it as well.

5.2 File Specification

A file specification uniquely identifies a file, indicating its location and its contents. Many DCL commands require file specifications.

The format of a file specification is as follows:

```
ddnn:[directory]name.type;ver
```

Parameters

ddnn:

Specifies the type of device and unit number on which the volume containing the file is mounted. A full device name consists of two alphabetical characters followed by one or two numbers, each from 0 through 7, followed by a colon. Examples include DU0:, DU1:, MS0:.

[directory]

Specifies the name of the directory in which the files are located.

Micro/RSX accepts either *named directories* or *numbered directories*. The format for a named directory is as follows:

[directory]	where directory is from one to nine of the following characters: the 26 letters from A through Z, and the numbers from 0 through 9.

Examples of named directories include [MINGUS], [RITTENBRG], [001002], and [A1B2C3].

The format for a numbered directory is as follows:

[g,m]	where g and m are octal numbers from 1 through 377.

Examples of numbered directories include [1,2] and [303,17].

See Volume 2, Chapter 12 for a description of the /NONAMED_DIRECTORY qualifier for the SET DEFAULT command. When your terminal's default is /NONAMED_DIRECTORY, then you can specify only numbered directories in the SET DEFAULT command.

name

Specifies the name of the file, which can be from zero through nine of the following characters: the 26 letters from A through Z, and the numbers from 0 through 9.

.typ

Specifies the file type. The type can be from zero through three of the following characters: the 26 letters from A through Z, and the numbers from 0 through 9. In general, the file type indicates the file contents. For example, .FTN is the file type for FORTRAN source programs. Null file types (0 length) are acceptable.

Always separate the name from the file type with a period (.).

There are no other restrictions on file types, but many system tasks use default file types for input and output files. These defaults and some system conventions on file types are summarized in Table 5–1.

;ver

Specifies the version number, a decimal number from 1 through 32767. The version number identifies different versions of the same file. When you create a file without specifying a version number, the system assigns the file version number 1. Each time you create a new version of a file—by editing it, for instance—the system adds one to the version number.

You must separate the file type and version number with a semicolon.

You can also create a file with an explicit version number.

If you create a file with version number 32767, you can be sure that file will not be inadvertently superseded by a file with a higher number; 32767 is the highest possible version number. Under normal circumstances, if you create a file with the same name and type as an existing file, the new file's version number is one higher than the old one. This cannot happen with the version number 32767. This feature is useful where you must have a particular version of a file for some reason.

In addition, you can use version 0 or version -1 in commands. Version 0 is the most recent version of a file. Version -1 is the oldest version of a file.

In most cases, Micro/RSX systems do not require you to enter the full file specification. Defaults are supplied for all fields of the file specification except the name (see Section 5.3). In addition, you can use wildcards to specify groups of related files (see Section 5.4).

5.3 Defaults in File Specifications

Except for the file name, if you omit a field of the file specification, the system automatically supplies the field as described here:

Field	Default
ddnn:	The system establishes your default device when you log in. SY: is always your default device. Logging in assigns SY: to some physical device. You can use SET DEFAULT to change the assignment of SY:; you can display the assignment with SHOW DEFAULT (see Section 5.10). You do not have to use the unit number for devices numbered 0. DU: is the same as DU0:, for instance.
[directory]	The system manager specifies your default directory when establishing your account. Your default directory can be any legal directory name. Normally, your default directory is either your last name or a directory whose numbers are identical to your User Identification Code. You can change the default directory with SET DEFAULT (Section 5.10.1); you can display the default directory with SHOW DEFAULT (Section 5.10.2).
.typ	Standard file types (see Table 5–1) are used as defaults. There is not a default file type in every situation. Neither EDT nor the CREATE command assigns a default file type. If you create a file without specifying a file type, the file type is null, meaning only the dot (.) is there. For instance, the following command: `$ CREATE NOFILTYPE [RET]` creates a file named NOFILTYPE.;1.
;ver	For input files, the default is the highest numbered version; for output files, it is the next higher version number, or 1 if no previous version exists.

The defaults make it possible for you to specify files without having to type in the full file specification.

Suppose, for example, that your default device is DU0:, which the operating system assigns the logical device name SY:, and your default directory is [PISMO]. Assume further that there are three sequentially numbered versions of a file called CLAM.TXT in the directory. If you want to print the highest numbered version of this file (version 3) on your terminal, you can use any of the following forms of the TYPE command, as they all produce the same result:

```
$ TYPE CLAM.TXT
$ TYPE CLAM.TXT;3
$ TYPE CLAM.TXT;0
$ TYPE [PISMO]CLAM.TXT;3
$ TYPE DU0:CLAM.TXT
$ TYPE DU0:[PISMO]CLAM.TXT
$ TYPE SY:CLAM.TXT
$ TYPE SY:[PISMO]CLAM.TXT;3
$ TYPE SY:[PISMO]CLAM.TXT;0
```

In most cases, you will use the first form, but if you want to keep a permanent record of the terminal session, you may choose one of the more explicit forms. In any case, the final form given is what the system uses, as that form has all the defaults supplied. (Note that version 0 is always assumed to be the highest numbered version, which is number 3 by definition in this example.)

Table 5-1: File Types

File Type	Use
.BAT	File containing batch processing commands. System convention. See Chapter 8 for more information about batch processing.
.BLD	Indirect command file used by the System Generation procedure to create files needed to build system tasks. System default.
.B2S	BASIC-PLUS-2 source program. System default. BASIC-PLUS-2 is available separately as a software option.
.CBL	COBOL source program. System default. COBOL is available separately as a software option.

Table 5-1 (Cont.): File Types

File Type	Use
.CMD	Indirect command file. System default. See Chapter 9 for more information about indirect command processing.
.DAT	File containing data, as opposed to code. System convention.
.DIR	File is either Master File Directory or User File Directory. System default.
.DMP	Dump file created by DMP, the File Dump Utility. System default. DMP is part of the Micro/RSX Advanced Programmer's Kit.
.FTN	FORTRAN-77 source program. System default. FORTRAN-77 is available separately as a software option.
.LOG	Log of batch processing session. System default. See Chapter 8 for more information about batch processing.
.LST	Listing file. System default.
.MAC	MACRO-11 source program. System default. The MACRO-11 Assembly Language is available in the Micro/RSX Advanced Programmer's Kit as a software option.
.MAP	Task Builder map file. System default. See Volume 2, Chapter 14, LINK and LIBRARY Commands, for more information. See also the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual,* available separately, and as part of the Advanced Programmer's Kit.
.MLB	Macro library. System default. See the Advanced Programmer's Kit for more information.
.OBJ	Object module output from assembler or compiler. System default. See the Advanced Programmer's Kit, or separate language documentation for more information.
.ODL	File containing Overlay Descriptor Language to be used by Task Builder. System default. See the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual,* available separately, and as part of the Advanced Programmer's Kit, for more information about overlays.
.OLB	Object module library. System default.

Table 5–1 (Cont.): File Types

File Type	Use
.PMD	File containing Postmortem Dump of interrupted task. System default. See the Advanced Programmer's Kit for more information.
.SYS	A bootable system image. System default.
.TMP	A temporary file. System convention.
.TSK	Task image file. System default.
.TXT	A text file. System convention.
.ULB	Universal library. System default.

5.4 Wildcards in File Specifications

In addition to the regular defaults for the current device, the current directory, and the most recent version, you can use wildcards with the commands in this chapter to set up temporary defaults for every part of the file specification except the device name.

Simple wildcarding uses the asterisk (*) to replace any or all fields in the file specification except the device name.

For instance, the following command:

```
$ DIRECTORY [*] RET
```

lists all files in all directories on the default volume. The [*] means "all directories."

The following command:

```
$ DIRECTORY [100,*] RET
```

lists all files in any numbered directories that have a group number of 100.

The following command:

```
$ DIRECTORY [*]TEXT.TXT RET
```

lists the most recent versions of all files named TEXT.TXT, regardless of the directory.

Likewise, the * in place of the version number means "all versions."

The following command:

```
$ DIRECTORY WOM.BAT;* RET
```

lists all versions of the file WOM.BAT in the default directory.

The asterisk (*) can also be used to replace an entire file name or file type in much the same way.

The following command:

```
$ DIRECTORY *.BAT RET
```

lists the most recent versions of all files with the type .BAT in the default directory.

All the examples that follow assume that you want a listing for files on the default volume and default directory.

The following command:

```
$ DIRECTORY COMMON.* RET
```

lists the most recent versions of all files with the name COMMON and any type.

The examples thus far have demonstrated the simple form of wildcarding, using the asterisk character (*) to replace an entire field in a file specification. Simple wildcarding works with all the commands in this chapter. The PRINT command accepts simple wildcarding, as does the SUBMIT command for batch processing.

For the commands DIRECTORY, DELETE, PURGE, COPY, RENAME, TYPE, APPEND, UNLOCK, and SET PROTECTION, a more elaborate form of wildcarding also is available. In these commands, within file names and file types, the * can be used in a more complex manner. The * actually means "match zero or more characters in this position."

Wildcards are safest to use with DIRECTORY than any other command because the DIRECTORY command does not manipulate the contents of the files. They are most difficult to use with RENAME and COPY. They are most dangerous to use with DELETE and PURGE. It is always a good idea to check your wildcard specifications with the DIRECTORY command before you do anything drastic.

Therefore, the following command:

```
$ DIRECTORY L*.TXT RET
```

lists the most recent versions of all files with the type .TXT whose names start with "L".

And the following command:

```
$ DIRECTORY *L*.TXT [RET]
```

lists the most recent versions of all files with the type .TXT whose names include an "L".

The same substitutions can also be used in file types, so that the following command:

```
$ DIRECTORY SNOBLO.L* [RET]
```

lists the most recent versions of all files with the name SNOBLO and the type beginning with an "L".

You can use more than one wildcard in file names and file types.

The following command:

```
$ DIRECTORY *F*D*.TXT [RET]
```

lists the most recent versions of all files with the type .TXT whose names include an "F" and a "D" in that order.

In addition, the same commands permit the percent sign (%) to be used as a wildcard, but only within file names and file types. The % means "match exactly one character in this position."

The following command, for instance:

```
$ DIRECTORY %.TXT [RET]
```

lists all files with the type .TXT and a single-character file name.

The following command:

```
$ DIRECTORY NOV%%85.TXT [RET]
```

lists all files with the type .TXT and a file name consisting of NOV and 85 separated by two characters.

Both kind of wildcards can be combined in a single file specification.

The following command:

```
$ DIRECTORY %L*T.TM% [RET]
```

lists all files whose names begin with a single character followed by an "L" and end with a "T" and with a file type consisting of .TM and another single character.

Wildcarding, combined with systematic policies of directory assignments, file names, and file types can add considerable flexibility and convenience to your use of the system.

5.5 Date-Related Qualifiers

The commands DIRECTORY, DELETE, PURGE, COPY, RENAME, TYPE, APPEND, UNLOCK, and SET PROTECTION also accept several other qualifiers that add further flexibility to these commands.

First, there are the date-oriented qualifiers:

```
/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
```

These qualifiers all depend on the creation date of the file as shown in the DIRECTORY listing. /DATE limits the operation of the command to files created on the specified date. /SINCE limits the operation of the command to files created on or after the specified date. /THROUGH limits the operation of the command to files created before or on the specified date. /SINCE and /THROUGH can be combined to limit the operation of the command to files created within a given range of dates. /TODAY limits the operation of the command to files created on the same day the command was issued.

You can enter the date in either of the following two forms:

dd-mmm-yy	as in 25-MAY-85
or	
mm/dd/yy	as in 05/25/85

The system always displays dates in the first form.

These same commands also accept the following qualifier:

```
/EXCLUDE:filespec
```

The /EXCLUDE:filespec qualifier allows you to exclude a file or files from the operation of the command. The file specification argument to /EXCLUDE must include a version number, but the version number can be *. Wildcards are accepted for all parts of the file specification argument to /EXCLUDE.

The /EXCLUDE qualifier makes it possible to do the following:

```
$ DELETE/EXCLUDE:MICHAEL.TXT;* [RET]
File(s)? *.TXT;* [RET]
```

This command deletes all files with the type .TXT, except those named MICHAEL.TXT.

5.6 Protection

Micro/RSX is a multiuser system. Each user is able to work without interference from any other user. One way Micro/RSX provides data privacy and system security is through restrictions on access to volumes and the files on the volumes.

The best protection for sensitive files is to put them on a diskette, and then lock that diskette away. The following section explains how files and volumes can be protected by software on Micro/RSX systems. No protection is completely safe, just as no lock is a defense against a determined and resourceful burglar.

In order to access a file, you must have access to the volume, access to the directory, and access to the file. Protection can be specified for all these entities.

5.6.1 File Protection

Each user has a unique User Identification Code (UIC) that the system manager assigns while setting up your account. Your UIC identifies you to the system.

When you create a file, you usually own it. Your UIC is an attribute of the file, identifying you as the owner. Each file also has a protection code. A protection code controls who can access a file and in what ways. You, as the file's owner, control this protection code.

UICs look like the following:

    [303,5]

The UIC consists of two numbers: the first number is your group number, and the second is your member number. Group and member numbers are octal and range from 1 through 377.

Generally, all users working on a particular project have the same group number. As you will see in the following description of file protection codes, a common group number can allow group members to share files easily. In addition, group numbers indicate whether or not you are a privileged user. A privileged user can look at or change any file on the system. Privileged users have group numbers of 10 or less; nonprivileged users have group numbers from 11 through 377.

The second number of your UIC identifies you as a particular member of a group. The combination of the group and member numbers uniquely identifies an individual user of the system.

Note that, if you are a privileged user, you can change your UIC. See Section 5.10.3 for a description of SET UIC.

A file protection code specifies four categories of users, as well as four kinds of file access that each category of user can have. When you attempt to access a file, the operating system checks your UIC to determine which of the four user categories you belong to. Your ability to access the file is limited to the types of access that the file's protection code grants to those categories.

The four categories of users in a file protection code are as follows:

SYSTEM	The *operating system* itself, and *privileged users*. A privileged user has a UIC with a group number of 10 or less.
OWNER	The *file owner* who is the user having the same UIC as the one the file was created under—that is, your UIC.
GROUP	All users having a UIC with the *same group number* as the one the file was created under.
WORLD	*All other* users.

The four kinds of file access that you can grant to the user categories are as follows:

READ	The user, or the user's tasks, may read, copy, print, or type the file. If the file is a task image file, READ access means you may run it. If you deny READ access to a class of users, they can't run the task.
WRITE	The user, or the user's tasks, may add new data to the file by writing to it. You can't edit a file without WRITE access, nor can you use the COPY command to write a file to a directory where you have no WRITE access.

EXTEND	The user, or the user's tasks, may change the amount of disk space allocated to the file. In practice, you always specify EXTEND access along with WRITE access in the protection code.
DELETE	The user, or the user's tasks, may delete the file. You can't delete a file without DELETE access.

Your UIC relates to the user categories in a file protection code in the following ways:

- Your UIC establishes you as either a privileged or nonprivileged user. If you are a privileged user, you have the file access rights granted to the SYSTEM category. Also, as a privileged user, you can change the protection code of any file. By changing a file protection code for the SYSTEM category, therefore, you can access any file in any way.
- When you create a file, the operating system gives your unique UIC to the file. Your UIC, then, becomes one of a file's attributes, establishing you as the file's owner. As the owner of a file, you can change its protection code to suit your needs. Also, you have the file access rights granted to the OWNER category.
- The first number in your UIC establishes you as a member of a group. As a member of a group, you have the file access rights granted to the GROUP category.
- You are always a member of the WORLD category.

Ordinarily, the operating system assigns the system's default protection code to files. In a protection code, the four types of file access are designated by their first letters. The default protection is as follows:

```
SYSTEM:RWED,OWNER:RWED,GROUP:RWED,WORLD:R
```

Note that under this default protection scheme any user can read your files, because the WORLD category has READ access. Furthermore, everyone in your GROUP category and all privileged users have full READ, WRITE, EXTEND, and DELETE access. They can all delete your files.

If you have files that you particularly want to protect against deletion, or if you want to limit the access people have to your files, use the SET PROTECTION or the SET PROTECTION/DEFAULT commands described in Sections 5.10.5 and 5.10.6. Be aware, though, that there is no way to protect your files from privileged users, because they can change the protection code for any file.

You can display the protection for any file with the SHOW PROTECTION command. The DIRECTORY/FULL command also displays file protection, in the following form:

```
[RWED,RWED,RWED,R]
```

In the DIRECTORY display, the groups are not given, but they appear in the same order as before: SYSTEM, OWNER, GROUP, WORLD. In the previous example, then, WORLD has READ privileges.

A number of DCL commands include qualifiers that help you use file protection. The most important is the /OWN qualifier to the COPY command. If you don't specify otherwise, when you copy one of your files to send it to someone else, you are the owner of the copy, even if it isn't in your directory.

Assume you have UIC [303,5] and you are placing a copy of a file you own into a directory called [MOONDOG], which is owned by UIC [200,2]. You use the following command:

```
$ COPY RAMP.TXT [MOONDOG]RAMP.TXT RET
```

The file RAMP.TXT is in both directories, but its owner in both cases is [303,5]. The owner of [MOONDOG] has only WORLD READ access to her copy of RAMP.TXT, as seen in the following directory listing, where the file owner is the first item on the second line of each file listing:

```
$ DIRECTORY/FULL RAMP.TXT, [MOONDOG]RAMP.TXT RET
Directory DUO:[THEFLU]
3-MAR-85 15:49

RAMP.TXT;52          (7516,5)       2./2.         1-JAN-85 13:12
   [303,5]  [RWED,RWED,RWED,R]

Total of 2./2. blocks in 1. file

Directory DUO:[MOONDOG]
3-MAR-85 15:49

RAMP.TXT;1           (14343,6)      2./2.         3-MAR-85 15:48
   [303,5]  [RWED,RWED,RWED,R]

Total of 2./2. blocks in 1. file

Grand Total of 4./4. blocks in 2. files in 2. directories
```

The /OWN qualifier to the COPY command specifies that the recipient of the copy is the owner, not the sender. The following commands illustrate this:

```
$ COPY/OWN RAMP.TXT [MOONDOG]RAMP.TXT RET
$ DIRECTORY/FULL RAMP.TXT, [MOONDOG]RAMP.TXT RET
Directory DU0:[THEFLU]
3-MAR-85 15:50

RAMP.TXT;52        (7516,5)       2./2.      1-JAN-85 13:12
   [303,5]  [RWED,RWED,RWED,R]

Total of 2./2. blocks in 1. file

Directory DU0:[MOONDOG]
3-MAR-85 15:50

RAMP.TXT;2         (14343,6)      2./2.        3-MAR-85 15:49
   [200,2]  [RWED,RWED,RWED,R]

Total of 2./2. blocks in 1. file

Grand Total of 4./4. blocks in 2. files in 2. directories
```

You do not need the /OWN qualifier if you are copying a file **from** somewhere else. Normally, when you create a file, you own it. When you copy a file to some other place, you are creating the file, so you must override your default ownership with the /OWN qualifier.

5.6.2 Directory Protection

Because a directory is a file, it has a protection code like that of a file. READ access is the right to list the directory, and WRITE access is the right to create files in the directory. If you do not have EXTEND access to a directory, you may be denied the right to create more than a certain number of files in the directory. DELETE access means the right to delete the directory file.

The default access for directory and file protection is the protection code the disk was mounted with. The default for mount protection is the protection code the volume was initialized with. See the discussion of CREATE/DIRECTORY in Section 5.7.2 and also Chapters 6 and 13 for more information.

The CREATE/DIRECTORY command includes a /PROTECTION qualifier that provides a means of overriding the default system protection. This qualifier sets the protection for the directory file itself, **not** for files in the directory.

Here is an example:

```
$ CREATE/DIRECTORY/PROTECTION:(SYSTEM:,OWNER:RWED,GROUP:,WORLD:) RET
Device and UFD? DU1:[RAGBAG] RET
```

This example creates a directory called [RAGBAG] on the volume mounted on device DU1:. The protection code is the protection code applied to the actual directory file DU1:[000000]RAGBAG.DIR. The protection code specifies that the owner (the person issuing the command) has full access to the directory and the files in it and that SYSTEM, GROUP, and WORLD have no access. These three groups may not list the directory (no READ privilege), nor can they create files in the directory (no WRITE or EXTEND privilege), or delete the directory itself (no DELETE privilege). That is, only the owner can do anything with the directory.

Remember that this command does nothing to the protection of the individual files in the directory. If a sophisticated user knows the file ID numbers of your files, they're still accessible. Remember also that although you can protect your files against SYSTEM access by privileged users, a privileged user can still issue a SET PROTECTION command and change the protection to SYSTEM:RWED, whereupon the privileged user has complete access.

You can display the protection of a directory file in much the same way as any other file. The following command displays the protection for the directory created in the previous example.

```
$ DIRECTORY/FULL DU1:[000000]RAGBAG.DIR RET
```

5.6.3 Volume Protection

A disk, diskette, or tape with files in Micro/RSX format on it is called a Files–11 volume. Volume protection works in much the same way as file protection except it affects the entire volume.

You can specify the volume protection at the time you initialize the volume or at the time you mount it. Both the INITIALIZE command and the MOUNT command (see Chapters 6 and 13) include a /PROTECTION qualifier and an /OWNER qualifier for setting volume protection. In addition, both MOUNT and INITIALIZE include a /FILE_PROTECTION qualifier that permits you to specify the protection for any new files created on that disk volume. The INITIALIZE/UPDATE command enables you to change the default protection on an existing volume. See Volume 2, Chapter 13 for more information.

Volume protection codes differ from file protection codes in that EXTEND protection is called CREATE protection. The function of the code is the same. Either EXTEND protection or CREATE protection refers to the right to allocate space on a volume. CREATE under volume protection refers to your ability to allocate space for a new file, while EXTEND under file protection refers to your ability to allocate space within an existing file.

The default protection applied to all volumes on the system that have not been otherwise protected is as follows:

```
SYSTEM:RWCD,OWNER:RWCD,GROUP:RWCD,WORLD:RWCD
```

(In fact, you can use a "C" in a file protection code or an "E" in a volume protection code and the command will be accepted, but it is good practice to use the correct protection category.)

If you need to know the protection of a volume (and the default file protection for new files created on the volume), include the /SHOW qualifier in your MOUNT command. See Chapter 6 for more information.

As shipped, the fixed disk uses the default volume protection and file protection. These can be changed with the INITIALIZE/UPDATE command. See Volume 2, Chapter 13, More About Devices and Volumes. The odds are that no one has changed the protection on the fixed disk. There is no simple way to display the protection of the fixed disk.

5.7 Creating Directories and Sequential Files

The CREATE command without a qualifier enables you to create a sequential file directly at your terminal. The CREATE/DIRECTORY command permits a user to create a directory.

5.7.1 CREATE

CREATE creates a sequential file and enables you to type text directly into the file from your terminal without using an editor.

Format

CREATE
File? filespec

CREATE filespec

Parameter

filespec

Specifies the name of the file to be created.

As soon as the command is entered, the cursor moves down a line. The file is open for input. Any text you type goes into the file.

When you have finished entering text, press CTRL/Z to close the file.

If you want to create an empty sequential file, simply press CTRL/Z first. See the examples.

Examples

```
$ CREATE [RET]
File? COPY.CMD [RET]
; Command file to move files from Hank's directory to my directory [RET]
[RET]
        .ENABLE SUBSTITUTION [RET]
.LOOP: [RET]
        .ASKS FI What file [RET]
        COPY DU2:[HCWII]'FI' 'FI' [RET]
        .GOTO LOOP [RET]
[CTRL/Z]
$
```

In this example, the user wanted to move a number of files from one directory to another. With CREATE, the user created an indirect command file at the terminal. After entering the CREATE command and file specification, the cursor (or print head) moves to the left margin. The terminal is attached for input to CREATE. Note that while CREATE is executing, the RETURN key acts only as a carriage return, and not as a means of entering commands to the system. No commands are accepted by DCL until after you press CTRL/Z (echoed as ^Z).

```
$ CREATE JOE.TXT [RET]
Joe called at 4:30 on Monday.  He'll be back from Switzerland on [RET]
Thursday, but he won't be in the office until Friday afternoon. [RET]
[CTRL/Z]
$
```

In this example, the user used CREATE to take notes on a phone call.

```
$ SHOW DEFAULTS [RET]
DU0:[POGO]  TT3:
$ MOUNT/NOSHAREABLE [RET]
Device? DU1: [RET]
Label? OKEFENOKE [RET]
$ CREATE/UFD [RET]
File? DU1:[ALBERT]ALLIGATOR.TXT [RET]
I declare a good-looking man looks good in anything. [RET]
[CTRL/Z]
$
```

In this example, the user wanted to create a file on a diskette and create a directory to put it in. The user placed a diskette that already had some files on it in diskette drive DU1: and issued the MOUNT command with the /NOSHAREABLE qualifier. This qualifier makes the diskette private. No one else can access it and the user who mounted it is privileged as far as that diskette goes. The user then issued the CREATE command with the /UFD qualifier, named a file including a directory name, and entered text into the file, all without changing the default directory.

```
$ CREATE CHURCHYLA.FEM [RET]
[CTRL/Z]
```

In this example, the user used CREATE to create an empty file for later use. There are times when you need to have a file present, but the file need not have any contents.

Notes

CREATE differs slightly in batch jobs. See Chapter 8 for further information.

If you create a file without specifying a version number and no file of that name and type already exists, the file is version 1. If a file of that name and type exists and you give no version number, the new file has a version number one higher than the highest numbered previous version.

If you specify a version number when creating the file, the system creates a file with the version number you give.

If you use CTRL/U when creating a sequential file, the text on the line is eliminated, but not the line itself. In other words, CTRL/U leaves a blank line behind when it deletes a line. CTRL/U, CTRL/R (refresh the line), and the DELETE key are the only editing facilities available to you when creating sequential files at the terminal.

5.7.2 CREATE/DIRECTORY

CREATE/DIRECTORY creates a directory on a Files–11 volume and enters the directory into the volume's Master File Directory (MFD). Privileged users can create directories on volumes mounted on any device. Nonprivileged users can create directories only on volumes mounted on their own private (allocated or nonshareable) devices.

Format

CREATE/DIRECTORY[/qualifier[s]]

Device and UFD? [ddnn:][[directory]]

CREATE/DIRECTORY[/qualifier[s]][ddnn:][[directory]]

Command Qualifiers

/ALLOCATION:n
/LABEL:volumelabel
/NOWARNINGS
/OWNER_UIC:[uic]
/PROTECTION:(code)

Parameters

[ddnn:][[directory]]

You must specify at least one of these parameters. If you specify one parameter, the default volume or directory is used for the other parameter. You cannot create a directory that matches both your default device and your default directory.

ddnn:

Specifies the disk on which you want to create a directory. This parameter defaults to your default device.

The volume must be a Files-11 volume mounted with the /NOSHAREABLE qualifier (or allocated) if you are not privileged.

[directory]

Specifies the name of the directory you want to create. This parameter defaults to the current default directory.

Directory names are from one to nine of the following characters: the 26 letters from A through Z, and the numbers from 0 through 9.

To retain compatibility with other RSX systems, Micro/RSX also accepts numbered directory names. The format for a numbered directory is [g,m], where g and m are octal numbers from 1 through 377.

Command Qualifiers

/ALLOCATION:n

Specifies the number of directory entries (file names) for which space is to be initially allocated. The value n is rounded up to the next multiple of 32.

The default is /ALLOCATION:32.

/LABEL:volumelabel

Specifies that the volume label that you supply be compared with the label on the volume. If the names match, a directory can be allocated. If they do not match, the command is rejected.

If you do not specify this qualifier, the volume label is not checked.

/NOWARNINGS

Suppresses error messages resulting from the command.

/OWNER_UIC:[uic]

Specifies the User Identification Code (UIC) of the directory's owner. A UIC establishes the protection status of the directory. See Section 5.6.2 for more information. The default is for the directory to be owned by you. You can specify any UIC as the owner. However, a nonprivileged user cannot always create files in a directory that is owned by a UIC different from his own.

/PROTECTION:(code)

Specifies the protection code for the directory file, **not** for the files in the directory. See Section 5.6.2 for more information.

Examples

```
$ CREATE/DIRECTORY RET
Device and UFD? DU2:[JRICE] RET
```

This example creates directory [JRICE] on the diskette volume mounted on device DU2:. Nonprivileged users can create directories only on volumes mounted with the /NOSHAREABLE qualifier. Privileged users can create directories on volumes mounted on any device.

```
$ CREATE/DIRECTORY DU2:[JRICE]RET
```

This example is identical in effect to the previous one.

```
$ CREATE/DIRECTORY RET
Device and UFD? [BURLESON] RET
```

This example creates directory [BURLESON] on the default device.

```
$ CREATE/DIRECTORY RET
Device and UFD? DU1: RET
```

This example creates a directory on the volume mounted on DU1: with the same name as the default directory.

```
$ DELETE/DIRECTORY [BDENT]BDENT.DIR RET
```

This example deletes the directory [BDENT] from the current device. All directory files are in [000000]. You must have DELETE access to the directory file to delete it. The files in the directory are no longer accessible.

5.8 Maintaining a Directory

The following commands provide you with the means for maintaining your directory. Disk space is an important resource, particularly on the fixed disk. You will probably want to have only a limited number of your most immediately useful files on line on the fixed disk. You should not keep files that you do not need. Files that you want to save, but do not need to have immediate access to, should be copied onto a diskette or tape and kept off line.

5.8.1 DIRECTORY

DIRECTORY displays information on files in directories.

Format

DIRECTORY[/format-qual][/destination-qual] [filespec[s]]

Command Qualifiers

Format Qualifiers

/BRIEF
/FREE[:ddnn:]
/FULL
/SUMMARY

Destination Qualifiers

/OUTPUT:filespec
/PRINTER

Other Qualifiers

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec
/REWIND

Parameter

filespec[s]

Specifies the file or files for which information should be displayed. If you do not supply a file specification, a complete directory listing of the default directory is displayed.

You can supply one or more file specifications, separated by commas. Directory information on the files you name is displayed.

You can use wildcards in place of any file specification field except the device field. If you do not supply a version number, only information on the most recent versions is displayed. However, if you do not supply a file type, the operating system assumes the file has no type at all, as if its file specification were something like FILE.;1. If you do not know the file type, use a wildcard.

You can display a listing of another directory by supplying the directory name in this field. You can also specify device names in the form ddnn: in this field.

If you name files in more than one directory or device, you should name files for the default directory first. If you name files from another device or directory first, the defaults are canceled. See the examples.

Command Qualifiers

Command qualifiers are in the following three groups:

1. *Format qualifiers* control the appearance and detail of the directory list.
2. *Destination qualifiers* control where the command output is sent.
3. *Other qualifiers* modify the list of files by creation date or exclusion.

If you do not supply a format qualifier, the display is in standard format, giving the file name, type, and version number, the number of blocks the file occupies, and the date and time of creation.

If you do not include a destination qualifier, the display appears on your terminal.

If you do not include any of the other qualifiers, the display includes all files that otherwise qualify.

Format Qualifiers

/BRIEF

Specifies that the display give file names, types, and version numbers only.

/FREE [ddnn:]

Specifies that the display give the free space and number of free file headers on the default device or a specified device.

/FULL

Specifies that the complete directory entry be displayed, including file ID number, blocks used and allocated, the owning UIC, and protection status of the file, in addition to all the information in the standard display.

/SUMMARY

Specifies that the display give only the total number of blocks allocated and used for the specified files. If you give no file specifications in the command, the display shows the total blocks allocated and used for the default directory.

Destination Qualifiers

These qualifiers direct output to an output file or to the line printer.

/OUTPUT:filespec

Specifies that the output of the DIRECTORY command be placed in a file having the file specification you supply.

/PRINTER

Specifies that the output of the DIRECTORY command be printed on your system's line printer, if you have one.

Other Qualifiers

/REWIND

Specifies that you want the tape to be rewound to the beginning before listing the directory. This qualifier is for magnetic tapes only.

Examples

```
$ DIRECTORY [RET]

Directory DU0:[BLOCKHEAD]
1-MAY-85 14:16

A.A;1                1.         25-SEP-84 12:29
AZ.CMD;5             1.         02-APR-85 13:03
BYE.CMD;1            1.         25-SEP-84 12:29
CLEAN.CMD;1          1.         10-JAN-85 08:29
DUN.CMD;4            1.         23-JAN-85 08:05
EDT.CMD;22           1.         07-NOV-84 15:56
LOX.CMD;1            1.         27-APR-85 10:21
LOGIN.CMD;6          1.         06-APR-85 15:56
TI.CMD;4             1.         30-APR-85 08:36

          .
          .
F.TSK;1              4.      C  25-SEP-84 12:30
G.TSK;1              4.      C  25-SEP-84 12:30
PONG.TSK;2           12.     C  25-SEP-84 12:31
BUZZ.TXT;2           4.         12-MAR-85 09:13
IZZY.TXT;1           2.         06-MAR-85 14:33
JIVE.TXT;1           1.         16-NOV-84 13:23
Total of 2906./3043. blocks in 160. files
```

This example shows a directory listing in the standard format.

The device and directory are named at the head of the listing, along with the date and the time. All files in the directory are listed because the user did not supply any file specifications. The files shown with a C in column 3 are contiguous files, in this case, task images. You may see files in a directory with an L in this position, signifying that the files are locked. Files are locked when they are closed improperly. See Section 5.9.4 for information on the UNLOCK command.

```
$ DIRECTORY/SUMMARY [RET]

Storage used/allocated for Directory DU1:[RSTONES]
1-MAY-85 14:15
```

```
Total of 2892./3033. blocks in 160. files
```

This example displays the output produced by the /SUMMARY qualifier. If the user had supplied one or more file specifications in the command, the summary would cover the blocks used and allocated for the specified files only.

```
$ DIRECTORY/FREE RET

DU1: has 383. blocks free, 417. blocks used out of 800.
Largest contiguous space = 203. blocks
17. file headers are free, 31. headers used out of 48.
```

This example displays information about the default device, which is one of the diskettes in this case. Most files need not be contiguous, so you probably don't need that information. The display also informs you that you cannot add more than 17 files totaling 383 blocks to the diskette. Both of these figures are absolute limits. That is, if you add one file totalling 383 blocks, that's it, or, if you add 17 files totalling only 17 blocks, you have reached the limit. See the description of the INITIALIZE command, Section 6.4, for more information on the capacity of the fixed disk and diskettes.

```
$ DIRECTORY/FREE LB0: RET

DU0: has 6074. blocks free, 15526. blocks used out of 21600.
Largest contiguous space = 3520. blocks
397. file headers are free, 3603. headers used out of 4000.
```

This example displays information about the pseudo device LB0:. Notice that the display identifies pseudo device LB0: as being DU0:. All systems in the RSX family run off the same pseudo device, LB0:, which can be many different physical devices. Most Micro/RSX systems run off DU0:, which is the fixed disk.

```
$ DIR TI.CMD RET

Directory DU0:[PANCHO]
1-MAY-85 14:17

TI.CMD;4            1.          30-APR-85 08:36

Total of 1./5. blocks in 1. file
```

This example displays information on the most recent version of the file named because the most recent version is the default. See next example.

```
$ DIR TI.CMD;* RET

Directory DU0:[PANCHO]
1-MAY-85 14:17
```

```
TI.CMD;4            1.          30-APR-85 08:36
TI.CMD;3            1.          27-APR-85 16:32
TI.CMD;2            1.          27-APR-85 16:29
TI.CMD;1            1.          27-APR-85 16:28
Total of 4./20. blocks in 4. files
```

This example displays information on all versions of the file named because the user supplied a wildcard for the version number, overriding the default of the most recent version.

```
$ DIR/FULL TI.CMD [RET]
Directory DUO:[PANCHO]
1-MAY-85 14:17
TI.CMD;4              (300,56)       1./5.            30-APR-85 08:36
  [303,5]   [RWED,RWED,RWED,R]     03-MAY-85 09:58(2.)
Total of 1./5. blocks in 1. file
```

This example displays full information on the most recent version of the file named. The number in parentheses is the file identification number assigned by the system when the file is created. The first number is the file number, and the second is the sequence number of the file. The UIC is the UIC of the owner, the protection status of the file, and the final column on the second row is the last revision date and revision level. The revision date is the date the file was last manipulated without a new version being created (for example, appending one file onto another). See the notes.

Protection status is listed in the order SYSTEM, OWNER, GROUP, WORLD. See Section 5.6 for an explanation of protection. You can change file protection with the SET PROTECTION command; see Section 5.10.5.

```
$ DIRECTORY/BRIEF *.CMD [RET]
Directory DUO:[PANCHO]
ADV.DMD;2
AZ.CMD;5
BYE.CMD;1
CLEAN.CMD;1
DAY.CMD;1
DTC.CMD;1
DUN.CMD;4
EDT.CMD;22
LOX.CMD;1
VTK.CMD;1
TI.CMD;4
```

This example displays brief information on the most recent version of all files of the type .CMD in the default directory.

```
$ DIRECTORY/TODAY [RET]

Directory DU0:[PANCHO]
5-MAY-85 13:44
Day of 5-MAY-85

2051CH4.MEM;1       339.      05-MAY-85 10:11
NEWINSDC3.RNO;7     10.       05-MAY-85 10:05
2051CH4.MEM;2       320.      05-MAY-85 10:12
FUT.MAI;1           1.        05-MAY-85 13:05
NEWINSDC3.RNO;6     9.        05-MAY-85 09:55
NEWINSDC3.RNO;10    10.       05-MAY-85 10:39
NEWINSDC4.RNO;3     7.        05-MAY-85 11:46
FRK.CMD;6           1.        05-MAY-85 13:43
NEWINSDC4.RNO;4     11.       05-MAY-85 13:43
LOG.LOG;1           0.        05-MAY-85 13:43

Total of 708./736. blocks in 10. files
```

In this example, the user asked for a directory of all files created the same day the command was issued. Note that the directory listing includes the date. See the next example.

```
$ DIRECTORY/TODAY/EXCLUDE:*.RNO;* [RET]

Directory DU0:[PANCHO]
5-MAY-85 13:45
*.RNO;* excluded
Day of 5-MAY-85

2051CH4.MEM;1       339.       05-MAY-85 10:11
2051CH4.MEM;2       320.       05-MAY-85 10:12
FUT.MAI;1           1.         05-MAY-85 13:05
FRK.CMD;6           1.         05-MAY-85 13:43
LOG.LOG;1           0.         05-MAY-85 13:43

Total of 672./693. blocks in 6. files
```

In this example, the user listed all files created the day the command was issued but excluded all files with the .RNO file type. Note that the file specification argument to /EXCLUDE has a wildcard version number. Observe also that the exclusion is noted in the heading of the directory listing.

```
$ DIR *.TSK, [BADGE]*.TXT [RET]

Directory DU1:[CREAM]
20-MAY-85 09:39
```

```
CLOCK.TSK;3          8.      C  15-MAY-85 08:55
F.TSK;1              4.      C  25-APR-85 12:30
G.TSK;1              4.      C  25-APR-85 12:30
TICTAC.TSK;4         19.     C  01-MAY-85 12:07
TREK.TSK;1           54.     C  25-APR-85 12:31

Total of 89./89. blocks in 5. files
Directory DU1:[BADGE]
20-MAY-85 09:39

FRONT.TXT;1          3.         16-JAN-85 11:23
EDITNEWS.TXT;12      25.        11-MAR-85 10:50
OZY.TXT;3            2.         11-MAR-85 10:50
TEXT.TXT;1           151.       11-MAR-85 10:50
NEW.TXT;1            0.      L  27-APR-85 14:36
AWARE.TXT;2          2.         27-APR-85 13:48
IZZY.TXT;10          2.         12-MAY-85 13:42

Total of 184./205. blocks in 7. files

Grand total of 273./294. blocks in 12. files in 2. directories
```

In this example, a user with the default directory of [CREAM] requested directory information on files in two directories. The default directory was applied to the first files named, the *.TSK files; these were listed first in the display. The second set of files displayed, the *.TXT files, were from directory [BADGE], which was explicitly specified in the command. Note the locked file. Compare this example with the next one.

```
$ DIR [BADGE]*.TXT, *.TSK RET

Directory DU1:[BADGE]
20-MAY-85 09:40

FRONT.TXT;1          3.         16-JAN-85 11:23
EDITNEWS.TXT;12      25.        11-MAR-85 10:50
OZY.TXT;3            2.         11-MAR-85 10:50
TEXT.TXT;1           151.       11-MAR-85 10:50
NEW.TXT;1            0.      L  27-APR-85 14:36
AWARE.TXT;2          2.         27-APR-85 13:48
IZZY.TXT;10          2.         12-MAY-85 13:42
ADVENT.TSK;1         151.    C  13-MAY-85 14:12
DUNGEON.TSK;2        242.    C  21-FEB-85 09:08
PONG.TSK;2           12.     C  13-MAY-85 14:13
YCLOCK.TSK;6         9.      C  08-APR-85 12:17
STAR.TSK;2           102.    C  13-APR-85 09:07
TICTAC.TSK;5         19.     C  01-FEB-85 11:31
VTCHS.TSK;12         26.     C  06-FEB-85 15:39
XCLOCK.TSK;6         9.      C  03-JAN-85 16:48
LIFE.TSK;10          4.      C  03-APR-85 13:28

Total of 758./780. blocks in 16. files
```

The command in this example is quite similar to the command used in the previous one, but notice the difference in results. The same user, with default directory [CREAM], issued the command but this time the directory [BADGE] was specified first, for the *.TSK files. Even though no directory was specified for the *.TSK files, these files were also listed from directory [BADGE]—**not** from [CREAM]. This demonstrates that specifying a directory in a DIRECTORY command resets the default directory for the rest of the command or until another directory is specified.

Notes

The full format for directory listings may include two dates. The first is the creation date of the file. The second date indicates the last time the file was revised by the system or a task for write access, such as for editing. The decimal number in parentheses following the second date is the number of times the file has been changed in this fashion.

Your directory is a file that you own, but which is kept in directory [000000] on the same volume as your directory. In directory [000000] you will find the Master File Directory (MFD), which is a file named 000000.DIR. The MFD is the directory for directory [000000]. All directories on the volume appear in this directory in the same form. The directory file for [303005] is called 303005.DIR; the directory file for [PANCHO] is PANCHO.DIR, and so forth.

These directory files have file protection like all other files. The protection is for the directory file, **not** for files in the directory. Because the directory is a file, READ access is the right to list the directory, and WRITE access is the right to create files in the directory. If you do not have EXTEND access to a directory, you may be denied the right to create more than a certain number of files in the directory. DELETE access means the right to delete the directory file. Because you own your directory file, you can usually delete it, but do not do so unless you're sure of what you are doing.

5.8.2 DELETE

DELETE deletes specified files and releases the storage space the files occupy.

Format

DELETE[/qualifier]
File(s)? filespec[s]

DELETE[/qualifier] filespec[s]

Command Qualifiers

/EXCLUDE:filespec
/LOG
/NOWARNINGS
/QUERY
/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yyy
/TODAY

Parameter

filespec[s]

Specifies the file or files to be deleted.

You must supply the name, type, and version number fields of the file specifications of the files you want to delete. Device and directory fields default to your current device and directory. You can use wildcards in any file specification field except the device field. You need not supply a file type to delete a file with a null file type.

You can only delete files to which you have DELETE access.

To name more than one file for deletion, separate their file specifications with commas.

Command Qualifiers

/LOG

Specifies that a list of the files deleted be displayed on your terminal.

/QUERY

Specifies that you want to decide which files should be deleted on an individual basis. Each file that is specified in the command is named. You may enter one of six characters:

Y	—	Deletes file named and goes on to next file.
N	—	Does not delete file named and goes on to next file.
G	—	(Go) Deletes the file and goes on to delete all other files specified.
Q	—	(Quit) Does not delete the file and exits the task. No more files are deleted.
CTRL/Z	—	Same as Q (Quit).
RETURN	—	Same as N (No).

Remember that you can specify files by default or wildcard. See the examples. If you do not specify a version number, /QUERY is the default. See the examples.

Examples

```
$ DELETE RET
File(s)? *.DAT;* RET
```

This example deletes all versions of all files of the type .DAT.

```
$ DELETE FILE.FIL;1, ;3, .TXT;5 RET
```

This example deletes versions 1 and 3 of FILE.FIL, and version 5 of FILE.TXT. Note that no file name or type is specified for version 3, and no name for .TXT;5.

```
$ DELETE/QUERY RET
File(s)? *.TMP;* RET
Delete file  DU1:[GOETHE]OGRE.TMP               [Y/N/G/Q]? Y RET
Delete file  DU1:[GOETHE]TROLL.TMP;1            [Y/N/G/Q]? Y RET
Delete file  DU1:[GOETHE]ORC.TMP;1              [Y/N/G/Q]? Y RET
Delete file  DU1:[GOETHE]ELF.TMP;1              [Y/N/G/Q]? N RET
Delete file  DU1:[GOETHE]HOBBIT.TMP;1           [Y/N/G/Q]? N RET
Delete file  DU1:[GOETHE]SNIPE.TMP;1            [Y/N/G/Q]? G RET
The following files have been deleted:
DU1:[303,5]SNIPE.TMP;1
DU1:[303,5]SNOPE.TMP;2
```

In this example, the user specified all files having the type .TMP in the DELETE command. Three files were deleted at the user's choice, and two were retained. The user then directed that all remaining files with the type.TMP be deleted. There were two more files, which were deleted and listed.

```
$ DELETE *.DOC RET
Delete file   DUO:[MABON]WITCH.DOC;2              [Y/N/G/Q]? Y RET
Delete file   DUO:[MABON]DRY.DOC;4                [Y/N/G/Q]? N RET
Delete file   DUO:[MABON]PAYCHECK.DOC;1           [Y/N/G/Q]? Y RET
Delete file   DUO:[MABON]LOADING.DOC;3            [Y/N/G/Q]? Y RET
Delete file   DUO:[MABON]LOADING.DOC;4            [Y/N/G/Q]? Y RET
Delete file   DUO:[MABON]LOADING.DOC;5            [Y/N/G/Q]? Y RET
Delete file   DUO:[MABON]LOADING.DOC;6            [Y/N/G/Q]? N RET
```

In this example, the user specified the file type .DOC and a wildcard for the file name. Because no version number was given, the DELETE command defaulted to the /QUERY qualifier, enabling the user to choose which files to delete.

```
$ DELETE/LOG *.LST;* RET
The following files have been deleted:
DU1:[JIMP]RANGER.LST;1
DU1:[JIMP]TONTO.LST;1
DU1:[JIMP]REID.LST;1
DU1:[JIMP]SILVER.LST;1
DU1:[JIMP]SCOUT.LST;1
DU1:[JIMP]HORNET.LST;1
DU1:[JIMP]KATO.LST;1
```

In this example, the user specified all files with the type .LST in the DELETE command and asked for a log of all files deleted.

Notes

You must have DELETE access to delete a file.

If you want to delete using wildcards, it is wise to get a directory listing using the same file specifications you plan to delete. In this way, you can be sure you are not deleting more files than you intend to delete.

Remember that under the default file protection on Micro/RSX systems, you have DELETE access to your own files and all files in directories with the same group number in the UIC.

See Section 5.9.3.1 for DELETE commands directed to the Queue Manager.

5.8.3 PURGE

PURGE deletes all but the latest versions of files, and releases the storage space the deleted files occupy.

Format

PURGE[/qualifier[s]]
File(s)? filespec[s]

PURGE[/qualifier[s]] filespec[s]

Command Qualifiers

/EXCLUDE:filespec
/KEEP:n
/[NO]LOG
/NOWARNINGS
/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY

Parameter

filespec[s]

Specifies the file group or file groups to be purged.

Multiple file specifications must be separated by commas. Wildcards can be substituted for directory, name, and type fields.

You can purge any files to which you have DELETE access.

Command Qualifiers

/KEEP:n

Specifies that the n latest versions of a file be retained. The value for n can be any number.

If you do not use this qualifier, all versions but the most recent of a given file are deleted. That is, the default form of the command includes the qualifier /KEEP:1. With the qualifier explicitly stated, all but the n highest numbered versions are deleted. PURGE assumes that version numbers are in numerical sequence and without missing numbers. See the examples.

If more than one file specification is given with the /KEEP qualifier, all but the latest n versions of all files listed are deleted.

/[NO]LOG

Specifies that the files deleted by PURGE be listed on your terminal. The default is /NOLOG.

Examples

```
$ PURGE [RET]
File(s)? TEMPER.TSK [RET]
```

In this example, all versions of TEMPER.TSK but the latest are deleted.

```
$ DIRECTORY ASDIC.TM1;*, ELPASO.TEX;* [RET]
DIRECTORY DB1:[303,5]
20-MAY-85 13:44

ASDIC.TM1;1         1.          20-MAY-85 13:41
ASDIC.TM1;2         1.          20-MAY-85 13:41
ASDIC.TM1;3         1.          20-MAY-85 13:41
ASDIC.TM1;4         0.          20-MAY-85 13:41
ASDIC.TM1;5         1.          20-MAY-85 13:42
ELPASO.TEX;1        1.          20-MAY-85 13:42
ELPASO.TEX;2        1.          20-MAY-85 13:42
ELPASO.TEX;3        1.          20-MAY-85 13:42
ELPASO.TEX;5        1.          20-MAY-85 13:43
ELPASO.TEX;6        1.          20-MAY-85 13:43
Total of 9./45. blocks in 10. files
$ PURGE/LOG/KEEP:3 ASDIC.TM1 [RET]
The following files have been deleted:
DB1:[303,5]ASDIC.TM1;1
DB1:[303,5]ASDIC.TM1;2
$
$ PURGE/LOG/KEEP:3 ELPASO.TEX [RET]
The following files have been deleted:
DB1:[303,5]ELPASO.TEX;1
DB1:[303,5]ELPASO.TEX;2
DB1:[303,5]ELPASO.TEX;3
$
$ DIRECTORY ASDIC.TM1;*, ELPASO.TEX;* [RET]
Directory DB1:[303,5]
20-MAY-85 13:46

ASDIC.TM1;3         1.          20-MAY-85 13:41
ASDIC.TM1;4         0.          20-MAY-85 13:41
ASDIC.TM1;5          1.          20-MAY-85 13:42
ELPASO.TEX;5         1.          20-MAY-85 13:43
ELPASO.TEX;6         1.          20-MAY-85 13:43

Total of 4./20. blocks in 5. files
```

In this example, the user started with two sets of five files. The five files named ASDIC.TM1 have version numbers in order. The five files named ELPASO.TEX are numbered 1,2,3,5,6. The user issued a PURGE command with the qualifier /KEEP:3. Versions 3, 4, and 5, of ASDIC.TM1 were kept, but only versions 5 and 6 of ELPASO.TEX were kept. This is because the /KEEP:3 qualifier does **not** save the three highest numbered files, but rather the highest numbered file and the next two lower numbered files of the same name in numerical sequence. If there had been a file ELPASO.TEX;4, it would have been saved. Because there was none, the PURGE command task exited, its work done.

Note

You can purge any file to which you have DELETE access.

5.8.4 COPY

COPY copies files.

COPY creates a sequential file copy of one or more sequential files, or of records with either indexed or relative file organization.

Format

COPY[/qualifier[s]]
From? infile[s][/qualifier]
To? outfile

COPY[/qualifier[s]] infile[s][/qualifier] outfile

Command Qualifiers

/OWN
/REPLACE
/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec

Parameters

infile[s]

Specifies the input file or files to be copied.

You must have READ access to a file to copy it.

Multiple file specifications, separated by commas, are accepted. If you specify multiple input files and a explicit output file, they will be concatenated to the output file in the order that you specify them.

outfile

Specifies a single output file to which the input file or files are copied.

You must have WRITE access to the directory to which you want to send a copy. Under the default protection, you have WRITE access to your directory and to all directories with the same group number for their owner's UIC. That is, a user with the UIC [303,5] can normally copy to directories whose owners also have the group number 303, such as [303,26].

You can change the name, type, and version number of the file when you enter this parameter. Wildcards in the place of the name and the type leave the name and type unchanged.

The output file can be created by COPY. The output file need not exist when you issue the command. If a file of the same name and type already exists, then the file you create has a version number one higher than the highest existing version. If you specify a version number for the output file field, then a file of that version number is created. If such a file already exists, the operation fails unless you specify the /REPLACE qualifier, which causes the existing file to be replaced by the input file.

Wildcards are acceptable for output files if the destination is another directory. If you have multiple input files and use wildcards for the output file, you create multiple output files, each with the name and type of the corresponding input file.

You can send copies to devices as well as to directories. See examples.

You can also use the COPY command to create multiple copies of the same file with the same or different names.

Command Qualifiers

/OWN

Changes the ownership of the file being copied to the destination directory. After execution, both directories own their respective copies. If you do not specify /OWN, the original UIC owns both copies. This can lead to confusion later when you try to edit or delete the file and find that you do not have access because of conflicts in file ownership.

If you are copying from another directory to your own, use this qualifier.

/REPLACE

If the output file has the same name, type, and version number as an already existing file at the destination, the first file is deleted and the file you have sent replaces it. The name, type and version number stay as they were.

Examples

```
$ COPY [RET]
From? [KERMIT]TSKBLD.CMD [RET]
To? TSKBLD.CMD [RET]
```

This example copies TSKBLD.CMD from [KERMIT] to the current directory and device. The file is still owned by the original owner, from whom it was copied.

```
$ COPY *.BAS DU1: [RET]
```

This example copies all files with the file type .BAS from the current device and directory to the same directory on the volume mounted on device DU1:.

```
$ COPY/OWN TSKBLD.CMD [KERMIT]BLDFIL.CMD [RET]
```

This example copies TSKBLD.CMD from the current directory and device to [KERMIT], assigns ownership to the owner of [KERMIT], and also changes the file name to BLDFIL.CMD.

```
$ COPY OLD1.FIL,OLD2.FIL [RET]
To? NEW.ONE [RET]
```

This example copies two previously existing files into one new file.

```
$ COPY CHARLA.DMP TT4: [RET]
```

This example prints a copy of CHARLA.DMP on TT4:. This is a convenient means of sending messages longer than one line. If the terminal is busy at the time you send the copy, the copy is held until the terminal is clear and then sent.

```
$ COPY FLY.TXT SPIDER.TXT [RET]
```

This example creates the file SPIDER.TXT with the same contents as FLY.TXT, both on the current device and directory.

Notes

C is the short form of copy.

COPY does not affect file organization. If you want to change file organization, use the CONVERT command, which is explained in Volume 2, Chapter 12, More About Files on Micro/RSX Systems.

Usually, READ access is much broader than WRITE access. Under the default protection setup, your READ access covers the whole system, but your WRITE access is limited to other directories with the same group number. Thus, you can copy from many places that you cannot copy to.

Also, when you copy a file, you do not copy its protection code. Your copy has the default protection code. Use SET PROTECTION if you want to change the protection code of the copy. See Section 5.6 for more information on protection and Section 5.10.5 for more information on the SET PROTECTION command.

5.8.5 RENAME

RENAME changes the name, type, or version number of an existing file.

Format

RENAME[/qualifier[s]]
Old file name? infile
New file name? outfile

RENAME[/qualifier[s]] infile outfile

Command Qualifiers

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yyy
/TODAY
/EXCLUDE:filespec

Parameters

infile

Specifies the file to be renamed.

You may give a wildcard for either the file name or the file type, or both. If you use a wildcard in these fields, you must supply an entry in the version number field. This may be a wildcard.

If you give a wildcard for version number, all versions retain their old version numbers. If no version number is supplied, only the highest version of the named file is renamed. It has the same version number as the old file. If other files having the new name exist, then you receive an error message.

outfile

Specifies the new name for the file.

Wildcards leave that portion of the file specification the same as before. No wildcard is needed for the version number.

Examples

```
$ RENAME [RET]
Old file name? INTRO.TXT [RET]
New file name? APPENDIX.TXT [RET]
```

In this example, the most recent version of INTRO.TXT becomes APPENDIX.TXT;1.

```
$ RENAME IZZY.TXT;4 FIZZY.* [RET]
```

In this example, IZZY.TXT;4 becomes FIZZY.TXT;1. Other versions of IZZY.TXT are not affected.

```
$ RENAME AMA.DOC;4 *.*;11 [RET]
```

In this example, AMA.DOC;4 becomes AMA.DOC;11.

```
$ RENAME MAIN.TSK;* EXTRA.TSK [RET]
```

In this example, all files named MAIN.TSK are renamed EXTRA.TSK. The version numbers remain the same, regardless of sequence and order of file creation.

```
$ RENAME MAIN.TSK;* SUB.* [RET]
```

In this example, all versions of MAIN.TSK are renamed SUB.TSK. Versions are in the order of creation, with numbers unchanged.

```
$ REN EXHAUST.*;* REFRESH.* [RET]
```

In this example, all files named EXHAUST of whatever type are renamed REFRESH. Their file types remain the same. Note that a wildcard is given for the input file type. This makes an entry in the version number field mandatory.

Notes

You cannot rename files across devices.

Using wildcards is tricky. Experiment before committing yourself.

5.9 More File-Related Commands

The commands in this section perform varied actions on your files. The TYPE command displays files on your terminal. The PRINT command prints files on the system line printer. The APPEND command alters the contents of certain files. The DIFFERENCE command displays the differences between two files. The UNLOCK command unlocks locked files.

5.9.1 TYPE

TYPE prints selected files on your terminal.

Format

TYPE[/qualifier[s]]
File(s)? filespec[s]

TYPE[/qualifier[s]] filespec[s]

Command Qualifiers

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec

Parameter

filespec[s]

Specifies the file or files to be printed on your terminal.

You can specify any file to which you have READ access.

The file name and type must be specified explicitly or with a wildcard (*). If no version is specified, the most recent version is printed. A wildcard in any field prints every file that matches otherwise. Both the * and the % wildcards are accepted.

You can also specify a device and directory. If you do not, these fields default to the current device and directory.

Multiple file specifications must be separated by commas.

Examples

```
$ TYPE RET
File(s)? ROMEO.TXT RET
       Everybody loves a lover.
```

This example prints the file ROMEO.TXT from the current device and directory on your terminal.

```
$ TYPE [SHAKEY]ROMEO.TXT RET
       A Rose by any other name might not be Pete.
```

This example prints the file ROMEO.TXT from the current device and directory [SHAKEY] on your terminal.

```
$ TYPE [*]*.FIL;* RET
This is the beginning of a file named ARTHUR.FIL.
It goes on for several lines.
          .
          .
          .
THIS MARKS THE INITIATION OF A FILE CALLED ALBERT.FIL.
IT TOO CONTINUES AT SOME LENGTH.
          .
          .
          .
Now comes a file that starts like this called Z.FIL.
And some more.
          .
          .
          .
At last, we reach the beginning of a file named FILLIE.FIL.
Again some more text and then
          .
          .
```

```
The last two lines of FILLIE.FIL.
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
$
```

This example prints on your terminal all versions of all files having the type .FIL located on the current device.

Notes

In general, Micro/RSX documentation uses the term "print" to refer to any output sent to a terminal and "type" to refer to any user input from a terminal. The name of the TYPE command is an exception to this rule.

If you are typing multiple files, the CTRL/O feature works in a slightly different fashion. When you enter a CTRL/O, output is descarded until you press another CTRL/O or until the beginning of the next file is reached, whereupon output is sent to your terminal again. This leaves you free to use CTRL/O on a file-by-file basis.

You can type any file, but generally only ASCII text files are readable.

5.9.2 PRINT

PRINT queues files for printing on a line printer, if any. It can also be used to queue jobs for other output devices that are serviced by the Queue Manager.

Format

```
PRINT[/qualifier[s]]
File(s)?  filespec[/qualifier[s]][,filespec[s]]

PRINT[/qualifier[s]] filespec[/qualifier[s]][,filespec[s]]
```

Command Qualifiers

```
/AFTER:TOMORROW
/AFTER:(dd-mmm-yy hh:mm)
       (mm/dd/yy hh:mm)
/COPIES:n
/DEVICE:ddnn:
/[NO]FLAG_PAGE
/FORMS:n
/JOB_COUNT:n
/[NO]JOB_PAGE
/LENGTH:n
```

/NAME:jobname
/PRIORITY:n
/QUEUE:queuename

File Qualifier

/COPIES:n

Parameter

filespec[s]

Specifies the file or files to be printed on the line printer.

Multiple file specifications must be separated by commas.

The asterisk character (*) can be used in directory, name, type, and version number fields, but only to stand for the entire field. Wildcarding of the form L*.TXT is not accepted by PRINT, nor is the percent sign (%) wildcard accepted.

If your file specification includes no file type, the default file type is .LST.

Command Qualifiers

/AFTER:TOMORROW

Specifies that you want the job printed after midnight.

This qualifier is particularly useful in command files or batch jobs that are run every day. If your job or command file includes a PRINT command, the /AFTER:TOMORROW qualifier makes whatever job you specify the first print job of the next day.

/AFTER:(dd-mmm-yy hh:mm:[ss])
(mm/dd/yy hh:mm:[ss])

Blocks the job until after the specified time. Depending on the print queues at that time, your job may be run immediately, or later, when its turn comes up in the queue.

The parentheses, hyphens (or slashes), colons, and the space between the calendar and clock fields are all necessary syntax. Note that a space is used only between the calendar and clock fields. The space is not necessary if either field is omitted.

You can specify the calendar field in either of the following two formats:

dd-mmm-yy	Uses a 1- or 2-digit number for the day, the first 3 letters for the month, and 3 numbers for the year.
mm/dd/yy	Uses a 1- or 2-digit number for the month, a 1- or 2-digit number for the day, and 2 numbers for the year.

If you leave out the calendar field, the day defaults to the current date.

The argument yy must be between 77 and 92. If you leave it out, the year defaults to the current year.

The argument hh:mm:ss is the hour, minute, and second.

If you leave out the clock field, the time defaults to 00:00:00 on the given date.

The argument ss need not be given because it defaults to 00. In fact, the field is not actually used; if you do specify this argument, it is checked for syntax only. The Queue Manager always takes a few seconds to start the job.

Specifying a time using /AFTER is equivalent to issuing a PRINT command at that time. The job may still have to wait in its queue to be printed.

/COPIES:n

Specifies the number of copies to be printed. The default is /COPIES:1. If your PRINT command includes this qualifier and specifies more than one file, the resulting QMG print job consists of the specified number of copies of the first file named, followed by that many copies of the second file, and so on. If you want to print multiple copies of files by sets, use the /JOB_COUNT qualifier. See the examples.

The /COPIES:n qualifier to the file specification can be used to override this qualifier for a given file in a list of file specifications.

/DEVICE:ddnn:

The /DEVICE qualifier specifies that you want the file printed on a particular device.

/[NO]FLAG_PAGE

Adds flag pages to each file in your print job. With /FLAG_PAGE, the number of flag pages is equal to the number of job flag pages that precede the job.

With /NOFLAG_PAGE (the default), your job is still preceded by a job flag page, but the files in the job are printed without any file flag page, separated only by form feeds. You can suppress the job flag page with the /NOJOB_PAGE qualifier. Figures 5–2 and 5–3 show the difference between the job flag page and the file flag page.

/FORMS:n

Specifies the type of form on which the job must be printed. The argument n can have a value from 0 through 256. The default is 0. See your system manager for details on how to specify the particular forms used at your installation.

/JOB_COUNT:n

Specifies the number of sets of files you want printed. The default is /JOB_COUNT:1. If your PRINT command includes this qualifier and specifies more than one file, the resulting QMG print job consists of a set consisting of the files named in order, followed by another set, and so on. If you want to print multiple copies of one or more files, but not in sets, use the /COPIES qualifier. See examples.

/[NO]JOB_PAGE

Specifies whether you want a job flag page printed at the beginning of the QMG print job. The default is /JOB_PAGE. Use /NOJOB_PAGE when you are printing on special forms, or for some other reason do not want to include a job flag page. See the /[NO]FLAG_PAGE qualifier for controlling file flag pages. Figure 5–2 and 5–3 show the difference between job flag pages and file flag pages.

/LENGTH:n

Sets the length of a logical page; n can be any number from 0 through 255. The default is n=0. If you set a length, a form feed is automatically generated if one is not found within n lines. If your line printer has perforations for 66-line pages, it will not print on the perforations in the paper if you set /LENGTH:60.

When you set the length at 0, the default, the logical page length is unlimited.

This qualifier has no effect on how the printer handles form-feed characters. A form feed still moves the printer to the top of the next physical form, such as the perforation in a sheet of line printer paper. This qualifier simply adds the requirement that the printer move to the top of the next physical page if no form-feed character is encountered within n lines. See the example using /LENGTH.

/NAME:jobname

Specifies a name for the print job. Your job name can be nine alphanumeric characters.

If you specify a job name, that name appears on the flag page at the beginning of the printed output. Otherwise, the file name of the first file is used as the job name.

The job name also appears in the SHOW QUEUE display. The job name includes the UIC under which it was entered.

There can be more than one job of the same name, but each job has a unique entry number which appears in the SHOW QUEUE display.

/PRIORITY:n

Sets the queue priority of the print job. For nonprivileged users, n can be from 0 through 150. Privileged users can set n up through 250. The default is 50.

The Queue Manager prints the highest priority jobs first. When two jobs have the same priority, the job that has been in the queue longest is printed first.

/QUEUE:queuename

Specifies the name of the print queue in which the job is to be placed. The default is the PRINT queue.

File Qualifier

File qualifiers override the corresponding command qualifier when you have given multiple file specifications as parameters to the PRINT command. See the examples. See Volume 2, Chapter 12 for several more file qualifiers to the PRINT command.

/COPIES:n

Overrides the /COPIES command qualifier when attached to a particular file specification.

Examples

```
$ PRINT RET
File(s)? PASHA.CMD RET
PRI -- Job 305, name "PASHA    ", submitted to queue "PRINT "
```

This example prints the most recent version of PASHA.CMD from the default device and directory on the line printer. The Queue Manager returns a message confirming that the job has been submitted and assigned

Figure 5-2: Job Flag Page

```
Micro/RSX    V3.0      [1,10]LOGIN      - NO PAGE LIMIT      15-APR-85    17:07:43
Micro/RSX    V3.0      FORM #0 - NORMAL HARDWARE FORMS      15-APR-85    17:07:43
Micro/RSX    V3.0      NO IMPLIED FORM FEED                 15-APR-85    17:07:43
Micro/RSX    V3.0      JOB CONTAINS ONE FILE                15-APR-85    17:07:43




 SSSS Y   Y  SSSS M   M  GGGG RRRR
S     Y   Y S     MM MM G     R   R
S      Y Y  S     M M M G     R   R
 SSS    Y    SSS  M   M G     RRRR
    S   Y        S M   M G GGG R R
    S   Y        S M   M G   G R  R
SSSS    Y   SSSS  M   M  GGG  R   R




L       OOO    GGGG  III  N   N
L     O   O G        I    N   N
L     O   O G        I    NN  N
L     O   O G        I    N N N
L     O   O G GGG    I    N  NN
L     O   O G   G    I    N   N
LLLLL  OOO    GGG   III   N   N




Micro/RSX    V3.0      [1,10]LOGIN      - NO PAGE LIMIT      15-APR-85    17:07:43
Micro/RSX    V3.0      FORM #0 - NORMAL HARDWARE FORMS      15-APR-85    17:07:43
Micro/RSX    V3.0      NO IMPLIED FORM FEED                 15-APR-85    17:07:43
Micro/RSX    V3.0      JOB CONTAINS ONE FILE                15-APR-85    17:07:43
```

ZK-4072-85

Figure 5-3: File Flag Page

```
Micro/RSX    V3.0      DU0:[1,2]LOGIN.TXT;1                    15-APR-85    17:07:48
Micro/RSX    V3.0      COPY 1 OF 1                             15-APR-85    17:07:48
Micro/RSX    V3.0      DELETION NOT SPECIFIED                  15-APR-85    17:07:48




L      OOO   GGGG  III  N   N
L     O   O G       I   N   N
L     O   O G       I   NN  N
L     O   O G       I   N N N
L     O   O G GGG   I   N  NN
L     O   O G   G   I   N   N   ..
LLLLL  OOO   GGG   III  N   N   ..




TTTTT X   X TTTTT   ;;    1
  T   X   X   T     ;;   11
  T    X X    T           1
  T     X     T     ;;    1
  T    X X    T     ;;    1
  T   X   X   T      ;    1
  T   X   X   T     ;    111




Micro/RSX    V3.0      DU0:[1,2]LOGIN.TXT;1                    15-APR-85    17:07:48
Micro/RSX    V3.0      COPY 1 OF 1                             15-APR-85    17:07:48
Micro/RSX    V3.0      DELETION NOT SPECIFIED                  15-APR-85    17:07:48
```

ZK-4073-85

an entry number. You can use this entry number in QMG commands. See the next section.

```
$ PRINT/COPIES:20 RET
File(s)? PASHA.CMD RET
PRI -- Job 706, name "PASHA    ", submitted to queue "PRINT "
```

This example prints 20 copies of PASHA.CMD. They are printed end-to-end with no flag page except at the beginning.

```
$ PRI/COP:20/FLAG_PAGE PASHA.CMD RET
PRI -- Job 321, name "PASHA    ", submitted to queue "PRINT "
```

This example prints 20 copies of PASHA.CMD, each with a flag page.

```
$ PRIN/COPIES:2 DAVID.TXT, DAN.TXT RET
PRI -- Job 18, name "DAVID    ", submitted to queue "PRINT "
```

This example prints two copies of DAVID.TXT followed by two copies of DAN.TXT. It does not print two sets of one copy of each file. If you want to print sets of files, see the next example.

```
$ PRINT/JOB_COUNT:2 DAVID.TXT, DAN.TXT RET
PRI -- JOB 311, NAME "DAVID    ", submitted to queue "PRINT "
```

This example prints a set consisting of a copy of DAVID.TXT followed by a copy of DAN.TXT, and then prints another set of DAVID.TXT followed by DAN.TXT.

```
$ PRI/FLAG PASHA.CMD, RAJAH.CMD, SHEIK.CMD RET
PRI -- Job 999, name "PASHA    ", submitted to queue "PRINT "
```

This example prints one copy of each file, each with its own flag page.

```
$ PRI/NAME:ARABY/FLAG PASHA.CMD, RAJAH.CMD, SHEIK.CMD RET
PRI -- JOB 805, NAME "ARABY    ", submitted to queue "PRINT "
```

This example prints one copy of each file, each with its own flag page, and gives the name ARABY to the job as a whole. The name ARABY appears in the print queue and at the head of the entire printed output.

```
$ PRINT/LENGTH:60 GLADRAG.MAC RET
PRI -- Job 22, name "GLADRAG  ", submitted to queue "PRINT "
```

This example prints a single copy of the file GLADRAG.MAC. This is a source program file without form-feed characters in it. The user wants to print it on line-printer paper, a physical form 66 lines long. To avoid printing over the perforations, the user has specified a logical length of 60 lines. Each time 60 lines have been printed, the logical page is complete and the printer moves to the top of the next physical page.

If the file has form feeds in it, the printer moves to the top of a physical page each time it encounters a form feed and each time it prints 60 lines without encountering a form feed.

```
$ PRI/COPIES:20 MOB.COR, RIFF.COR, RAB.COR/COP:19, PRO.COR RET
PRI -- Job 77, name "MOB      ", submitted to queue "PRINT "
```

This example prints 20 copies each of MOB.COR and RIFF.COR, 19 copies of RAB.COR, and then 20 copies of PRO.COR.

```
$ PRI/AFTER:(4-JUL-84 13:13) CONST.CON RET
PRI -- Job 809, name "CONST    ", submitted to queue "PRINT "
```

This example blocks the job in its queue until the specified date and time. The file itself remains in its directory. Deleting the file from its directory does not take it out of the queue, but does prevent it from being printed.

```
$ PRI/AFTER:(17:00) GUNGA.DIN RET
PRI -- Job 765, name "GUNGA    ", submitted to queue "PRINT "
```

This example prints the file after 5 p.m. on the day the command is entered. If the command is entered after the specified time, the file is printed immediately.

Notes

P is the short form of PRINT.

The PRINT command does not accept the percent (%) wildcard. The asterisk (*) wildcard can only be used to substitute for a complete part of the file specification and **not** for part of a file name or file type as most of the file-related commands do.

For more information on the PRINT command, see Volume 2, Chapter 12. There are several additional qualifiers to PRINT described there.

See the next section for a description of the SHOW QUEUE command, which displays information about print jobs in queues.

PRINT is to print jobs as SUBMIT is to batch jobs.

Once your job is in a queue, you can modify some job attributes with SET QUEUE. See the next section.

The Queue Manager automatically deletes .PMT and .DMP files after they are printed.

The flag pages and the job pages have different formats for the banner text. The job name page shows the UIC the PRINT command was issued under on the first line and a 9-character job name (derived from an explicit job name or the file name of the first file in the job) on the second line. The flag page shows the full 9-character file name on the first line and the file type and version number on the second line. See Figures 5–2 and 5–3.

Entry numbers run from 1 through 999 and are unique. Various QMG commands permit you to identify your job either by this number or by job name. The job name includes your UIC. See the next section and also the expanded material in Volume 2, Chapter 12.

5.9.3 Queue Manager Commands

The PRINT command sends jobs to the line printer by way of the Queue Manager (QMG). The Queue Manager is a system task that also distributes jobs for batch processing. Passing jobs to the Queue Manager is called "queuing jobs."

The Queue Manager distributes print jobs queued by user tasks, by system tasks that output listings such as MACRO listings or task builder maps, and jobs queued by the PRINT command.

Once jobs are in queues, you can display queue information with the SHOW QUEUE command. You can alter the status of jobs in queues with the SET QUEUE command. You can hold jobs in queues with the HOLD /ENTRY command, release them from queues with the RELEASE/ENTRY command, or delete jobs in queues with the DELETE/ENTRY command.

Once you have issued a PRINT command, you may want to alter the way that file is processed or you may decide that you did not want to process that file at all. This section explains how to make these changes.

For example, you issue the following PRINT command to queue the file ANTRIM.TST to the line printer:

```
$ PRINT ANTRIM.TST RET
PRI - Job 22, name "ANTRIM   ", submitted to queue "PRINT "
```

However, you did not want to print that file. The file you wanted to print is ANTRIM.TXT. To prevent ANTRIM.TST from printing, you must remove that file from the queue. To do this, you use the SHOW QUEUE command and either the DELETE/ENTRY command or the STOP/ABORT command depending upon the status of your job.

The SHOW QUEUE command shows you the position of your job on the queue (as well as that of other jobs) and whether or not your job is currently being processed. The following example explains about SHOW QUEUE and the output it gives:

```
$ SHOW QUEUE [RET]
** PRINT QUEUES **
PRINT  =>  LPO
  [303,3]    ANTRIM     ENTRY:22                 ACTIVE ON LPO
       >     1 DUO:[MARK]ANTRIM.TST;1
LPO    =>  LPO
** BATCH  QUEUES **
BATCH  =>  BPAO
```

ANTRIM.TST is entry number 22 in the PRINT queue. The Queue Manager assigns an entry number to each job as a means for keeping track of which jobs are processed when. Usually, this is done on a first-come, first-serve basis. Also, note that ANTRIM.TST is currently being printed on line printer LP0: as shown in the third line of the display. To stop your currently printing job, you have to issue the STOP/ABORT command. This command stops your job on the line printer immediately.

Use the STOP/ABORT command in the following manner:

```
$ STOP/ABORT LPO: [RET]
$
```

When the dollar sign prompt ($) returns, your job has stopped printing.

If your job has not started printing, issue the SHOW QUEUE command as in the previous example. This time, the output this command gives is slightly different from before, notably, the ACTIVE ON LP0 is missing. For example:

```
$ SHOW QUEUE [RET]
** PRINT QUEUES **
PRINT  =>  LPO
  [303,3]    ANTRIM     ENTRY:22
             1 DUO:[MARK]ANTRIM.TST;1
LPO    =>  LPO
** BATCH  QUEUES **
BATCH  =>  BPAO
```

To delete this job from the PRINT queue, you use the DELETE/ENTRY command and the job's entry number (described in the previous example) as follows:

```
$ DELETE/ENTRY:22 [RET]
$
```

Your job has been removed from the PRINT queue and will not be printed.

Once ANTRIM.TST is removed from the PRINT queue, reissue the PRINT command with ANTRIM.TXT as the file you want processed.

The following sections detail the various commands that use the QMG, their qualifiers, and examples on how the commands are used.

5.9.3.1 DELETE/ENTRY

The DELETE/ENTRY command deletes jobs from a queue or files from a job by specifying the job's entry number.

Format

DELETE/ENTRY:nnn[/FILE_POSITION:n]

Command Qualifiers

/ENTRY:nnn

Deletes a queue entry by number. This number is always unique.

/FILE_POSITION:n

Identifies the file by the file's position in the job.

Examples

```
$ DELETE/ENTRY:301 RET
```

This command deletes the job from queue by referencing the job's unique entry number (in this example, job number 301).

```
$ DELETE/ENTRY:301/FILE_POSITION:2 RET
```

This example deletes only the second file appearing in job 301. You may only refer to the file you want to delete by the file's position in the print job (the numbered order in which you entered the file specification). You cannot specify the file you want to delete by referring to its file specification.

Note

You see the entry number for a job when you issue the PRINT command. You can also find the entry number, and the file position for files in the job, with the SHOW QUEUE command.

5.9.3.2 SHOW QUEUE and SHOW PROCESSOR

Use the commands described in this section to display information about queues, the jobs in the queues, and spooled devices.

An unadorned SHOW QUEUE command lists all jobs in all queues. Qualifiers can be used to limit the display to specific queues, specific jobs, and the like.

You may examine the contents of your system's queues (in decreasing degree of detail) by entering one of the following command qualifiers:

/FULL
/FILES
/BRIEF

5.9.3.3 SHOW QUEUE

SHOW QUEUE displays information about QMG print jobs.

Format

SHOW QUEUE [queuename][/qualifier[s]]

Command Qualifiers

/BRIEF
/DEVICE
/ENTRY:nnn
/FILES
/FORMS[:n]
/FULL
/OWNER_UIC:uic
/PRINT

Parameter

queuename

Displays QMG information for a single queue, either print or batch. The main print queue is called PRINT. Other print queues have names determined by the system manager.

Command Qualifiers

/BRIEF

Displays only queues, queue assignments, and jobs in queues.

You may examine specific attributes of queues by entering one of the following qualifiers.

/DEVICE

Limits information to print queues.

/ENTRY:nnn

Limits information to a particular job entry referenced by the job's unique entry number.

/FILES

Displays information about queues, queue assignments, jobs in queues, and files that compose jobs in queues. The attributes of the jobs are not displayed. This display format is the default of SHOW QUEUE.

/FORMS[:n]

Limits information to jobs that are to be printed on a specified form. If the argument n is omitted, the display shows all jobs that are other than Form 0.

/FULL

Displays detailed information about queues, queue assignments, jobs, the attributes of jobs in queues, and files that compose jobs in queues.

/OWNER_UIC:uic

Limits information to jobs owned by the specified uic.

/PRINT

Limits information to Print queues. This qualifier produces the same information as the /DEVICE qualifier.

Examples

The following examples show the three different output displays (FULL, FILE, BRIEF).

```
$ SHOW QUEUE/FULL RET

** PRINT QUEUES **
PRINT  => LPO
  [7,25] MARK       ENTRY:22      FORM:0      ACTIVE ON LPO
      PRI:50  LEN:0  PAGE:0  NORESTART  FLAG:JOB  NOLOWER  COP:1
        > 1 DUO:[GREGG]MARK.LST;1     COP:1        NODELETE
  [7,40] ANITA      ENTRY:24      FORM:0
      PRI:50  LEN:0  PAGE:0  NORESTART  FLAG:JOB  NOLOWER  COP:1
          1 DUO:[LAKE]ANITA.MEM;12    COP:1        NODELETE
LPO    => LPO
** BATCH QUEUES **
BATCH  => BAPO
$
```

This display shows all information about the queues and jobs in queues. In this example, the queue PRINT sends jobs to LP0:.

There are two print jobs in the queue PRINT:

- [7,25]MARK, entry number 22, which is currently being printed on LP0:
- [7,40]ANITA, entry number 24, which is waiting to be printed

The attributes of the job are indicated in the display, and the files that make up each job are also listed. The angle bracket (>) indicates which file of a job is currently being processed.

```
$ SHOW QUEUE/FILE RET
** PRINT QUEUES **
PRINT  => LP0
  [7,25] MARK      ENTRY:22                 ACTIVE ON LP0
       > 1 DU0:[GREGG]MARK.LST;1
  [7,40] ANITA     ENTRY:24
         1 DU0:[LAKE]ANITA.MEM;12
LP0    => LP0
** BATCH QUEUES **
BATCH  => BAP0
```

This display, the default format, shows the queues in the same state. It does not show the attributes of the job. It only shows the form number of the job and the attributes of the files if other than the default values were specified. Because /FILE is the default, you need not specify it.

```
$ SHOW QUEUE/BRIEF RET
** PRINT QUEUES **
PRINT  => LP0
  [7,25] MARK      ENTRY:22                 ACTIVE ON LP0
  [7,40] ANITA     ENTRY:24
LP0    => LP0
** BATCH QUEUES **
BATCH  => BAP0
```

This display only shows the names, entry numbers, and status of the jobs in the queues. Also, if a form other than Form 0 is specified for a print job, the form will appear in the display.

5.9.3.4 SHOW PROCESSOR

SHOW PROCESSOR displays information about the initialized characteristics of spooled devices and batch processors, printers, and other output devices under control of the Queue Manager.

Format

SHOW PROCESSOR [processorname]
Displays information about all processors.

SHOW PROCESSOR/PRINT or SHOW PROCESSOR/DEVICE
Shows all print processors.

Example

Enter the following command on your terminal:

```
$ SHOW PROCESSOR [RET]
```

Information about print processors, batch processors, and spooled input devices will be displayed at your terminal:

```
** SPOOLED DEVICES **
LP0    <= PRINT LP0
          LOWER      FORM:0   FLAG:2
          CURRENT JOB:  [CRAZED]FUP      ENTRY:59
** BATCH PROCESSORS **
BAP0   <= BATCH
```

This display shows all information about spooled devices and batch processors. In this example, spooled device LP0: can receive jobs from queues PRINT and LP0:.

The only batch processor, BAP0, can receive jobs only from the queue BATCH.

LP0: is currently printing job [CRAZED]FUP, entry number 59.

5.9.3.5 SET QUEUE

SET QUEUE modifies attributes given to either print jobs, or files that compose jobs in queues. Such jobs and files have been entered in queues by the PRINT command. You cannot change the attributes of an active job.

Format

SET QUEUE/ENTRY:nnn/qualifier[/qualifier[s]]

Command Qualifiers

/AFTER:(dd-mmm-yy hh:mm)
/FORMS:n
/JOB_COUNT:n
/LENGTH:n
/PRIORITY:n

File Format

SET QUEUE/ENTRY:nnn/FILE_POSITION:n/qualifier[/qualifier[s]]

File Qualifiers

/COPIES:n

Command Qualifiers

/AFTER:(dd-mmm-yy hh:mm)

Changes the time after which your job will be printed. The job will be BLOCKED until the time and date you specify. The job will not necessarily be printed at exactly the time you state but will be eligible after the time you state.

If you do not supply the calendar field, the default is the current date. If you do not supply the clock field, the default is midnight on the date given in the calendar field.

If you supply both the clock and calendar fields, you must separate them with a space.

/FORMS:n

Changes the FORMS attribute of your print job. See Section 5.9.2 for an explanation of the /FORMS qualifier to the PRINT command.

/JOB_COUNT:n

Changes the number of copies of a job you want to have printed. See File Qualifiers for an explanation of how to modify the number of copies of a file contained inside of a job. See Section 5.9.2 for an explanation of the /COPIES qualifier to the PRINT command.

/LENGTH:n

Changes the length of a logical page (number of lines per printed page) of your print job. A line printer will move to the top of a physical page every time n lines have been printed on a page or a form feed is encountered . See Section 5.9.2 for an explanation of the /LENGTH qualifier to the PRINT command.

/PRIORITY:n

Changes the queue priority of a print job. Nonprivileged users may set priorities up through 150. Privileged users may set priorities up through 250. See Section 5.9.2 for an explanation of the /PRIORITY qualifier to the PRINT command.

File Qualifiers

/FILE_POSITION:n

Changes the operation of /COPIES:n or /[NO]DELETE as they apply to a file contained in a job in queue. The number n refers to the file position in the job. Use SHOW QUEUE to determine its position.

/COPIES:n

Changes the number of copies of a file you want to have printed within a single printing of a print job.

Examples

```
$ PRINT/JOB_COUNT:2 JILL.TXT,JOHN.TXT/COPIES:3 RET
```

The SHOW QUEUE/FULL command output might look like this:

```
** PRINT QUEUES **
PRINT  => (LPO)
  [304,1]    JILL   ENTRY:696    FORM:0
      PRI:50  LEN:0  PAGE:0  NORESTART  FLAG:JOB  NOLOWER  COP:2
            1 DU1:[DICK]JILL.TXT;1      COP:1       NODELETE
            2 DU1:[DICK]JOHN.TXT;1      COP:3       NODELETE
LPO    => LPO
```

To print two copies of the file JOHN.TXT instead of the three copies specified in the initial print command, you would type the following SET QUEUE command on your terminal and enter it:

```
$ SET QUEUE/ENTRY:696/FILE_POSITION:2/CO:2 RET
```

The SHOW QUEUE/FULL command output on your terminal then would look like this:

```
** PRINT QUEUES **
PRINT  => (LPO)
  [304,1]   JILL   ENTRY:696    FORM:0
      PRI:50  LEN:0  PAGE:0  NORESTART  FLAG:JOB  NOLOWER  COP:2
            1 DU1:[DICK]JILL.TXT;1      COP:1       NODELETE
            2 DU1:[DICK]JOHN.TXT;1      COP:2       NODELETE
LPO    => LPO
```

Note that the file in position 2, JOHN.TXT, will have two copies instead of three printed.

Notes

When you issue the PRINT command, you specify attributes of the QMG job through command qualifiers. SET QUEUE command qualifiers change the matching attributes.

You cannot change the attributes of an active job.

You can delete files or change the number of copies to be printed by specifying /FILE_POSITION:n in your SET QUEUE command line.

5.9.3.6 STOP/ABORT

The STOP/ABORT command stops the current job on a line printer immediately. Privileged users can stop any job. Nonprivileged users can stop their own jobs.

Format

STOP/ABORT   printer

Parameter

printer

Specifies the line printer whose active job you want to stop.

Example

```
$ STOP/ABORT LPO: RET
$
```

This command stops the currently active print job on LP0:. This job is deleted from the queue and the next eligible job is queued.

Note

As soon as the active job is deleted, QMG passes the next eligible job to the processor. The processor has not been aborted or killed, only the active job on that processor.

5.9.3.7 HOLD/ENTRY and RELEASE/ENTRY

You can specify that a print job be HELD in a queue when you issue the PRINT command, or you can hold it with the HOLD command.

You can release such jobs with the RELEASE command.

Note that when a system crashes, the Queue Manager automatically holds those jobs in the queues so that the jobs are not lost. However, when the system is restored, you must release those held jobs with the RELEASE command.

Format

HOLD/ENTRY:nnn
Blocks a job in its queue until it is explicitly released.

RELEASE/ENTRY:nnn
Unblocks a job that has been held in queue.

Examples

```
$ PRINT/HOLD/NAME:MARY MUSE.TXT RET
```

The SHOW QUEUE/FULL command output might look like this:

```
** PRINT QUEUES **
PRINT  => LP0
  [231,56]   MARY   ENTRY:20    FORM:0     HELD
      PRI:50  LEN:0  PAGE:0  NORESTART  FLAG:JOB  NOLOWER  COP:1
        1 DU1:[AMUSED]MUSE.TXT;1        COP:1       NODELETE
LP0    => LP0
```

To unblock MUSE.TXT, you would type the following RELEASE command on your terminal:

```
$ RELEASE/ENTRY:20 RET
```

The SHOW QUEUE/FULL command output would look like this:

```
** PRINT QUEUES **
PRINT  => LP0
  [231,56]   MARY   ENTRY:20    FORM:0     ACTIVE ON LP0
```

```
     PRI:50  LEN:0  PAGE:0  NORESTART  FLAG:JOB  NOLOWER  COP:1
       >   1 DU1:[AMUSED]MUSE.TXT;1    COP:1      NODELETE
LP0    => LP0
```

Note

The HELD status of the job in the previous example was first changed to WAITING, and then, since there were no jobs in queue, became ACTIVE.

5.9.4 UNLOCK

UNLOCK unlocks locked files. Locked files are files that have been improperly closed because a task aborted or stopped execution while the file was open. Locked files are identified by an L in the directory listing.

Format

UNLOCK[/qualifier[s]]
File(s)? filespec[s]

UNLOCK[/qualifier[s]] filespec[s]

Command Qualifiers

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec

Parameter

filespec

Identifies the locked file(s) that you want to unlock.

Example

```
$ Directory TOPEKA.KAN [RET]
Directory DB0:[303,5]
29-MAY-85 13:13

TOPEKA.KAN;3    32.     L     25-MAY-85 12:29

Total of 32./40. blocks in 1. files
$
$
$ UNLOCK TOPEKA.KAN [RET]
```

```
$
$
$ DIRECTORY TOPEKA.KAN [RET]
Directory DBO:[303,5]
29-MAY-85 13:14

TOPEKA.KAN;3     32.          25-MAY-85 12:29

Total of 32./40. blocks in 1. files
```

In this example, the user found a locked file, perhaps through an error message stating that the file could not be opened. The locked state is indicated by the L in the third column of the directory listing. After the user issues the UNLOCK command, the file is no longer locked.

Notes

In many cases, a locked file has no contents because the task that opened the file aborted before writing to the file. In this case, you do not need to unlock the file before deleting it.

Another common case of file locking is a data file that has been opened by some task that aborted or exited without closing the file. In this case, you have to unlock the data file before running the task again. If the file is written to by the task, it may have been corrupted.

5.10 SET and SHOW

You can type SET commands to establish your default device, directory, and User Identification Code, and SHOW commands to display what these defaults are. You can change the protection code for any existing file in your directory with the SET PROTECTION command. In addition, you can establish your own default file protection code with SET PROTECTION/DEFAULT and use SHOW PROTECTION to display the default protection.

See Section 5.9.3 for SET and SHOW commands for the Queue Manager.

5.10.1 SET DEFAULT

SET DEFAULT changes where the operating system looks for information from your current device and directory to those indicated in the command. SET DEFAULT establishes your default device or directory, or both. That is, when you do not indicate a device or a directory in a file specification or command, the operating system supplies these defaults.

The protection codes for the directory and its files control your activity in the directory. If you are a nonprivileged user, you have the directory and file access rights of the WORLD category, and—depending upon your User Identification Code (UIC)—the GROUP category. If you are a privileged user, you have the access rights of the SYSTEM category. Of course, if your UIC matches the directory owner's, then you have the privileges of the OWNER category.

Note that a privileged user can change his UIC with the SET UIC command. If you intend to create or alter files in a directory other than your own, it is good practice to change your UIC to match the directory owner's. See Section 5.10.3.

SET DEFAULT works differently if you previously set your terminal's default to /NONAMED_DIRECTORY. See Volume 2, Chapter 12 for more information.

Format

SET DEFAULT[/qualifier] [default]]

Command Qualifier

/[NO]NAMED_DIRECTORY

Parameters

default

Specifies a device and/or directory or logical name to be used as the default device or directory in file specifications.

If you specify a physical device name, terminate the device name with a colon.

If you specify a directory name, you must enclose it in the square brackets. (Note that if your terminal's default is /NONAMED_DIRECTORY, the directory field accepts only numbered directories in the form [g,m].)

If you specify a logical name, the logical name must translate to an existing physical device and/or directory. If the logical name is to translate to a physical device, be sure that when you make the logical name assignment, you terminate the physical device name with a colon. When you issue the SHOW DEFAULT command, the system displays the actual physical device, not the logical name.

Command Qualifier

/[NO]NAMED_DIRECTORY

SET DEFAULT/NAMED_DIRECTORY is the default on Micro/RSX. This command allows SET DEFAULT to accept either named or numbered directories. The format for a named directory is [directory], where directory is one to nine of the following characters: the 26 letters from A through Z, and the numbers from 0 through 9. Examples of named directories include [MAG], [DIDDLYBOP], and [1POTATO2].

The format for a numbered directory is [g,m], where g and m are octal numbers from 1 through 377. Examples of numbered directories are [100,2] and [202,10].

If you issue SET DEFAULT/NONAMED_DIRECTORY, you can only specify numbered directories in the SET DEFAULT command. Operating with /NONAMED_DIRECTORY has some subtle side effects. Its use is recommended only for experienced users of other systems in the RSX family who need to maintain compatibility with older versions.

Some application programs do not recognize named directories. See Volume 2, Chapter 12 for some suggested solutions to this problem.

Examples

```
$ SET DEFAULT DU1:[GRAMPUS] RET
```

This example sets DU1: as the default device and [GRAMPUS] as the default directory. All subsequent commands default to this device and directory unless you explicitly state otherwise.

```
$ SET DEF DU1: RET
$ ASSIGN DU1: SY: RET
```

The two commands in this example are equivalent. In each case, the default device becomes DU1:. When you set the default device, SET DEFAULT actually assigns SY as a logical name for the device you want. SY is the logical device name that represents your default device.

5.10.2 SHOW DEFAULT

SHOW DEFAULT displays the current default device and directory for your terminal. It also shows whether you are operating with /NAMED_DIRECTORY or /NONAMED_DIRECTORY as your default, and your User Identification Code (UIC).

Format

SHOW DEFAULT

Example

```
$ SHOW DEFAULT [RET]
DU1:[HUSKY]   Named   TT6:
Protection UIC:   [15,20]
```

This example shows the display from SHOW DEFAULT, giving the default device, DU1:, and directory, [HUSKY], for your terminal, TT6:. Named indicates that your default is /NAMED_DIRECTORY—that is, you can specify either a named or numbered directory in all file specifications and commands. Your User Identification Code (UIC) identifies you to the operating system and controls your ability to access files.

5.10.3 SET UIC

SET UIC is a privileged command that changes your User Identification Code (UIC). A UIC is an attribute of a directory and its files, establishing ownership. If you intend to create or alter files in a directory other than your own, you can use this command to change your UIC to that of the directory's owner.

A UIC controls the protection codes of both a directory and its files. These protection codes determine who can access the directory and its files, and in what ways. See Section 5.6.1 for a description of how UICs relate to file protection codes.

If you change directories, but do not change your UIC to match that of the directory's owner, the following problems can occur:

- If you create a file in the directory of a nonprivileged account—that is, in a directory with a UIC group number greater than 10—the owner of that directory cannot easily change or delete that file. Most file protection codes restrict access in the WORLD category to READ, which is the operating system default for this category. Because your

UIC owns the file, the owner of the directory may fall into the WORLD category, and therefore can only read the file.

- Several system command files, as well as the command files of some users, set the UIC of the file's owner to match the UIC of the directory's owner. This matching of UICs gives the command file full access to all the files in the directory. If the files in that directory are owned by a different UIC, the command file may fail.

Note that SET UIC does not change your directory. Typically, after executing this command, you issue the SET DEFAULT command to change your directory to the one belonging to the owner of the new UIC.

SET UIC acts differently if your terminal's default is /NONAMED_DIRECTORY. See Volume 2, Chapter 12 for more information.

Format

SET UIC [uic]

Parameter

[uic]

Specifies the User Identification Code (UIC) that you want to make your default. The format of a UIC is [g,m], where the first number is the group number, and the second is the member number. Group and member numbers are octal and range from 1 through 377.

Group numbers indicate privilege status. The UICs of privileged accounts have group numbers of 10 or less.

Examples

```
$ DIRECTORY/FULL [000000]GIDEON.DIR RET
Directory DU0:[000000]
12-MAY-85 14:12

GIDEON.DIR;1        (576,5040)     1./1.     C 12-MAY-85 07:28
  [200,1]  [RWED,RWED,RWED,R]

Total of 1./1. blocks in 1. file
$
$ SET UIC RET
UIC? [200,1] RET
$
$ SET DEFAULT [GIDEON] RET
```

Assume that you want to create or alter files in [GIDEON], another directory located on your current device. A full listing of the directory file itself, located in [000000]GIDEON.DIR, shows that the UIC of the directory's owner is [200,1]. You then issue the SET UIC command to change your current UIC to that of the directory's owner. Finally, you type the SET DEFAULT command to move to the other directory.

Notes

As a privileged user, you can manipulate the files of another directory in any way. Note that you need only issue the SET UIC command if you intend to create or alter files (WRITE) in another directory. However, when you are working with the files of another directory, it is good practice to always use this command.

If you change your UIC to that of a nonprivileged account, you can change back to a privileged account by issuing the SET UIC command again.

You can type either the SET UIC or the SET DEFAULT command first; the sequence does not matter.

5.10.4 SHOW UIC

SHOW UIC displays your User Identification Code (UIC). Your UIC is unique and identifies you to the operating system. In addition, your UIC determines whether you are a privileged or nonprivileged user.

Each file and directory has a UIC associated with it, which identifies the owner. Each file and directory also has a protection code. The way in which your UIC relates to both the UIC and protection code of a file or directory controls your ability to access that file or directory. See Section 5.6 for a description of protection.

Nonprivileged users always have the same UIC, but privileged users can change their UIC with the SET UIC command.

Format

```
SHOW UIC
```

Example

```
$ SHOW UIC [RET]
Protection UIC: [303,17]
```

This example shows the display from SHOW UIC. The first number is the group number, and the second is the member number. Group and member numbers are octal and range from 1 through 377.

Generally, all users working on a particular project have the same group number. Each user account has a unique member number, however.

Group numbers indicate privilege status. The UICs of privileged accounts have group numbers of 10 or less. The user issuing this SHOW UIC command, therefore, is nonprivileged.

5.10.5 SET PROTECTION

SET PROTECTION changes the protection code of an existing file or files.

The protection code determines which users may access a file, and in what ways.

Nonprivileged users can change the protection of any files in their own directory. Privileged users can change the protection of any file.

Format

SET PROTECTION:(code)[/qualifier[s]] filespec[s]

SET PROTECTION[/qualifier[s]]
File? filespec[s]
Code? (code)

SET PROTECTION[/qualifier[s]] filespec[s] (code)

Command Qualifiers

/DATE:dd-mmm-yyy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec

Parameters

filespec[s]

Specifies the file or files for which you are changing the protection. Multiple file specifications, separated by commas, and wildcards are permitted.

(code)

Specifies which user categories can access a file and what each user category may do to the file. The parentheses are required.

There are four categories of users in a file protection code, as follows:

SYSTEM	The operating system itself, and privileged users, those with group numbers of 10 or less.
OWNER	The user having the same UIC as the one the file was created under.
GROUP	All users with the same UIC group number as the one the file was created under.
WORLD	All other users.

There are also four kinds of access to files as follows:

READ	The user, or the user's tasks, may read, copy, print, type the file, or run it, if it is a task image.
WRITE	The user, or the user's tasks, may add new data to the file by writing to it.
EXTEND	The user, or the user's tasks, may increase the amount of disk space allocated to the file.
DELETE	The user, or the user's tasks, may delete the file.

The system default protection code is expressed as follows:

```
(SYSTEM:RWED,OWNER:RWED,GROUP:RWED,WORLD:R)
```

Under this code, the system, and privileged users, have full access to your files. You, as well as others with your group number, also have full access to your files. Other nonprivileged users can only read your files.

All files have the system default protection, unless you change their protection code using the SET PROTECTION command, or establish an alternative default protection using the SET PROTECTION/DEFAULT command.

You specify the code in the SET PROTECTION command using the same format as shown in the previous example. Note, however, that you need to name only the user groups whose access rights you want to change and the types of access you want to grant to those groups. If you want to deny all access to a group, simply name the group and omit the colon (:) and the code for the access form.

Examples

```
$ SET [RET]
FUNCTION? PROTECTION [RET]
CODE?(SYSTEM:RWED,OWNER:RWED,GROUP,WORLD) [RET]
File(s)? SANTONE.TEX [RET]
$
$ SHOW PROTECTION SANTONE.TEX [RET]
System:RWED, Owner:RWED, Group:No Access, World:No Access
```

This example changes the protection of the file SANTONE.TEX so that privileged users and the owner have full access to the file and all others have no access to the file. The SHOW PROTECTION command displays the protection of the file. Issuing the SHOW PROTECTION command is a good way of checking that you have changed the file's protection code correctly.

```
$ SET PROTECTION *.*;* (SYS:RWE,OWN:RWE,GRO:R,WOR:R) [RET]
```

This example sets the protection for all files in the current default directory. No class of user can delete the files in the directory; other nonprivileged users cannot alter files in the directory.

Notes

You must have access to the volume, the directory, and the file before you can access a file. See Section 5.6.1 for more information.

Although you can protect your files against access by privileged users, privileged users can change the protection status of any file. Thus, you can prevent privileged users from inadvertently accessing your files, but you cannot prevent them from deliberately changing the protection status and doing what they will with your files.

If you deny READ access to a task image file, the task cannot be run.

Other commands accepting a protection code include INITIALIZE, INITIALIZE/UPDATE, MOUNT, and CREATE/DIRECTORY.

Other commands with qualifiers associated with file ownership include:

- INITIALIZE/OWNER_UIC—See Volume 2, Chapter 13.
- INITIALIZE/UPDATE/OWNER_UIC—See Volume 2, Chapter 13.
- INSTALL/UIC—See Volume 2, Chapter 15.
- LINK/OPTION:UIC—See Volume 2, Chapter 14 and the *RSX-11M /M-PLUS and Micro/RSX Task Builder Manual* (which is available separately and as part of the Advanced Programmer's Kit).
- MOUNT/OWNER—See Volume 2, Chapter 13.
- RUN/UIC—See Volume 2, Chapter 15.
- SHOW QUEUE/OWNER_UIC—See Volume 2, Chapter 12.
- SHOW TASK/DYNAMIC/OWNER—See the Advanced Programmer's Kit.

5.10.6 SET PROTECTION/[NO]DEFAULT

SET PROTECTION/DEFAULT establishes your personal default protection code for all files that you create after issuing this command. A file protection code controls the types of access that other system users have to your files.

Note that this command establishes your default file protection only for the current terminal session. To set your default file protection for all future terminal sessions, place the SET PROTECTION/DEFAULT command in your LOGIN.CMD file.

SET PROTECTION/NODEFAULT removes your personal default file protection. After issuing this command, the files you create receive the volume default protection. Although a volume can have any default file protection, this default is usually the same as the system default of (SYSTEM:RWED,OWNER:RWED,GROUP:RWED,WORLD:R).

To change the protection code for existing files, see the description of SET PROTECTION in Section 5.10.5.

Format

```
SET PROTECTION:(code)/DEFAULT
SET PROTECTION/NODEFAULT

SET PROTECTION/DEFAULT
Code?  (code)
```

Parameter

(code)

Specifies which user categories are permitted access to the file and what each user category may do to the file. The parentheses are required.

The file protection code has four user categories and four types of access allowed for each user category. See Section 5.10.5 for a description of the protection code format.

The first time you issue SET PROTECTION/DEFAULT in a terminal session, you must specify all four user categories. If you want to deny access to a category, omit the colon and the access code after the category name.

If you issue SET PROTECTION/DEFAULT again during the terminal session, you only need to specify the categories for which you want to change the access code. When you omit a user category in the protection code, you keep the current default protection for that category.

Examples

```
$ SET PROTECTION/DEFAULT RET
CODE?(SYSTEM:R,OWNER:RWED,GROUP:R,WORLD) RET
```

This example establishes your default file protection. Because you are issuing this command for the first time in the terminal session, you specify all four user categories. The code limits users in the SYSTEM and GROUP categories to READ access. You retain the standard volume default protection for the OWNER category of READ, WRITE, EXTEND, and DELETE. You deny all file access to the WORLD category by omitting the colon and access code after the category name.

```
$ SET PROTECTION:(GROUP)/DEFAULT RET
```

Assume that you type this command later in the same terminal session that you executed the command in the first example. Here you change your default protection for the GROUP category, denying these users any access to your files. Your default protection for the other three user categories remains unchanged. Note that you only needed to specify the user category for which you wanted to change the file access.

```
$ SET PROTECTION/NODEFAULT RET
```

This command removes your personal default file protection. The files that you create after issuing this command receive the volume default protection.

Notes

If you place SET PROTECTION/DEFAULT in your LOGIN.CMD file, you must specify all four user categories.

The system manager can use the Account File Maintenance Program (ACNT) to establish a default file protection for your account. This method has the same effect as placing SET PROTECTION/DEFAULT in your LOGIN.CMD file: both methods establish your default file protection for each terminal session. Note, however, that you can modify your default file protection for any portion of a terminal session by typing the SET PROTECTION/[NO]DEFAULT command.

If both SET PROTECTION/DEFAULT and ACNT set your default file protection, then the protection indicated in SET PROTECTION/DEFAULT is your default.

SET PROTECTION/NODEFAULT removes your personal default protection, regardless of whether SET PROTECTION/DEFAULT or ACNT established it.

Error Messages

Illegal user default protection code

Explanation: You did not specify all four user categories when issuing SET PROTECTION/DEFAULT the first time during a terminal session.

User Action: Reenter the command line, specifying all four user categories in the protection code.

No protection specified for any field.

Explanation: You attempted to set your default file protection code to deny file access to all four user categories—that is, you specified a file protection code that looks like (SYSTEM,OWNER,GROUP,WORLD).

User Action: The minimum default file protection code is READ access for one user category. Reenter the command line, specifying at least the minimum file access.

5.10.7 SHOW PROTECTION

SHOW PROTECTION displays your personal default file protection code.

Your default file protection can be established in two ways: either by issuing the SET PROTECTION/DEFAULT command, or by using the Account File Maintenance Utility (ACNT) to enter a protection code for your account. See the description of SET PROTECTION/DEFAULT in Section 5.10.6.

If you do not set your own default file protection, then SHOW PROTECTION issues the following message:

```
No user default protection specified.
```

Format

SHOW PROTECTION

Example

```
$ SHOW PROTECTION RET
System:RWED, Owner:RWED, Group:R, World:No Access
```

This example shows the display from SHOW PROTECTION, indicating that your default file protection grants full access to the SYSTEM and OWNER categories, READ access to the GROUP category, and no access to the WORLD category.

Chapter 6

Devices and Volumes

Peripheral devices are the actual hardware elements that, with the computer, make up a computer system. In general, a peripheral device is anything that is not part of the CPU or main memory. Peripherals include terminals, disks, and line printers.

Peripheral devices handle all input to and output from the system. One primary function of the operating system is to manage efficiently all the peripheral devices in the system.

On Micro/RSX systems, the primary peripheral devices are the terminals, the fixed disk, and the cassette drives, but many Micro/RSX systems also have magnetic tape devices or other disks, particularly RL02 drives.

In Micro/RSX terminology, a file is an owner-named area on a volume. A volume is a collection of files stored on a magnetic medium, such as a disk or magnetic tape. This medium must be physically placed on a drive and logically mounted before the system can access the data on the volume.

This chapter briefly introduces the important concepts concerning devices and volumes. In addition, the chapter describes the DCL commands that deal with devices and volumes. See Volume 2, Chapter 13, More About Devices and Volumes, for more information.

6.1 Devices on Micro/RSX Systems

Table 6–1 lists the devices found on Micro/RSX systems. Each installation has a different configuration of physical devices. Each physical device has an associated hardware controller, which serves as an interface between the device hardware and the CPU. In addition, each device has a device driver, which is the software interface between the operating system and the device controller.

The table also lists pseudo devices found on Micro/RSX systems. A pseudo device is an entity the system or user treats as an I/O (input/output) device, although it is not actually any particular physical device. A pseudo device name is a pseudonym through which actual physical devices can always be reached. See Volume 2, Chapter 13 for more information on pseudo devices.

Individual devices are identified by a 2-letter mnemonic and an octal unit number, terminated by a colon. If you omit the unit number, the system defaults to unit number 0. That is, if you want to specify DU0: in a command, DU: will do.

Table 6–1: Devices on Micro/RSX Systems

Mnemonic	Device
CL:	Console listing pseudo device
CO:	Console output pseudo device
DD:	DECtape II (TU58)
DL:	RL01/RL02 disk drives
DU:	RA60/RA80/RA81 disk drives
	RC25 disk drive
	RD51/RD52 disk drives
	RX50 diskette drive
DY:	RX02 diskette drives
LB:	System default pseudo device
LP:	Line printer
MS:	TK25/TS05/TS11 tape drives

Table 6–1 (Cont.): Devices on Micro/RSX Systems

Mnemonic	Device
MU:	TK50 tape drive
NL:	Null device
RD:	Reconfiguration driver
SP:	Spooling pseudo device
SY:	User default pseudo device
TI:	Terminal input pseudo device
TT:	Terminal
VT:	Virtual terminal
WK:	Workfile pseudo device

Micro/RSX systems support a variety of devices including the terminal, line printer, disk, and tape unit. The line printer is called a unit record device. Disks and magnetic tapes are mass-storage devices. Disks are random-access devices and magnetic tapes are sequential-access devices. Each of these device types is discussed in more detail elsewhere in this chapter and in Volume 2, Chapter 13.

Devices are informally identified by the 2-letter mnemonic identifying the device driver (such as DL: for RL02), but you should remember that the driver may support more than one device. For instance, the DU: driver supports the RD51 and RD52 fixed disks and the RX50 diskettes.

Magnetic tapes are a sequential-access medium. Random-access media, such as disks, are used when speed is the most important virtue. Magnetic tapes are used when economy and transportability are more important than speed.

The Micro/RSX Base Kit includes a magnetic tape device driver, but all other tape software is on the Micro/RSX Advanced Programmer's Kit. Tape use on the Base Kit is limited to tapes used for backup and tapes mounted as foreign volumes. See the *Micro/RSX System Manager's Guide* for more information on backing up files and volumes.

On Micro/RSX systems, the information contained on the magnetic medium is called a volume. To be used directly on the system, disk volumes must be in Files–11 format. With Files–11 volumes, you can list directories, type files, create new files, and so forth. All other disk volumes are considered foreign. To be used directly on the system, magnetic tape volumes must be ANSI standard. All other tape volumes are considered foreign.

The /FOREIGN qualifier to the MOUNT command (Section 6.2) allows you access to foreign volumes. Before being initialized, tapes and disks are foreign; they must be mounted foreign to be initialized. For information on initializing disk volumes, see the INITIALIZE command in Section 6.4.

For information on initializing tapes, see Volume 2, Chapter 13. Unless you have the Micro/RSX Advanced Programmer's Kit, you can only mount tapes foreign or use them for backup.

6.1.1 Physical Devices, Pseudo Devices, LUNs, and Logical Names

The purpose of devices of any kind is to handle I/O from tasks. Micro/RSX systems offer you a number of ways to name and access physical devices. You can directly access to the physical devices by naming them in commands or file specifications. However, you can use several forms of indirect access to physical devices as well.

The operating system often accesses physical devices as pseudo devices. Tasks access physical devices through logical unit numbers (LUNs). LUNs establish a relationship between the I/O done by the task and the devices on the system that the task needs to access. Finally, you can give any physical device a logical name. See Volume 2, Chapter 13, More About Devices and Volumes, for more information.

6.1.2 Summary of Device and Volume Use Commands

All volumes, Files–11 or foreign, must be mounted for any access. Foreign volumes are mounted with the MOUNT/FOREIGN command.

Multiple users can mount a volume on a shareable device. The volume remains mounted until all users who have mounted it dismount it.

Ownership of a device on a system with multiuser protection is of four kinds:

- A *private volume* is a volume mounted on a private device. A private device is a device that has been allocated, or mounted with the /NOSHAREABLE qualifier, and can be mounted by the owner only.

- A *shareable volume* is a volume mounted /SHAREABLE by the first user to mount it and can be mounted by any other user who knows the volume label. The volume stays mounted until the last user to mount it dismounts it.
- A *public volume* is a volume mounted on a public device. Devices are made public by the privileged SET DEVICE/PUBLIC command or the privileged /PUBLIC qualifier to the MOUNT command. The system owns public devices. Mounting a public device means other users can access the device without mounting it. Only a privileged user can set a device public or nonpublic. Public devices cannot be allocated.
- An *unowned volume* is a volume on a device that has not been allocated or mounted by anyone. An unowned volume can be mounted by anyone.

Any user who mounts a volume has full access to that volume within the limits of privilege and volume and file protection. Privileged users have privileged access. Nonprivileged users have nonprivileged access. However, nonprivileged users do have privileged access to volumes mounted on their private (allocated) devices.

A mounted device cannot be set public or nonpublic, and cannot be allocated or deallocated.

The LOGOUT command issued by any user automatically dismounts any volumes mounted from that terminal and deallocates any devices allocated from that terminal. A mounted public device is not dismounted by LOGOUT, however.

6.2 MOUNT

MOUNT declares a volume to be known to the system, and sets the volume on-line, and ready for use. Some qualifiers can be used with any MOUNT command; some are limited to mounting disks and others are limited to mounting magnetic tapes. All tape qualifiers to MOUNT are described in Volume 2, Chapter 13, but unless you have the Advanced Programmer's Kit you can only mount tapes foreign or use them for backup.

Format

For Disks and Other Random-Addressable Devices

MOUNT[/qualifier[s]] ddnn: volumelabel

Command Qualifiers

/FILE_PROTECTION:(code)
/FOREIGN
/OVERRIDE:IDENTIFICATION
/OWNER:[uic]
/PROTECTION:(code)
/PUBLIC
/[NO]SHAREABLE
/SYSTEM (synonym for /PUBLIC)
/[NO]WAIT
/[NO]WRITE

Parameters

ddnn:

Specifies the device on which the volume is to be mounted. You can mount only one disk or other random-addressable device.

volumelabel

Specifies the volume label, that is, the name associated with the volume. Volume labels are mandatory for nonprivileged users. You must supply a volume label for each volume you want to mount. Disk and random-addressable volume labels can be as many as 12 characters.

Volume labels (or identifiers) can include any alphanumeric character without restriction. If you are mounting an ANSI-standard magnetic tape, you can also specify nonalphanumeric symbols such as the following:

```
<space> ! " % _ ' () * + , - . / : < = > ?
```

Labels including these characters must be enclosed in quotation marks ("). If the label includes the quotation mark itself ("), the quotation mark must be followed by another quotation mark. The extra quotation marks do not count in figuring the length of the label. See the examples.

Note that the at sign (@), semicolon (;), and dollar sign ($) are not accepted in volume labels.

Command Qualifiers

/FILE_PROTECTION:(code)

Specifies the default file protection for any new files created on the volume while it is mounted. The file protection code is enclosed in parentheses. See Section 5.6 for more information on file protection codes and the relationship between file protection and volume protection. See also the /PROTECTION qualifier.

The default file protection can be overridden by specifying another protection when the file is created.

If this qualifier is not included, the value specified at the time the volume was initialized is applied.

/FOREIGN

Specifies that the volume being mounted is not in Files–11 format (if a disk volume) or not in ANSI format (if a tape volume). This qualifier is required for foreign volumes. Note that before a tape or disk has been initialized, it is a foreign volume.

/OVERRIDE:IDENTIFICATION

Allows privileged users to mount a volume without using the volume label. IDENTIFICATION is the default argument for /OVERRIDE. All other arguments to OVERRIDE are for tapes and are described in Volume 2, Chapter 13.

/OWNER:[g,m]

Specifies the owner of the volume. The brackets are required syntax. This qualifier overrides the OWNER value established when the volume was initialized. The owner value is used with file and volume protection. See Section 5.6.

/PROTECTION:(code)

Specifies the volume protection for Files–11 disks. This protection overrides the volume protection established when the volume was initialized. The /PROTECTION qualifier combines with the /OWNER qualifier to control access to the volume. See Section 5.6 for more information on volume protection.

/PUBLIC
/SYSTEM

These two qualifiers are synonyms. The /SYSTEM qualifier is included for VAX/VMS compatibility.

Specifies that the mounted volume be available to all users who are allowed access under the volume protection and file protection codes established for the volume when mounted. See the discussion on file protection and volume protection in Section 5.6. These qualifiers are privileged.

If you state this qualifier explicitly when mounting a volume on an allocated (private) device, the device is automatically deallocated and set public.

If you do not state this qualifier explicitly, and the device is already set public, the mount defaults to /PUBLIC.

See the discussion of public, private, and shareable volumes in Volume 2, Chapter 13.

/[NO]SHAREABLE

Specifies whether the volume is to be mounted shareable.

A volume mounted /SHAREABLE can be mounted multiple times by the same or different users. Each user's access is determined by the volume-protection and file-protection codes established for the volume when mounted. See the discussion of file protection and volume protection in Section 5.6.

If you mount the volume /SHAREABLE and the device is allocated or set public, the device is automatically deallocated or set nonpublic.

A volume mounted /NOSHAREABLE is dedicated for your private use. No other user can access the volume. For Files–11 volumes mounted /NOSHAREABLE, your privileges are SYSTEM privileges. See the discussion of file protection and volume protection in Section 5.6.

Mounting a volume nonshareable is the equivalent of allocating the device and then mounting it. See Volume 2, Chapter 13 for a description of the ALLOCATE command. If you specify /NOSHAREABLE explicitly when mounting a volume on a device, the device is allocated and set nonpublic.

If the device is already allocated, the default is /NOSHAREABLE.

If the device is not allocated or set public, the default is /SHAREABLE.

If the device is set public, the default is /PUBLIC (synonym: /SYSTEM).

See the discussion of public, private, and shareable volumes in Section 6.1.2 and Volume 2, Chapter 13.

/[NO]WAIT

Specifies whether you require operator assistance in performing the mount. This qualifier is used mainly for batch jobs, indirect command files, or terminals distant from the machine room. The default is /WAIT for mounts in batch jobs and indirect command files, and /NOWAIT for interactive mounts.

If the mount is included in a batch job or indirect command file, or if you specify /WAIT in an interactive mount, a message concerning the mount is sent to the operator's console and the mount is not completed until the operator physically places the disk on the drive and spins it up. If you specify /NOWAIT in a batch job or indirect command file, or if the mount is interactive, no message appears. The medium must have been placed on the device and readied for access (that is, spun up and on line). (See the examples.)

For more information on mounts from batch jobs, see the discussion of MOUNT in Chapter 8.

/[NO]WRITE

Specifies whether the volume is to be write-protected. The default is /WRITE. If /WRITE is specified or implied, the volume can be written to as permitted by the volume protection and file protection codes established for the volume when mounted. See the discussion of volume protection and file protection in Section 5.6. If a volume is mounted with the /NOWRITE qualifier, no one may write to the volume.

Examples

```
$ MOUNT RET
Device? DU1: RET
Volume ID? HOTROD RET
```

This example mounts the volume labeled HOTROD on device DU1:. If you are doing nothing more than mounting a volume to read from it or write to it, you will probably not need a more complicated MOUNT command than this.

The command sets a series of default values that do not concern the everyday user. All default values for the mount come from the Volume Home Block. This block is written when the volume is created by INITIALIZE or by some other system task (such as a backup utility) that creates a file structure on the volume. Most of the qualifiers to MOUNT are used to override values in the Volume Home Block. You can also alter the Volume Home Block with the INITIALIZE/UPDATE command. These operations are described in Volume 2, Chapter 13.

```
$ MOUNT/WAIT DU1: WHIZZER [RET]
```

In this example, the user wants an operator to place the medium containing the proper volume in place on DU1:. The command may appear in a batch job, indirect command file, or interactive mount, perhaps from a user on a remote termimal. The following message appears on CO: (TT0:), the operator's console:

```
Please mount volume WHIZZER on DU1:
Type
UNS (DCL START) when ready
or
RES (DCL CONTINUE) to reject request.
```

The operator performs the requested operation and the mount takes place, if the device was available and the operator typed START. (There is no MCR on Micro/RSX systems.) If the device was not available and the operator typed CONTINUE, the mount fails. The batch processor attempts to continue the job. The same is true if the command appears in an indirect command file.

Notes

You should not confuse mounting with physically placing the disk or other magnetic medium on a drive. Naturally, you must place the medium on a drive and spin it up (for a disk) or load it (for a tape) before you can do anything else with it. However, mounting enables the system software to access the medium, either to read or write data, or to establish a file structure for the volume.

There are four states of device ownership:

Private	The device has been made private through the ALLOCATE command or the /NOSHAREABLE qualifier to MOUNT. Only you can access the volume.
Shareable	The device has been mounted /SHAREABLE. Any user can mount the volume.
Public	The device has been made public through the SET DEVICE /PUBLIC command or the /PUBLIC (or /SYSTEM) qualifier to MOUNT. Only a privileged user can set a device public.

Unowned No one has mounted a volume on the device, nor is it allocated or set public.

See the discussion of public, private, unowned, and shareable devices and mounted volumes in Volume 2, Chapter 13.

You cannot put comments in a MOUNT command line.

DISMOUNT counteracts MOUNT. See Section 6.3 for a description of DISMOUNT with examples.

See Volume 2, Chapter 13 for more information on how the operating system controls devices and volumes. See also the description of the INITIALIZE command in Section 6.4.

Error Messages

MOU—ACP not in system

Explanation: The task specified as the ACP, or the default ACP, is not installed in the system.

User Action: See your system manager. Chapter 13 in Volume 2 has more information about ACPs.

MOU—Device not in system -ddnn:

Explanation: The command specified a device not present in the system.

User Action: Retype the command after checking the device list (SHOW DEVICES) or see your system manager.

MOU—Device specified twice

Explanation: The command specified the same device twice.

User Action: Retype command after checking for proper syntax.

MOU—Driver not loaded

Explanation: The command named a device whose driver is not loaded.

User Action: See your system manager.

MOU—Failed to attach device -ddnn:

Explanation: The command named a device that was attached by another task and could not be mounted.

User Action: Check the device list (SHOW DEVICES) to find out if the device is in use. See your system manager.

MOU—Home block I/O error

Explanation: An I/O error was detected in trying to read the Volume Home Block. This message often indicates that the device is not ready, or that the disk has not been initialized.

User Action: Try again. Switch the diskette to the other drive and try again. If the error recurs, you may have the wrong disk, or one that has not been initialized.

MOU—Illegal keyword combination

Explanation: The command specified conflicting qualifiers.

User Action: Retype command after checking for proper syntax.

MOU—Index file I/O error

Explanation: MOUNT could not read either the index file header or the storage allocation file.

User Action: See your system manager. See Volume 2, Chapter 13 for more information.

MOU—No such device available

Explanation: The command named a device not present in the system.

User Action: Retype the command after checking the device list (SHOW DEVICES) or see your system manager.

MOU—Not file-structured device

Explanation: The command named a device that is not supported as a Files–11 device.

User Action: Retype the command after checking for proper device name.

MOU—Parameter conflicts with mounted volume

Explanation: An attempt was made to mount a previously mounted volume using qualifiers that conflict with those specified when the volume was originally mounted.

User Action: Check with SHOW DEVICES and confirm qualifiers with other user. See your system manager. This may happen with a MOUNT/SHAREABLE command.

MOU—Storage bit map file I/O error

Explanation: An I/O error was encountered while reading the storage allocation file.

User Action: Check to be sure that you have the correct magnetic medium, or the volume has not been initialized. See your system manager.

MOU—Unit allocated to or in use by another user.

Explanation: The command specified a device that is already in use.

User Action: Check on the status of the device with SHOW DEVICES and take the appropriate action.

MOU—Unsupported file header format

Explanation: The Volume Home Block does not conform to Files–11 format. The volume may be corrupted.

User Action: The volume may already be mounted with the /FOREIGN qualifier, or some other special condition is in effect. See your system manager.

MOU—Volume already mounted -ddnn:

Explanation: An attempt was made to mount a volume on a device that already had a mounted volume.

User Action: See your system manager.

MOU—Wrong volume label

Explanation: The volume label on the label and the volume label in the command do not match.

User Action: Retype the command after checking for proper volume label. Privileged users can use the /OVERRIDE:IDENTIFICATION qualifier.

6.3 DISMOUNT

DISMOUNT counteracts MOUNT. DISMOUNT marks the volume logically off line and disconnected from the file system. Marking a volume for dismount prevents programs from opening new files on the volume. After all open files on the volume have been closed, the volume is dismounted.

Format

DISMOUNT[/qualifier[s]]
Device? ddnn: [volumelabel]

DISMOUNT[/qualifier[s]] ddnn: [volumelabel]

Command Qualifiers

/ALL
/PUBLIC
/SAVE
/SYSTEM (synonym for /PUBLIC)
/TERMINAL:ttnn:

Parameters

ddnn:

Specifies the device on which the volume is mounted.

volumelabel

Specifies the volume label or File Set ID for magnetic tape. This parameter is optional, but if it is specified, the label or File Set ID is checked against the mounted volume. There is no prompt for this parameter.

Command Qualifiers

/ALL

Specifies that all volumes mounted from the terminal at which the command is issued be dismounted. A message informs you of each dismount as it takes place. No device name or volume label is accepted with this qualifier. See the example.

/PUBLIC
/SYSTEM

Causes all users to be dismounted from a volume. This is a privileged qualifier. A DISMOUNT/PUBLIC frees the device, no matter who has mounted the volume on it. This is the only way to dismount a public volume. The /SYSTEM qualifier is a synonym for /PUBLIC, included for compatibility with VAX/VMS.

/SAVE

Specifies that the disk is to remain spinning in the drive and can be accessed for read or write operations by privileged tasks. This is a privileged qualifier.

The most common use of /SAVE is when saving or backing up the system disk. If you simply want to leave the disk spinning, use the /NOUNLOAD qualifier. Remember, these values can also be set with the /DEFAULT qualifier to MOUNT. See Section 6.2.

/TERMINAL:ttnn:

Allows a privileged user to dismount a volume mounted from another terminal.

Examples

```
$ DISMOUNT [RET]
Device? DU2: [RET]
DIS -- TT7:  Dismounted from DU2: *** Final Dismount Initiated ***
```

This example dismounts a volume on device DU2:. No other user had the volume mounted when the DISMOUNT command was issued.

```
$ DISMOUNT DU1: HOTROD [RET]
DIS -- TT3: Dismounted from DU1:
```

This example dismounts a volume on device DU1:. The user specified the label HOTROD to be sure the correct volume was mounted on the device. The volume was mounted shareable and because another user also has it mounted, the informational message does not include the notification of final dismount.

```
$ LOGOUT [RET]
   DMO -- TT1:  Dismounted from DU: *** Final dismount ***
   Have a good morning
   22-APR-85 TT1: Logged off
```

This example dismounts a volume as part of the LOGOUT procedure. The user had the volume mounted on the device and had not dismounted it before logging out. LOGOUT dismounted the volume. If the device had been made private through the /NOSHAREABLE qualifier to MOUNT, or the ALLOCATE command, LOGOUT also deallocated it. This is the equivalent of DISMOUNT/ALL.

The informational message is headed by DMO rather than DIS for historical reasons.

```
$ DISMOUNT/ALL [RET]
DIS -- TT1:   Dismounted from DU1:
DIS -- TT1:   Dismounted from DU2:  *** Final dismount ***
```

This example dismounts all volumes mounted from the terminal at which the DISMOUNT was issued.

```
$ DISMOUNT/PUBLIC DU1: [RET]
DIS -- TT2:   Dismounted from DU1:
DIS -- TT4:   Dismounted from DU1:
DIS -- TT5:   Dismounted from DU1:  *** Final dismount ***
```

In this example, a privileged user dismounts all users who mounted the volume on DU1:.

Notes

DISMOUNT counteracts MOUNT.

LOGOUT issues a DISMOUNT/ALL automatically. DISMOUNT/ALL dismounts all volumes mounted from the terminal at which the command is entered.

You cannot include comments in a DISMOUNT command line.

The messages to your terminal indicate only that the volume is marked for dismount. The actual completion of the dismount is noted on the operator's console.

CAUTION

When the dismount operation is complete, the ACP prints the following message on the operator's console (CO:):

```
*** ddnn Dismount complete
```

This message does not appear until all files that are open on the volume are closed. Do not remove the medium from the drive until this message appears on CO: (TT0:). If you remove the medium before the message appears, the present volume may be corrupted, and the next volume mounted on that device will be corrupted.

Error Messages

DIS—Checkpoint file still active

Explanation: The command attempted to dismount a volume that contained an active checkpoint file. The volume cannot be dismounted until the checkpoint file has been closed.

User Action: Wait, or, if privileged, you can issue a SET DEVICE:ddnn: /NOCHECKPOINT and reissue the DISMOUNT after you receive the system message indicating that the checkpoint file is no longer active.

DIS—Volume not mounted

Explanation: The command specified a device that was not mounted.

User Action: Retype command after checking SHOW DEVICES for mounted devices.

DIS-Volume not mounted by TI:

Explanation: The command attempted to dismount a volume mounted from another terminal.

User Action: Dismount the volume from the proper terminal, or have a privileged user dismount the volume.

DIS—Wrong volume label

Explanation: The command included an incorrect volume label.

User Action: Reissue the command without specifying a volume label or check the volume label to be sure you are dismounting the right volume.

6.4 INITIALIZE

INITIALIZE produces a volume in Files–11 format. Diskettes must be initialized before you can do anything with them.

When you initialize a diskette, you erase all existing files. Initialization also writes a dummy bootstrap and a Volume Home Block, and builds the directory structure.

Several qualifiers to INITIALIZE and a great deal more information, including tape initialization, is included in Volume 2, Chapter 13. If you do not have the Advanced Programmer's Kit you have no need to initialize tapes.

When initializing diskettes, you generally do not need any qualifiers.

Format

```
INITIALIZE[/qualifier[s]]
Device?  ddnn:
Label?  volumelabel

INITIALIZE[/qualifier[s]] ddnn: volumelabel
```

Command Qualifiers

```
/BAD_BLOCKS:arg
                AUTOMATIC
                MANUAL
                NOAUTOMATIC
/FILE_PROTECTION:(code)
/OWNER:[g,m]
/PROTECTION:(code)
```

Parameters

ddnn:

Specifies the name of the device on which the magnetic medium to contain the volume has been placed. In most cases, this is DU1: or DU2:.

volumelabel

Specifies the label the volume is to be initialized with. The label names the volume and must be specified by nonprivileged users when they mount the volume. The volume label is, in effect, a password controlling access to the volume. The volume label can be as many as 12 characters long.

Command Qualifiers

Defaults to INITIALIZE enable you to initialize a volume in a standard fashion, but the qualifiers allow much more flexibility.

/BAD_BLOCKS:arg

AUTOMATIC
MANUAL
NOAUTOMATIC

AUTOMATIC reads the bad-block descriptor file created by the ANALYZE/MEDIA command and automatically determines the bad-block information for the volume. This is the default. See the *Micro/RSX System Manager's Guide* for more information.

MANUAL specifies that bad-block information for the volume is to be entered manually.

NOAUTOMATIC specifies that bad-block information is to be ignored.

/FILE_PROTECTION:(code)

Specifies the default protection for all files on the volume being initialized. See Section 5.6 for more information on file protection.

The default file protection code is as follows:

```
(SYSTEM:RWED,OWNER:RWED,GROUP:RWED,WORLD:R)
```

/OWNER:[g,m]

Specifies the owner of the volume. Group and member numbers range from 1 through 377. The default volume owner is [1,1]. The /OWNER value is used for checking volume protection. See Section 5.6 for more information on volume and file protection.

/PROTECTION:(code)

Specifies the default protection for new files created on the volume. See the discussion of volume protection in Section 5.6. The code for volume protection is similar to that for file protection, except that the "E" for EXTEND protection is replaced by a "C" for CREATE protection.

The default volume protection is as follows:

```
(SYSTEM:RWCD,OWNER:RWCD,GROUP:RWCD,WORLD:RWCD)
```

Example

```
$ INITIALIZE [RET]
Device? DU1: [RET]
Volume ID? HONOLULU [RET]
```

This example initializes a volume with the volume label HONOLULU on device DU1:. In most cases, this is all you will have to do.

Notes

Remember, when you initialize a disk volume, you are erasing all information that was previously on the disk or diskette. Initialize with care.

Before you can initialize a disk volume, you must issue a MOUNT command with the /FOREIGN and /NOSHAREABLE qualifiers. The /FOREIGN qualifier is required because the volume is considered not in Files–11 format before initialization. The /NOSHAREABLE qualifier is required because you cannot initialize a disk volume if any other user has access to it. See the explanation in the MOUNT command description in Section 6.2.

INITIALIZE creates a Volume Home Block and five files used by the system. Many of the qualifiers to INITIALIZE put values in the Volume Home Block. Most of these values can be overridden through similar qualifiers to MOUNT. You can change the values in the Volume Home Block with the INITIALIZE/UPDATE command. See Volume 2, Chapter 13 for more information on these operations.

Error Messages

INI—Allocation for system file exceeds volume limit

Explanation: The system was unable to allocate a system file from the specified block because of intermediate bad blocks or end-of-volume.

User Action: Reenter command with different argument for the /INDEX: qualifier. See Volume 2, Chapter 13.

INI—Bad block file corrupt - data ignored

Explanation: Although the /BAD_BLOCKS qualifier was selected, or defaulted to, the bad block data on the disk was not in the correct format and was therefore ignored.

User Action: Process the medium with the ANALYZE/MEDIA command and initialize again. See Volume 2, Chapter 13 for more information.

INI—Block(s) exceed volume limit

Explanation: The specified block or blocks exceeded the physical size of the volume.

User Action: Retype command after checking for proper qualifier values. See Volume 2, Chapter 13.

INI—Boot block write error

Explanation: An error was detected in writing out the volume boot block.

User Action: Reenter the command. If it still does not work, tell your system manager.

INI—Checkpoint file header I/O error

Explanation: An error was detected in writing out the checkpoint file header.

User Action: Reenter the command. If it still does not work, tell your system manager. See Volume 2, Chapter 13 for information on the checkpoint file.

INI—Command I/O error

Explanation: INITIALIZE encountered an I/O error while reading the command line.

User Action: Retype command line. If this fails, tell your system manager.

INI—Data error

Explanation: The command specified a bad block number or contiguous region that was too large.

User Action: Reenter command after checking proper syntax. See Volume 2, Chapter 13.

INI—Device allocated to other user -ddnn:

Explanation: Command specified a private device not allocated to your terminal. You can make a device private with the ALLOCATE command or MOUNT/NOSHAREABLE.

User Action: Make sure you entered the correct device name. Find out who has the device allocated and why before taking further action.

INI—Device not in system

Explanation: Command specified a device not in the current system.

User Action: Retype command after checking SHOW DEVICES for proper device name.

INI—Device not ready -ddnn:

Explanation: The device was not up to speed (spun up).

User Action: Wait and try again.

INI—Driver not loaded

Explanation: Command specified a device for which the driver is not loaded.

User Action: Retype command after checking SHOW DEVICES for proper device name.

INI—Failed to attach device -ddnn:

Explanation: INITIALIZE failed to attach the specified device.

User Action: Wait and try again. Check SHOW DEVICES. See your system manager.

INI—Failed to read bad block file

Explanation: Although the /BAD_BLOCKS:AUTOMATIC qualifier was specified, or defaulted to, no bad-block data was found.

User Action: Process the medium through the ANALYZE/MEDIA command and try again. See Volume 2, Chapter 13 for more information.

INI—Not file structured

Explanation: The system does not support the device named as a Files–11 device.

User Action: Retype command after checking proper syntax. See Volume 2, Chapter 13.

INI—Public device -ddnn:

Explanation: Command attempted to initialize a volume on a public device.

User Action: None. Nonprivileged users can only initialize volumes on private devices.

INI—Storage bit map file error

Explanation: The system failed to read the header of the file [000000]BITMAP.SYS.

User Action: See your system manager. A privileged user may be able to fix the problem using the INITIALIZE/UPDATE command, described in Volume 2, Chapter 13.

INI—Undefined density selection

Explanation: Command specified an illegal density argument for the device named.

User Action: Retype command after checking for proper density argument. See Volume 2, Chapter 13.

INI—Volume mounted Files–11

Explanation: Command attempted to initialize a volume mounted Files–11.

User Action: Check to make sure you have the right volume.

INI—Volume mounted foreign with ACP

Explanation: Command attempted to initialize a volume mounted /FOREIGN but with the /ACP qualifier.

User Action: Remount the volume without the /ACP qualifier.

INI—Volume name too long—volumelabel

Explanation: Command included a volume label that was too long.

User Action: Retype command but limit diskette volume label to 12 characters.

6.5 SET and SHOW

You can display assignments and set and display certain device characteristics. See Volume 2, Chapter 13 for information on device assignment and the SET DEVICE command.

6.5.1 SHOW DEVICES

SHOW DEVICES displays information about the devices included in the system.

Format

SHOW DEVICES[:dd[nn:][/attribute]]

Device Attributes

/[NO]PUBLIC	
/WIDTH:ddnn:	
/[NO]SYSTEM	synonym for /PUBLIC

Argument

dd[nn:]

If you specify just the dd portion of the device name, as in the following:

```
$ SHOW DEVICE:DU [RET]
```

the command displays information about all devices of that type on the system. If you specify the full device name, as in the following:

```
$ SHOW DEVICE:DU1: [RET]
```

the display shows you information about DU1: only.

Device Attributes

If you do not include an attribute or a device name, SHOW DEVICES displays a list of all the devices on the system, including terminals and pseudo devices.

/[NO]PUBLIC
/[NO]SYSTEM

Displays a list of all devices set (or not set) PUBLIC. The /[NO]SYSTEM qualifier is a synonym included for VAX/VMS compatibility.

/WIDTH

Displays the size of the I/O buffer (line length) for a particular device, including a terminal.

Display Information

The display from SHOW DEVICES can include a number of messages. See the examples.

ddnn:

A device name in the first column indicates the device or pseudo device for which information is being displayed. A device name in the second column indicates a device to which the corresponding device in the first column has been redirected. See Volume 2, Chapter 13 for more information.

TTnn:

A terminal name in the second column, followed by the word PRIVATE, indicates that the device named in the first column has been allocated by the user logged in on the terminal in the second column.

MOUNTED

Indicates that the device is mounted. For privileged users, the message also includes the volume label.

BUF=

Indicates the line length (I/O buffer size).

PUBLIC

Indicates that the device has been set public.

If your command was SHOW DEVICES/PUBLIC, or /NOPUBLIC, the display is PUB=ddnn: or NOPUB=ddnn:

TYPE=

Indicates the device type by model name, for example, RD51, RX50.

MARKED FOR DISMOUNT

Indicates that a mounted device has been marked for dismount, but that the dismount has not been completed, meaning that files are still open on the volume. The volume cannot be remounted while it is marked for dismount.

OFFLINE

Indicates that the system tables contain entries for the device, but that the device is not included in the current configuration.

[directory] LOGGED ON

Indicates that the terminal is logged on and that your current directory is [directory].

LOADED

Indicates that a loadable device driver is currently loaded.

UNLOADED

Indicates that a loadable device driver is currently not loaded.

SPOOLED

Indicates that a device is under the control of the Queue Manager.

WCHK=
NOWCHK=

Indicates a device with write-checking enabled or not enabled.

Examples

```
$ SHOW DEVICES [RET]
TT0:      Loaded
TT1:      DU0: Spooled Loaded
TT2:      [BODDLY] - Logged in Loaded
TT3:      Loaded
TT4:      [JEROME] - Logged in Loaded
TT5:      Loaded
VT0:      Loaded
VT1:      Loaded
RD0:      Loaded
DU0:      Public Mounted Loaded Type=RD51
DU1:      TT2: - Private Mounted Loaded Type=RX50
DU3:      Offline Loaded Type=unknown
NL0:      Loaded
```

```
TI0:
CO0:      TT0:
CL0:      TT0:
SP0:      DU0:
LB0:      DU0:
SY0:      DU0:
```

This example shows the display from SHOW DEVICES. All devices and pseudo devices are included. For terminals, the name in brackets is the default directory. The only information shown for pseudo devices is the name and the device to which they are redirected. Privileged users also see volume labels.

```
$ SHOW DEVICES DU: [RET]
DU0:      Public Mounted Loaded Type=RD51
DU1:      TT2: - Private Mounted Loaded Type=RX50
DU3:      Offline Loaded Type=unknown
$
```

This example displays information about all devices of the type DU:. The mnemonic (DUnn:) identifies the device controller. Both the fixed disk and the diskettes use the same device controller, although the media are very different.

```
$ SHOW DEVICES/PUBLIC [RET]
PUB=DU0:
PUB=DU1:
PUB=LP0:
```

This example displays information about all public devices. Public devices are accessible to all users. They may be mounted by any user to assure continued access to the volume mounted on the device.

```
$ SHOW DEVICE:LP0:/WIDTH [RET]
BUF=LP0:00132.
$ SHOW DEVICE:TT1:/WIDTH [RET]
BUF=TT11:00080.
```

This example displays the line lengths of a line printer and a terminal. The line length is determined by the size of the I/O buffer, which is why the word BUF appears.

Chapter 7
Running Tasks

Tasks on Micro/RSX are programs that can be run on the system. Tasks that are used all the time can be installed, which makes them immediately available when needed. Tasks that are used only occasionally can be kept in a file in a directory and run from there. In general, the system manager decides which tasks should be installed and which should be run from task image files.

This chapter includes brief descriptions of the following commands: RUN, ABORT, INSTALL, and REMOVE, as well as commands for displaying information about running and installed tasks.

There is a great deal more information about task execution in Volume 2, Chapter 15, More About Running Tasks.

For more information about what makes a task, see Volume 2, Chapter 14, LINK and LIBRARY Commands, and also *Programming on Micro/RSX*.

7.1 Two Kinds of RUN Command

Just as there are two kinds of tasks you can run on Micro/RSX systems, there are two ways of using the RUN command:

1. Running tasks in *task image files*
2. Running *installed tasks*

Tasks from directories are tasks included in task image files. To run these tasks, the system must temporarily install them. To run such tasks, simply type RUN followed by the file specification. See Section 7.1.1.

Installed tasks are always ready to run. You can see a list of installed tasks with the SHOW TASKS/INSTALLED command. You cannot run all installed tasks. Many are used by the system. For those you can run, simply type RUN and give the task name. Most large applications run as installed tasks. See Section 7.1.2.

7.1.1 RUN Command for Task Image Files

When used to run an uninstalled task, RUN is actually a combination command, encompassing INSTALL, RUN, and REMOVE. This command is the only way a nonprivileged user can install or remove a task.

Format

```
RUN[/qualifier[s]]
Task?  [$]filespec

RUN[/qualifier[s]] [$]filespec
```

Command Qualifiers

```
/TASK_NAME:taskname
/TIME_LIMIT:n[u]
/UIC:[g,m]
```

Parameter

[$]filespec

Specifies a task image file on a mounted Files–11 volume. This task is installed, run, and then removed when it has finished executing.

The default file type is .TSK, but if you include the .TSK in the file specification, then you can be sure that you are running the file from the default directory, and not an installed task with the same name. See the examples.

The dollar sign ($) directs the system to search first for the file in the system directory and then in the library directory. In this case, the dollar sign specifies a device and directory and thus counts as an element of a file specification.

Command Qualifiers

These qualifiers establish how the uninstalled task is to be installed.

/TASK_NAME:taskname

Specifies the name under which the task is to be run. The default is to run the task under a name derived from the name of the terminal from which the RUN command was issued, as discussed in Volume 2, Chapter 15.

Use this qualifier when you want to run two tasks simultaneously using the RUN command. Otherwise, the second RUN command causes the following error message to appear on your terminal:

```
Run -- Task name already in use
```

See the examples.

Task names are restricted to six Radix–50 characters. The Radix–50 character set consists of the 26 uppercase letters, the 10 numerals, and the period (.) and dollar sign ($).

/TIME_LIMIT:n[u]

Allows you to limit the amount of CPU time the task can run. The default is /TIME_LIMIT:3M, which is quite a lot. The "M" stands for minutes. If you give simply a number as an argument, the time unit defaults to minutes, but you can also specify a time limit in seconds by using a number and an "S", as in /TIME_LIMIT:30S.

/UIC:[g,m]

Specifies the default UIC for the task. This is a privileged qualifier. This UIC determines in what protection class the task belongs and thus determines file access.

The square brackets are required syntax.

Examples

```
$ RUN ROBOT.TSK [RET]
```

This example installs, runs, and removes upon completion the task contained in the task image file ROBOT.TSK from the default directory on the default device.

```
$ RUN ROBOT [RET]
```

This example works differently depending on whether or not there is an installed task named ROBOT. If there is an installed task named ROBOT, this command runs it. If there is not such as task installed, this command works the same as the previous example.

Obviously, it is best not to have both installed and uninstalled tasks using the same name, but it can happen. If you include the file type, or any other field of a file specification, including the dollar sign ($), you can avoid running an installed task. See the notes for more information.

Notes

The default on Micro/RSX systems is to run only one task per terminal at one time, but in fact Micro/RSX allows you to run many tasks from one terminal at the same time. You can change this characteristic with the SET TERMINAL/NOSERIAL command, described in Volume 2, Chapter 11, More About Using Your Terminal.

For information about placing files in the system directory or library directory so that users can run them with the dollar-sign ($) convention, see Volume 2, Chapter 15, More About Running Tasks.

This section describes RUN as it works with any portion of a file specification included. In general, you do not have to include the file type, as .TSK is the default. In one unusual circumstance, however, you must explicitly enter a file type. If there is a task already installed in the system that has the same task name as the file name of the task image file containing the task you want to run, the RUN command will run the installed task rather than the one you want to run. In such an instance, you must include the file type to force the RUN command to access the task image file.

Some error messages resulting from RUN are labeled INS rather than RUN because this form of RUN installs the task automatically and the error occurs during the installation.

7.1.2 RUN Command for Installed Tasks

RUN also initiates the execution of installed tasks. Privileged users can use RUN to initiate the execution of installed tasks on a schedule by creating entries in the system clock queue. See Volume 2, Chapter 15.

Format

```
RUN[/qualifier[s]]
Task?   taskname

RUN[/qualifier[s]] taskname
```

Command Qualifier

/UIC:[g,m]

Parameter

taskname

Specifies the name of the installed task to be run. The names of installed tasks are one through six Radix–50 characters. If the System Task Directory (STD) contains no entry under the task name you supply, the system searches the default directory on the default device for a file named taskname.TSK. If RUN finds such a file, it installs, runs, and removes it.

Command Qualifier

/UIC:[g,m]

Specifies the default UIC for the task. This is a privileged qualifier. The task's UIC determines what file protection class it belongs in, and thus directly influences file access.

The square brackets are required syntax.

Notes

The dollar sign prompt ($) returns immediately after you issue a RUN command for an installed task. This does not indicate that the task has completed execution.

You can display entries in the clock queue with SHOW CLOCK_QUEUE. See Volume 2, Chapter 15.

Error Message

RUN—Invalid time parameter

Explanation: Command specified an invalid time argument.

User Action: Check for proper syntax and reenter command.

7.2 ABORT

ABORT forces an orderly end to a running task or to the action of a specific command. On most Micro/RSX systems, you can simply use CTRL/C to abort any task running from your terminal.

Tasks can also be aborted by other tasks or by error-handling routines. If this happens, some of the messages discussed below can appear on your terminal without your having issued an ABORT command.

Nonprivileged users can abort any task running on their own terminal. Privileged users can abort any task.

Format

ABORT[/COMMAND][/qualifier[s]] commandname

ABORT/TASK[/qualifier[s]] [taskname]

Command Qualifiers
/COMMAND
/TASK
/TERMINAL:ttnn:

Parameters

The parameters differ depending on whether you are aborting a command or a task.

The default is to abort a command. See the examples.

commandname

Specifies the command whose effect you want to cancel. This parameter can be used only when the /TASK qualifier is not present. You must specify at least the first three characters of the command verb.

taskname

Specifies the name of the task you want to abort. This parameter requires the presence of the /TASK qualifier. If you use the /TASK qualifier and do not specify a task name, you will get an error message, "Illegal task name."

Command Qualifiers

/COMMAND

Specifies that you want to abort a command. This is the default qualifier and need not be included.

/TASK

Specifies that you want to abort a task by name.

/TERMINAL:ttnn:

Specifies that a task from some terminal other than your own be aborted. This is a privileged qualifier.

Examples

```
$ ABO RUN RET
11:11:11 Task "TT10" terminated
        Aborted via directive OR CLI
```

This example aborts a task initiated with the RUN command running on your terminal. See the next example.

```
$ RUN TREK RET
ORDERS: STAR DATE = 2000
You must destroy the Klingon invasion forces of 29 battle
cruisers. You have 40 solar years to complete your mission.
Ready? CTRL/C
```

In this example, the user installs the interactive task TREK by means of a RUN command. Then, instead of providing the task with requested input, the user issues a CTRL/C and aborts the task.

```
$ ABORT/TASK MACT3 RET
```

This example aborts the task named MACT3. Nonprivileged users can abort tasks running on their own terminals. Privileged users can abort tasks running on any terminal by name.

```
$ DIRECTORY *.RNO RET

Directory DUO:[MONGO]
18-MAY-85 16:28

ALTCLI.RNO;3          213.     27-APR-85 16:52
ANNI.RNO;6            3.       30-JAN-85 09:30
APRCOM.RNO;4          9.       15-APR-85 10:35
BEST.RNO;5            40.      28-APR-85 15:50
CATCH.RNO;4           7.       22-MAR-85 09:39
CHAP11SMG.RNO;13      49.      13-MAR-85 16:36
CHAP12SMG.RNO;51      223.     08-APR-85 13:15
CHARSET.RNO;5         6.       15-FEB-85 11:57CTRL/C
$
```

In this example, the user issues a DIRECTORY command and sees that the file she was checking for is present. Rather than wait for the directory listing to complete, the user types CTRL/C and the directory listing halts immediately.

```
$ ABORT/TERMINAL:TT3: ACRO RET
```

This example, issued by a privileged user, aborts the ACRO task running on another terminal. Notification of the abort appears on TT3: but not on the terminal from which the ABORT command was issued.

Notes

A is the short form of ABORT.

There can be a number of messages issued with the abort. These messages are explained in Volume 2, Chapter 16.

LOGOUT also aborts nonprivileged tasks running from your terminal.

Tasks can also be aborted by Executive directives issued by other tasks, in particular, by error-handling routines.

7.3 INSTALL

INSTALL includes a specific task in the System Task Directory, thus making it known to the system. This is a privileged command. A number of qualifiers and other information about INSTALL are discussed in Volume 2, Chapter 15.

This is a privileged command. Nonprivileged users install tasks temporarily through RUN, which includes most of the same qualifiers as INSTALL. See Volume 2, Chapter 15.

Format

INSTALL[/qualifier[s]]
File(s)? [$]filespec

INSTALL[/qualifier[s]] [$]filespec

Command Qualifiers

/[NO]CHECKPOINT
/PRIORITY:n
/TASK_NAME:taskname
/UIC:[g,m]

Parameter

[$]filespec

Specifies the name of the task image file containing the task that you want to install. The default file type is .TSK. The dollar sign ($), if present, directs the system to search for the file first in the system directory and then in the library directory. If you do not include the /TASK_NAME qualifier, the task will be installed under a name based on the first six characters of the file name unless another name was assigned through the TASK= option of the Task Builder. See the examples for more information.

Command Qualifiers

/[NO]CHECKPOINT

Specifies whether or not the task is checkpointable. The default is set at link time. This qualifier overrides the link-time checkpointability specification.

/PRIORITY:n

Specifies the priority at which the task is to run. The default is set at link time. This qualifier overrides the default set at link time. The argument n can range from 0 through 250.

/TASK_NAME:taskname

Specifies the name by which the task is to be referenced. The default is set at link time. This qualifier overrides the link-time specification. See the examples for more information.

/UIC:[g,m]

Specifies the default UIC for the task. This task UIC determines in what protection class the task belongs and thus directly influences file access. The brackets are required syntax.

Examples

```
$ INSTALL FATE [RET]
```

This example looks for a task image file named FATE.TSK on the default device in the default directory and installs it as FATE in the STD, assuming no other name was specified at link time.

```
$ INSTALL $LATE [RET]
```

This example looks in the system directory (by convention, [1,54] and the library directory (by convention, [3,54]) for a task image file named LATE.TSK and installs it as LATE in the STD, assuming no other name was specified at link time.

```
$ INSTALL BUCKING [RET]
```

This example looks for a task image file named BUCKING.TSK on the default device in the default directory, and installs it as BUCKIN in the STD, assuming no other name was specified at link time.

```
$ INSTALL/TASK_NAME:DOOR GATE [RET]
```

This example looks on the default device in the default directory for a task image file named GATE.TSK and installs it as DOOR in the STD, overriding any task name specified at link time.

Notes

REMOVE counteracts INSTALL. See Section 7.4.

You can display the attributes of installed tasks with SHOW TASKS /INSTALLED. See Section 7.5.1.2.

INSTALL can also install commons. See Volume 2, Chapter 15.

7.4 REMOVE

REMOVE takes a task name out of the System Task Directory. The task is no longer installed.

REMOVE/REGION takes the name of a region out of the Common Block Directory and the partition list. This is a privileged command.

Format

REMOVE[/qualifier]
Task? taskname

REMOVE[/qualifier] taskname

Command Qualifier

/REGION

Parameter

taskname

Specifies the name of the task you want to remove.

If you want to remove a region, specify the /REGION qualifier and the name of the region.

Command Qualifier

/REGION

Specifies that you want to remove a region from the Common Block Directory.

Examples

```
$ REMOVE WONTON [RET]
```

This example removes the task named WONTON from the System Task Directory. The task is no longer installed.

```
$ REMOVE/REGION DONJON [RET]
```

This example removes the common region named DONJON from the Common Block Directory.

Notes

To remove an active task, you must first abort it.

If a task is fixed, REMOVE first unfixes it and then removes it. See Volume 2, Chapter 15 for information on fixed tasks and the FIX command.

REMOVE cancels all time-based requests for the task to run.

REMOVE deallocates all receive-by-reference and receive-data packets for the task and detaches all attached regions. See the *RSX-11M/M-PLUS and Micro/RSX Executive Reference Manual* in the Advanced Programmer's Kit for more information.

7.5 SET and SHOW

All users can display information about active and installed tasks.

Privileged users can change the priority of active tasks.

See Volume 2, Chapter 15 for other SET and SHOW commands having to do with running tasks.

7.5.1 SHOW TASKS

SHOW TASKS displays information about active and installed tasks on the system.

7.5.1.1 SHOW TASKS/ACTIVE

SHOW TASKS/ACTIVE displays information about active tasks in brief and full formats.

Brief Format

SHOW TASKS/ACTIVE[:ttnn:][/qualifier[s]]

Command Qualifiers

/BRIEF
/ALL

Argument

ttnn:

If you name a terminal in the command, the display shows in brief form the tasks active at that terminal. If you do not name a terminal, the display shows in brief form the tasks active at your terminal.

Command Qualifiers

/BRIEF

Specifies that you want to display information about active tasks in the brief format. This is the default and need not be specified.

The brief format includes task names and the originating terminal in parentheses next to each task name.

/ALL

Specifies that you want to display information about all tasks active on the system. The default is to show information about tasks active at your terminal only.

Examples

```
$ SHOW TASKS/ACTIVE [RET]
MCR...  (TT5:)
SHOT5   (TT5:)
```

This example shows the brief display from SHOW TASKS/ACTIVE. In this case, the tasks are MCR..., the central command dispatcher, and SHOT5, the SHOW task itself.

```
$ SHOW TASKS/ACTIVE/ALL [RET]
LDR...  (COO:)
RMDEMO  (TTO:)
SHOH2   (HT2:)
NETACP  (COO:)
SHOT5   (TT5:)
DBOFCP  (COO:)
     .
     .
     .
TT4     (TT4:)
EDIT2   (TT2:)
EDIT3   (TT3:)
```

This example shows the brief display from SHOW TASKS/ACTIVE/ALL.

Full Format

SHOW TASK[:taskname]/ACTIVE/FULL

Argument

taskname

If you include a task name, the display shows full information on that task. If you do not name a task, the display shows full information on all currently active tasks. See Volume 2, Chapter 15 for more information on this form of the command.

7.5.1.2 SHOW TASKS/INSTALLED

SHOW TASKS/INSTALLED displays information about installed tasks in either brief or full format.

Format

SHOW TASKS[:taskname]/INSTALLED[/qualifier]

Command Qualifiers

/BRIEF
/FULL

Argument

taskname

Specifies the task for which you want information displayed. If you do not specify a task name, information on all installed tasks is displayed.

Command Qualifiers

/BRIEF

Requests information on installed tasks in a brief format. This is the default and need not be specified. The format of the display is as follows:

```
taskname ident parname priority size ddnn:-lbn [memstate]
```

The elements in this display are as follows:

taskname	The name of the task
ident	The task version identification (or the version of the prototype task)
parname	The partition in which the task is installed
priority	The task's priority
size	The size of the task in bytes
ddnn:	The device from which it is to be loaded
lbn	The logical block number of its disk address
memstate	The task memory state, which can be FIXED, CHECKPOINTED, or blank

If the task version identification is missing (with the rest of the line moved left) or if it is garbage, the task was installed from a disk that is no longer present. If the task version number is a date, such as 07JUL, the task was compiled on that day.

/FULL

Requests the full format of the SHOW TASKS/INSTALLED command. See Volume 2, Chapter 15 for more information on this form of the command.

7.5.2 SET PRIORITY

SET PRIORITY alters the priority of an active task. This is a privileged command.

Format

```
SET PRIORITY
Priority?  pnum
Task?  taskname

SET PRIORITY:n taskname
```

Parameters

pnum

Specifies the new priority you want to assign to the task. Priority numbers can range from 1 through 250.

taskname

Specifies the name of the active task whose priority you want to alter.

Notes

Notice that SET PRIORITY changes both the running and default priority of a task. A task can change its own priority through the ALTP$ Executive directive. With this directive, only the running priority is changed.

There is no SHOW PRIORITY command. You must use SHOW TASKS /FULL instead.

Chapter 8

Preparing a User Batch Job

A batch processor is a task that allows you to have a complete interactive terminal session without being present. The commands and data that you use to complete a session are contained in a user batch job. Each user batch job is run on a software terminal called a "virtual terminal." After your batch job has run, a batch log is created, which is a record of the activity on the virtual terminal.

This chapter discusses how to prepare a user batch job, how to queue batch jobs, and how to read a batch log. This chapter also describes in detail the batch-specific commands, as well as batch-specific error messages.

If you are creating a batch job for the first time, you should be familiar with the various DCL commands and the Indirect Command Processor.

8.1 How a User Batch Job Works

To complete a user batch job, the batch processor creates a virtual terminal. A virtual terminal passes the commands and data in your batch job to CLIs and other tasks. A user batch job completes the following functions:

1. Logs itself in
2. Issues commands
3. Supplies data for tasks
4. Responds to exit status codes returned by tasks
5. Logs itself out

You can queue one or more batch jobs using the SUBMIT command, and this process is called a batch chain.

Each batch job that is submitted to the queue has the same UIC as the terminal from which the SUBMIT command was entered. In other words, the same default login UIC that is created when you log in to an interactive terminal is created for you when your batch job is logged into a virtual terminal.

Note

If you are a privileged user, you can run a batch job under a UIC different from your login UIC.

When a batch job is complete, a batch log is created for your records by the virtual terminal. The batch log includes the commands in the batch chain, the time they were executed, the error messages received, comments, and, optionally, login messages and the data blocks for the job. Once the batch job is completed, the record is spooled to the line printer unless otherwise specified. See Section 8.7 for more information on the batch log.

8.2 How to Prepare a User Batch Job

A user batch job is a file that contains commands and data that run from a virtual terminal. The commands in a user batch job can consisit of DCL commands or indirect command files, or they may be batch-specific commands. The batch processor interprets these commands and processes the information as if you were holding an interactive session.

The batch-specific commands include a login and a logout command, data commands, and label and sequence control commands. The login and logout controls enable the batch processor to log you in and out of each session. The data control commands allow you to include user input in your user batch job. User input is any data that would be input in an interactive session. Label and sequence control commands allow you to program responses to errors that occur from within your user batch job. Section 8.3 has descriptions of each of the batch-specific commands.

8.2.1 The Batch Command Line

The general format of the batch command line is as follows:

```
$ [label:] [command]
```

All commands in batch jobs must be preceded by a dollar sign ($). The dollar sign informs the batch processor that the line is to be interpreted as a command.

A label consists of from one to six characters terminated by a colon (:). Labels, used with the GOTO command, perform error-handling functions. The GOTO command may appear in any part of your batch job and does not have to precede or immediately follow the label. See Section 8.3 for a description of the GOTO command.

Commands include batch-specific commands, CLI commands such as DCL, and indirect command files. All of these commands may be used together in a user batch job. Note that the DCL LOGIN and LOGOUT commands cannot be used in a user batch job because there are batch-specific commands to perform those functions.

DCL commands are documented elsewhere in this manual. Batch-specific commands are described in Section 8.3. Details on using indirect command files are described in Section 8.2.1.3.

8.2.1.1 Comments

The exclamation point (!) is used to include comments in the user batch job. Any information following a comment (!) is not processed by the batch processor or issued to the CLI, but is recorded in the batch log.

Use the following format to enter comments:

```
$!comment
```

Comments (!) can include any alphanumeric character. For example:

```
$!$$$Let's talk cash$$$
```

The first dollar sign is recognized by the batch processor, but the dollar signs and the text following the comment (!) are there for your reference in the batch log.

8.2.1.2 Continuation Lines

DCL commands that do not fit on one line can be continued on the next line by placing a hyphen (-) in the last character position on the line. Continuation lines cannot begin with a dollar sign, a space, or a tab, and they cannot be labeled. For example:

```
$DELETE BIG.MAC;2,JIMMY.MAC;1,LIZ.MAC;1,JAMES.MAC;2,-
MERRI.MAC;5
```

This command line executes as if it were on one line. You cannot use continuation lines with batch-specific commands.

8.2.1.3 Indirect Command Files

The at sign (@) invokes an indirect command file from within a batch job. When the system accesses the indirect command file, the system executes the commands in it. Use the following command line to include an indirect command file in your user batch job:

```
$@indirect[.CMD]
```

You must use the dollar sign to include indirect command files in your batch job. The .CMD file type is optional. You may invoke the command file from the default device and the default directory, or you may specify a device and directory. For example, you can include indirect command files in your batch job as follows:

```
$@BACKUP
```

This command line invokes the BACKUP.CMD file on the default device and the default directory and passes it to the system to be processed.

```
$@DU0:[1,2]BACKUP
```

This command finds the file named BACKUP.CMD on DU0: in directory [1,2] and passes it to the system to be processed.

Note that the default file type for indirect command files is .CMD. The default file type for the SUBMIT command is also .CMD (see Section 8.5. You should be careful not to give batch jobs and indirect command files the same file name because you could inadvertently submit an indirect command file as a batch job. See Chapter 9 for more information about indirect command files.

8.3 Batch-Specific Command Descriptions

This section describes batch-specific commands to be used for batch processing. There are three groups of batch-specific commands:

1. Login and logout commands
2. Data commands
3. Sequence control commands (for error handling)

8.3.1 Login and Logout Commands

JOB

The JOB command marks the beginning of a user batch job and must appear first in each file in a batch chain. Using the JOB command is equivalent to the LOGIN command in DCL.

Format

JOB[/qualifier] [loglabel] [[uic]]

Command Qualifier

/TIME:(hh:mm)

or

/TIME:m

Limits the job to hours (hh) and minutes (mm or m) of CPU time. The default is 3 minutes.

Parameters

loglabel

Specifies the name of your batch job. The loglabel can be up to six alphanumeric characters. This loglabel appears in the batch log and does not affect the processing of your job.

If no loglabel is entered, the corresponding spaces in the batch log are left blank.

[uic]

Specifies the UIC. When specifying a UIC, you must include the brackets in the specification. Only privileged users can specify a UIC different from the UIC under which they submit the batch job.

If you do not include a UIC, the system defaults to the UIC under which you submit your batch job.

The UIC and login messages appear in the batch log. If you want to suppress login messages, use the slash format in your user batch job (see the examples).

Examples

```
$JOB
```

This is the simplest form of the JOB command. It logs the user onto a terminal. The UIC is the same as the UIC the SUBMIT command was issued under. Any system information messages appear in the batch log.

```
$JOB [20/20]
```

This example logs the user in under UIC [20,20] and suppresses login messages.

```
$JOB BATJOB
```

In this example, the loglabel BATJOB appears in the user job heading in the logfile. Note that the loglabel can be up to six alphanumeric characters long.

```
$JOB/TIME:5
```

This example specifies that the job must not exceed 5 minutes of CPU time. The job ends automatically after 5 minutes of CPU time has elapsed.

EOJ

The EOJ command marks the logical end of a user batch job. This command is equivalent to LOGOUT in DCL.

An EOJ command must be placed at the end of each user batch job in a batch chain. The batch processor logs out each job before beginning the next job in a batch chain.

Note that if an end-of-file is encountered before the EOJ command, the batch processor assumes an EOJ command.

Format

EOJ

8.3.2 Data Commands

DATA

The DATA command marks the beginning of a data block included in the batch job. A data block is any required user input other than DCL, indirect commands, or user-written commands for the batch job.

Format

DATA[/qualifier[s]]

Command Qualifiers

/[NO]COPY

Directs the batch processor to print data blocks in the batch log. Data blocks include any block of information that a command in the batch log file may require to execute. This data block must follow the command line.

The /NOCOPY qualifier prevents printing of the data blocks in the batch log.

The default is /COPY.

/DOLLARS[:"string"]

Allows you to include user input that is preceded by a dollar sign ($). Normally, when the batch processor encounters a dollar sign, it assumes this is the end of a data block.

To end a data block when using the /DOLLARS qualifier, you must specify either an EOD command or a "string". A "string" may be up to 15 characters in length and must be placed at the end of your data block. This "string" does not appear in your batch log.

The default "string" is $EOD.

Note that if you use a "string" when specifying the /DOLLARS qualifier, the batch processor ignores all EODs until the "string" is encountered.

Examples

```
$RENAME
PETE.TXT
REPEAT.TXT
$PRINT REPEAT.TXT
```

```
$RENAME
$DATA
PETE.TXT
REPEAT.TXT
$PRINT REPEAT.TXT
```

These two examples are the same. Both examples rename PETE.TXT to REPEAT.TXT and print REPEAT.TXT.

In the first example, the first line without the dollar sign is taken as data because the previous line called for user input. In the second example, the DATA command is used to identify the text following the RENAME command as a data block. In both examples, the next dollar sign encountered marks the end of the data block. Note that the DATA command is optional unless one of its qualifiers is needed.

```
$SAM.TSK
$DATA/DOLLARS
$!The following information is a data block.
$1  Foryou
$2  Forme
$3  Forsam
$!This is the end of the data block.
$EOD
```

In this example, the task SAM requires, as input, information beginning with a dollar sign. To include data beginning with a dollar sign in the task SAM, you must use the DATA command with the /DOLLARS qualifier.

Using the DATA command with the /DOLLARS qualifier allows data beginning with a dollar sign to be passed to the task instead of the batch processor. The information following the DATA/DOLLARS command is a data block. An EOD command is required to end the data block.

```
$JOB NEWJOB
$!This file creates a batch command file called CAT.BAT within the
$!batch job NEWJOB
$COPY TI: CAT.BAT
$DATA/DOLLARS:"MOUSE"
$!The following information
$!is to be included in the file
$!CAT.BAT until the string
```

```
$!"MOUSE" is encountered.
$JOB
$RUN SPOT.TSK
$DATA/DOLLARS
$40.00  Vets
$20.00  Chow
$10.00  Grooming
$EOD
$EOJ
MOUSE
$EOJ
```

In this example, a batch job is created called NEWJOB. When the batch processor encounters another JOB command, another batch command file is created called CAT.BAT. Information is read from the TI: (virtual terminal) and copied to the file CAT.BAT. Because a "string" MOUSE was used in the file CAT.BAT, the batch processor ignores all EOD commands until the string "MOUSE" is encountered. The file CAT.BAT is a separate batch command file with a batch job.

Note that the first EOD command and the first EOJ command are ignored by the batch processor, but are included in the file CAT.BAT. The last EOJ command ends the batch job. Remember, if a "string" is not used, the batch processor ends the data block when it encounters an EOD command.

EOD

The EOD command marks the end of a data block. If you use the DATA command with the /DOLLARS qualifier, the data block is closed by default when an EOD is encountered. If you used a "string" with /DOLLARS, the batch processor ignores all EOD commands until the "string" is encountered.

Format

EOD

8.3.3 Sequence Control Commands

STOP

The STOP command stops the batch job. You can use it alone or with the ON or IF commands.

Format

```
STOP
ON  statuscode THEN STOP
IF  statuscode THEN STOP
```

A STOP command stops the batch processor. This command may be used alone or with an ON or IF command. the status codes are WARNING, ERROR, and SEVEREERROR.

A STOP command without a qualifier or command following it is interpreted by the batch processor. A STOP command followed by a qualifier or any other command line is passed to the current CLI.

You may have more than one STOP command in a user batch job when it is used with the ON or IF commands.

If you stop a user batch job in a batch chain, the batch processor continues with the next batch job.

Example

```
$JOB
$!This is a test.
$STOP
```

This is the simplest form of the STOP command. In this example, the STOP command is equivalent to an EOJ command.

If you use a STOP command with the ON or IF command, the batch processor does not halt the batch job, but rather performs the action specified by these commands.

Note that if your batch job contains another batch job, the batch processor continues with the next user batch job in the batch chain.

CONTINUE

The CONTINUE command can be used with the ON or IF commands. This command allows the batch processor to continue processing jobs when it encounters a status code other than SUCCESS.

The use of the CONTINUE command alone does not affect processing.

Format

```
CONTINUE
ON  statuscode THEN CONTINUE
IF  statuscode THEN CONTINUE
```

The status codes are WARNING, ERROR, and SEVEREERROR.

GOTO

The GOTO command instructs the batch processor to move to a line that contains a specified label and begin processing from there. This command can reference forward or backward. Use it alone or with the ON or IF commands.

Format

```
GOTO  label
 .
 .
 .
label:[command]
```

Parameter

label:[command]

The label can be up to six alphanumeric characters long. The referenced label may precede or follow the GOTO command line. Note that the referenced label must have a colon (:) following it. The command following the label is optional.

Examples

```
$GOTO NEXJOB
     .
     .
     .
$NEXJOB:DIR
```

In this example, the batch processor is instructed to skip to the label NEXJOB. The NEXJOB reference uses the DIR command to display directories.

```
$ON WARNING GOTO MOON
```

In this example, the batch processor is instructed to skip to the label MOON if a WARNING or more severe error is encountered.

Note that your referenced label may precede or follow the GOTO command line. Because of this, it is possible to get caught in an infinite loop. To avoid looping, use the /TIME qualifier with the JOB command so that you can specify a limit to your batch job.

ON

The ON command, when used with the STOP, CONTINUE, or GOTO commands, specifies the action to be taken by the batch processor if an error occurs.

Format

ON statuscode THEN STOP

ON statuscode CONTINUE

ON statuscode GOTO label

Parameter

statuscode

The status codes are WARNING, ERROR, and SEVEREERROR.

The action taken by the batch processor depends on the status code you use and the commands you use it with. For example, if you use ON WARNING THEN STOP, the processor stops the batch job when a message other than success is returned.

If you choose ERROR, the batch processor ignores all warning messages in your batch job, but performs the specified action for all ERROR and SEVEREERRORs encountered.

If you choose SEVEREERROR, the batch processor ignores both warnings and errors encountered in your batch job and performs a specified action when a SEVEREERROR is encountered.

If no warnings or errors are encountered, the batch processor assumes a SUCCESS statuscode. ON SUCCESS is not a valid command, however.

Note that if the status code is encountered, the ON command defaults to ON WARNING THEN STOP until another ON or SET command is encountered. You may set the ON command at any point in a user batch job.

Examples

```
$ON WARNING THEN STOP
```

This is the default setting for the ON command. If you do not override it with another ON command or a SET command, any WARNING or more severe status return code will stop the batch job.

```
$ON ERROR THEN GOTO JEEPERS
```

Once the batch processor encounters this command, it ignores all WARNINGs. When the batch processor receives an ERROR or SEVEREERROR code, it skips directly to the line labeled $JEEPERS: and processes that command and all commands following that label.

SET

The SET command enables or disables the ON command setting.

Format

```
SET  NO ON

SET  ON
```

The SET NO ON command disables the ON command setting, including the default. The batch processor ignores the status codes set by the ON command until a SET ON or IF command is encountered. Any errors detected while in a SET NO ON state are included in the batch log.

SET ON enables the ON command after a SET NO ON command has been issued. The batch processor returns to the status set by the last ON command.

IF

The IF command is used to check the status code of a given command in a batch job. When an IF command is specified and the status code is encountered, IF specifies the action to be taken by the batch processor.

Format

IF statuscode THEN STOP

IF statuscode CONTINUE

IF statuscode GOTO

Parameters

statuscodes

The status codes are WARNING, ERROR, and SEVEREERROR.

The IF command overrides an ON command if the status code in the IF command line and the status code in the ON command line match.

Note that the IF command also overrides a SET command.

Example

```
$JOB KIDS
$!IF command example
$RUN PARENT.TSK
$!The task PARENT.TSK uses three status codes to determine whether
$!the batch job should continue. If any of the status codes are encountered,
$!the job moves to the label specified in the IF command line.
$IF SEVEREERROR THEN GOTO DOOM
$IF ERROR THEN GOTO BED
$IF WARNING THEN GOTO ROOM
$!PARENT.TSK moves to the label PLAY if the exit status of the task
$!does not match the conditions set by the IF command.
$GOTO PLAY !If success assumed
$!If the status code SEVEREERROR is encountered, the job moves to the
$!label DOOM, broadcasts to the terminal and STOPS the batch job.
$DOOM:
$BRO TT21: "Severe error. Run ends".
$STOP !If SEVEREERROR encountered.
$!If the status code ERROR is encountered, the job moves to the label
$!BED, broadcasts to the terminal, and starts a new task called
```

```
$!NEWKID.TSK. Once the NEWKID.TSK has completed, the job moves
$!to the label PLAY.
$BED:
$BRO TT21: "Error encountered. Running NEWKID.TSK"
$RUN NEWKID.TSK !If ERROR encountered.
$GOTO PLAY
$!If the status code WARNING is encountered, the job moves to the label
$!ROOM, broadcasts to the terminal, and moves to the label PLAY.
$ROOM:
$BRO TT21: "WARNING received. Job continues".
$GOTO PLAY
$!The label PLAY specifies an end-of-job command.
$PLAY:EOJ
```

In this example, three actions are given by the IF command. Should the exit status of a task match the conditions set by the IF command, the specified action is taken. Any ON command in effect is ignored.

8.4 Allocating Devices and Mounting Volumes from Batch Jobs

Your user batch job may require allocating devices and mounting volumes. If it does, running your batch job in your absence presents some problems, because you may not be present physically to load the device or to mount the volume. The batch processor cannot issue commands involving that volume, if the device is not allocated and the volume is not mounted.

If you have an operator who can physically load the device and mount the volume in your absence, you may use the ALLOCATE command and the MOUNT command.

Because you will not know in advance which drive is available, the ALLOCATE command uses a generic qualifier to allocate the first available drive. Likewise, the MOUNT command has a qualifier that enables the batch processor to pause until your volume is physically mounted.

Use the following format in your user batch job to allocate a drive:

```
$ ALLOCATE[/TYPE:devtype]dd: ddn:
```

The qualifier /TYPE is necessary whenever a device name (dd) refers to more than one kind of device. Once you have specified the above command line, the ALLOCATE command allocates the first available drive and assigns the logical name (ddn:) you specified.

The MOUNT command follows the ALLOCATE command in your batch job. Use the MOUNT command with the /WAIT qualifier to notify the operator (by sending a message to the console terminal) to load the volume with the specified label on a specified drive. For example:

```
$MOUNT/WAIT ddn: label
```

The MOUNT command pauses until the operator loads the device and restarts the MOUNT command. When the MOUNT command is completed, the remaining commands in the batch job are executed.

Note that you must use the DEALLOCATE command and the DISMOUNT command as you would in an interactive session.

Examples

```
$JOB
$!This batch job restores files from an RX51
$!disk pack labeled CHECKME to the fixed disk.
$ALLOCATE/TYPE:RX50 DU: INO:
$MOUNT/WAIT INO: CHECKME
$BACKUP/NOINI/DIR INO: DUO:
$DISMOUNT INO:
$DEALLO INO:
$EOJ
```

In this example, you do not know the physical device name, so you give it a logical device name, IN0:.

If you intend to load and allocate the device yourself on a specific drive, you can use the MOUNT/NOSHAREABLE command, as shown in the next example:

```
$JOB
$!This batch job will get a full directory listing
$!of an RX51 diskette that is already loaded and
$!spun up.
$MOU DU1:IMREADY/NOSHAREABLE
$DIRECTORY/PRINT DU1:[*]/FU
$DMO DU1:
$EOJ
```

8.5 SUBMIT

The SUBMIT command is used to enter user batch jobs into the batch queue. If more than one batch job is submitted to the queue, it is called a batch chain.

This section explains the qualifiers that are available with the SUBMIT command to control the processing of your batch jobs.

Format

SUBMIT[/qualifier[s]] filespec[s][/filequalifier[s]]

Command Qualifiers

```
/AFTER:(dd-mmm-yy hh:mm)
       (mm/dd/yy hh:mm)
/[NO]HOLD
/NAME:jobname
/[NO]RESTART
/PRIORITY:n
/QUEUE:queuename
```

File Qualifiers

```
/[NO]DELETE
/[NO]TRANSFER
```

Batch Logfile Qualifiers

```
/NAME:jobname
/[NO]LOGFILE
/[NO]PRINT
/PRINT:printqueue
```

Parameter

filespec[s]

Specifies the file(s) containing the user batch job(s).

If you include multiple file specifications, separate them using commas. You can use wildcards (*) in the directory, name, type, and version fields of the file specifications.

If your file specification includes no file type, the default file type is .CMD.

Command Qualifiers

/AFTER:(dd-mmm-yy hh:mm)
(mm/dd/yy hh:mm)

Blocks the job until after the specified time. Depending on the status of other jobs in the batch queue, your job may be run immediately or at a later time.

You can specify the calendar filed in either of two formats:

dd-mmm-yy	Uses a 1- or 2-digit number for the day, the first 3 letters for the month, and a 2-digit number for the year.
mm/dd/yy	Uses a 1- or 2-digit number for the month, a 1- or 2-digit number for the day, and 2-digit number for the year.

If you leave out the calendar field, it defaults to the current date.

If you leave out the clock field, the time defaults to 00:00:00.

/[NO]HOLD

Specifies that the job be held in its queue. You can release the job with the RELEASE/JOB command.

/NAME:jobname

Specifies a job name for a batch job. The job name can be up to nine alphanumeric characters in length.

If you specify a job name, that name appears in the output from the SHOW QUEUE commands. If you do not specify a job name, the job name defaults to the name of the first file specification submitted to the queue.

If you specify /NOPRINT and a job name, the file name of the batch log is the same as the job name and the file type is .LOG.

Note that you can label individual jobs in a batch chain using the label field of the JOB command.

/[NO]RESTART

Indicates whether or not a job should be restarted if it has been interrupted.

/RESTART causes the batch job to be put in a held state if it has been interrupted. To start the job again, you must use the RELEASE\JOB command.

/NORESTART does not change the status of a batch job that has been aborted. A job that has been aborted is deleted from the queue. This is the default.

/PRIORITY:n

Sets the queue priority of the batch chain. The highest priority job runs first. For example, if two jobs have the same priority, the job that has been in the queue longer runs first.

For nonprivileged jobs, n can be any number from 1 through 150. Privileged users can set priority from 1 through 250.

The default for n is 50.

This qualifier does not affect the running priority of tasks included in your batch job.

/QUEUE:queuename

Specifies the name of the batch queue in which the job is to be placed.

The default is the BATCH queue.

File Qualifiers

/[NO]DELETE

Indicates whether a user batch job(s) should be deleted or not from the user's directory.

The /DELETE qualifier marks a user batch job(s) for deletion from the user's directory after the job has completed processing. The batch job(s) is always deleted from the batch queue.

The default is /NODELETE.

/[NO]TRANSFER

Specifies whether or not the system should make temporary copies of batch command files that have been submitted from a volume mounted on a nonpublic device.

If you use the /TRANSFER qualifier, you may dismount volumes after the SUBMIT command has completed processing. This is the default.

Note: if a volume has files such as task files that are needed to complete a batch job, do not dismount the volume.

When you specify the /NOTRANSFER qualifier, no copies are made. The volumes must remain mounted until batch processing is complete.

Batch Logfile Qualifiers

/NAME:jobname

Gives a name to the batch job and the batch log.

In the batch log file, this name appears in the page heading.

When you issue the SHOW QUEUE command, this name appears in the display.

If you do not specify a name, the name defaults to the first file name in your batch job.

/[NO]LOGFILE

Specifies whether a batch log file should be created or not.

/LOGFILE produces a batch log file.

/NOLOGFILE suppresses the creation of a batch log file.

The default is /LOGFILE.

/PRINT[:printqueue]

Causes the batch log file to be queued to a line printer and deleted after it is printed. The temporary file is created in SP:[1,7].

Although the PRINT queue is normally used, you may spool the batch log file to any print queue by specifying an optional queuename.

The default is /PRINT:PRINT.

/NOPRINT

Suppresses the automatic printing of the batch log, but creates a permanent copy of the batch log file. The file can be found under the same directory and device name as the first batch job in your batch chain.

The batch log file name is the same as the batch job name with the file type .LOG.

Examples

```
$ SUBMIT SUPER.BAT RET
```

In this example, the batch file SUPER.BAT is spooled to the default batch queue BATCH. The name of the job is SUPER. The log file SUPER.LOG will be printed and then deleted.

```
$ SUBMIT/PRINT SUPER.BAT RET
```

In this example, the /PRINT qualifier is the default. The command is equivalent to the previous one.

```
$ SUBMIT/NOPRINT SUPER.BAT RET
```

In this example, a permanent log file is created in your directory. After the batch job completes, you may examine the log file SUPER.LOG at your terminal or use the PRINT command to spool it to a line printer.

```
$ SUBMIT/NOLOGFILE SUPER.BAT RET
```

In this example, no log file is produced.

```
$ SUBMIT CLARK.BAT,ROBIN.BAT,SUPER.BAT RET
```

The batch files in this example are processed in the order submitted. They are processed by the default batch processor, without interruption. Each file contains a user batch job. The three files when submitted in this manner are a batch chain. The name of the batch chain is CLARK and the name of the log file is CLARK.LOG. The log file contains a record of each of the three user batch jobs.

```
$ SUBMIT/NAME:POOH BEAR.BAT,TIGGER.BAT/DELETE,CHRIS.BAT RET
```

In this example, the name POOH is given to the batch chain. The batch file TIGGER.BAT is deleted after it is processed.

```
$ SUBMIT/QUEUE:BAT2/AFTER:(2-JUN-85) MOMSIE.BAT RET
```

In this example, the batch file MOMSIE.BAT is submitted to the queue BAT2. MOMSIE.BAT will be processed on June 2, 1985 after midnight.

If you want to submit a batch file from a directory other than your own, you must specify the directory and the file name. For example, if your directory is [KOOL], you may submit a batch file from the directory [HEATWAVE] as follows:

```
$ SUBMIT/NOPRINT [HEATWAVE]BURN.BAT RET
```

In the above example, the batch file BURN.BAT is submitted from the directory [HEATWAVE] with the /NOPRINT qualifier. The /NOPRINT qualifier suppresses the automatic printing of the batch log, but creates a permanent copy of the batch log file.

8.6 Queuing Jobs

The Queue Manager (QMG) is a system task that distributes jobs to output devices or batch processors. Passing jobs to the Queue Manager is called "queuing jobs." When the Queue Manager releases jobs for processing, this action is called "dequeuing jobs."

The Queue Manager distributes jobs to batch processors. These user batch jobs are queued by the SUBMIT command and are called QMG batch jobs. A QMG batch job is a chain of one or more user batch jobs to be processed. The SUBMIT command specifies a QMG batch job.

Once jobs are in queues, you can display queue information with the SHOW QUEUE command. You can alter the status of jobs in queues with the SET QUEUE command. You can hold jobs in queues with the HOLD/ENTRY command or release them from queues with the RELEASE /ENTRY command.

8.6.1 How to Use the Queue Manager for Batch Jobs

Once you have issued a SUBMIT command, you may want to alter how that file is processed or you may decide that you did not want to process that file at all. This section explains how to manipulate the QMG so that your file is processed the way you want it to be.

For example, you issue the following SUBMIT command to process the file ANTRIN.BAT to the line printer:

```
$ SUBMIT ANTRIN.BAT [RET]
SUB - Job 22, name "ANTRIN   ", submitted to queue "BATCH "
```

However, you did not want to submit that file. The batch file you wanted to process is ANTRIM.BAT. To prevent ANTRIN.BAT from being processed, you must remove that file from the BATCH queue. To do this, you use the SHOW QUEUE command and the DELETE/ENTRY command.

The SHOW QUEUE command shows you the position of your job on the queue (as well as other jobs) and whether or not your job is currently being processed. The following example shows you how to issue the SHOW QUEUE command, and displays the resulting output.

```
$ SHOW QUEUE [RET]
** PRINT QUEUES **
PRINT  =>  LPO
LPO    =>  LPO
** BATCH  QUEUES **
BATCH  =>  BPAO
  [303,3]   ANTRIN    ENTRY:824           ACTIVE ON BAPO
            1 DUO:[QUAINT]ANTRIN.BAT;1
$
```

ANTRIN.BAT has an entry number of 824 in the BATCH queue. The Queue Manager assigns an entry number to each job as a means of keeping track of which jobs are processed when. Usually, this is done on a first-come, first-served basis. Also, ANTRIN.BAT is currently being processed, as noted by the seventh line of the display. To stop your job from processing further (or if your job has not begun to be processed), you have to issue the DELETE/ENTRY command. This command stops your job immediately. You use the DELETE/ENTRY command in the following manner:

```
$ DELETE/ENTRY:824 [RET]
$
```

Your job has been removed from the BATCH queue and will not be processed any further. However, a batch log is still created even when you abort the batch job. When your job is finished (or aborted), the batch log is put on the PRINT queue to be printed on the line printer. (For more information on the batch log file, see Section 8.7.) The output from the SHOW QUEUE command after you delete your job from the batch processor is as follows:

```
$ SHOW QUEUE [RET]
** PRINT QUEUES **
PRINT  =>  LPO
LPO    =>  LPO
  [303,3]   ANTRIN    ENTRY:824         ACTIVE ON LPO
            1 SPO:[1,7]ANTRIN.LOG;1                   DELETE
** BATCH  QUEUES **
BATCH  =>  BPAO
$
```

Once ANTRIN.BAT is removed from the BATCH queue, reissue the SUBMIT command with ANTRIM.BAT as the file you want processed.

The following sections detail the various commands for using the QMG, their qualifiers, and examples of how the commands are used.

8.6.2 DELETE

The DELETE command is used to delete jobs from a queue or files from a job.

Format

DELETE/ENTRY:nnn[/FILE_POSITION:n]

This command allows the deletion of a job in queue by specifying the job's entry number. You can also delete a single file in a job by specifying the /FILE_POSITION:n qualifier.

Parameters

/ENTRY:nnn

Deletes a queue entry by number. The number is unique.

/FILE_POSITION:n

Identifies the file by the file's position in the job.

Examples

```
$ DELETE/ENTRY:301 RET
```

This command deletes the job from queue by referencing the job's unique entry number (in this example, job number 301).

```
$ DELETE/ENTRY:301/FILE_POSITION:2 RET
```

This example deletes only the second file appearing in job 301. You may only refer to the file you wish to delete by the file's position in the print job (the numbered order in which you entered the file specification). You cannot specify the file you wish to delete by referring to its file specification.

Notes

Associating a particular file specification with the numbered order in which it was entered into a queue can be accomplished by entering a SHOW QUEUE command, and examining the output on your terminal.

DELETE/QUEUE works on all categories of jobs in any queue. You can delete ACTIVE jobs, WAITING jobs, TIME-BLOCKED jobs, or HELD jobs.

There can be more than one job with the same name from the same UIC. The DELETE/QUEUE command will delete the first job of a given name in the queue.

8.6.3 SHOW QUEUE and SHOW PROCESSOR

You may use the commands described in this section to display information about queues, the jobs in the queues, spooled devices, and batch processors.

The commands described in the following section allow you to access information about QMG batch jobs.

A SHOW QUEUE command will show all jobs in all queues. Qualifiers can be used to limit the display to specific queues, specific jobs, and the like.

You may examine the contents of your system's queues (in decreasing degree of detail) by entering one of the following command qualifiers:

/FULL
/FILES
/BRIEF

The commands described in Section 8.6.3.2 display information about spooled devices and batch processors under the control of the Queue Manager.

8.6.3.1 SHOW QUEUE

SHOW QUEUE displays information about QMG batch jobs.

Format

SHOW QUEUE   [queuename][/qualifier[s]]

Command Qualifiers

/FULL
/FILES
/BRIEF
/ENTRY:nnn
/NAME:jobname
/OWNER_UIC:uic
/BATCH

Command Qualifiers

/FULL

Displays detailed information about queues, queue assignments, jobs, the attributes of jobs in queues, and files that compose jobs in queues.

/FILES

Displays information about queues, queue assignments, jobs in queues, and files that compose jobs in queues. The attributes of the jobs are not displayed. This display format is the default of SHOW QUEUE.

/BRIEF

Displays only queues, queue assignments, and jobs in queues.

You may examine specific attributes of queues by entering one of the following qualifiers.

SHOW QUEUE queuename

Displays QMG information for a single queue. The information displayed may be for batch queues.

/ENTRY:nnn

Limits information to a particular job entry referenced by the job's unique entry number.

/NAME:jobname

Limits information to jobs with the specified job name.

/OWNER_UIC:uic

Limits information to jobs owned by the specified uic.

/BATCH

Limits information to batch queues.

Examples

The following examples show the three different output displays (FILE, FULL, BRIEF).

```
$ SHOW QUEUE RET
** PRINT QUEUES **
PRINT  => LPO
LPO    => LPO
** BATCH QUEUES **
BATCH  => BAPO
  [7,40]MJRA         ENTRY:23                    ACTIVE ON BAPO
       > 1 DU1:[7,40]MJRA.BAT;1
$
```

This display, the default format, shows the queues in the same state. It does not show the attributes of the job. It only shows the form number of the job and the attributes of the files if other than the default values were specified.

```
$ SHOW QUEUE/FULL [RET]
** PRINT QUEUES **
PRINT  => LPO
LPO    => LPO
** BATCH QUEUES **
BATCH  => BAPO
  [7,40]MJRA      ENTRY:23                ACTIVE ON BAPO
      PRI:50  NORESTART  LOG  PRINT:PRINT
       > 1 DU1:[7,40]MJRA.BAT;1                     NODELETE
$
```

This display shows all information about the queues and jobs in queues. In this example, the batch job [7,40]MJRA, entry number 23, is running on batch processor BAP0.

The attributes of the job are indicated in the display, and the files that make up each job, are also listed. The angle bracket (>) indicates which file of a job is currently being processed.

```
$ SHOW QUEUE/BRIEF [RET]
** PRINT QUEUES **
PRINT  => LPO
LPO    => LPO
** BATCH QUEUES **
BATCH  => BAPO
  [7,40]MJRA         ENTRY:23                ACTIVE ON BAPO
$
```

This display only shows the names, entry numbers, and status of the jobs in the queues.

8.6.3.2 SHOW PROCESSOR

SHOW PROCESSOR displays information about the initialized characteristics of spooled devices and batch processors, printers, and other output devices under control of the Queue Manager.

Format

SHOW PROCESSOR [processorname]
Displays information about all processors.

SHOW PROCESSOR/BATCH
Shows all batch processors.

Example

Enter the following command line on your terminal:

```
$ SHOW PROCESSOR RET
```

Information about print processors, batch processors, and spooled input devices will be displayed at your terminal in the following manner:

```
** SPOOLED DEVICES **
LP0    <= PRINT LP0
** BATCH PROCESSORS **
BAP0   <= BATCH
          CURRENT JOB:   [303,3]ENGINE     ENTRY:588
$
```

This display shows all information about spooled devices and batch processors.

The only batch processor, BAP0, can only receive jobs from the Queue BATCH.

8.6.4 SET QUEUE

SET QUEUE modifies attributes given to batch jobs or files that compose jobs in queues. Such jobs and files have been entered in queues by the SUBMIT command. You cannot change the attributes of an active job.

Format

SET QUEUE/ENTRY:nnn/qualifier[/qualifier[s]]

Job Qualifiers

/AFTER:(hh:mm dd-mmm-yy)
/PRIORITY:n
/[NO]RESTART

File Format

SET QUEUE/ENTRY:nnn/FILE_POSITION:n/qualifier[/qualifier[s]]

File Qualifiers

/FILE_POSITION:n
/[NO]DELETE

Job Qualifiers

/AFTER:(dd-mmm-yy hh:mm)

Changes the time after which your job will be processed. The job will be BLOCKED until the time and date you specify. The job will not necessarily be processed at exactly the time you state, but will be eligible after the time you state.

If you do not supply the calendar field, the default is the current date. If you do not supply the clock field, the default is midnight on the date given in the calendar field.

If you supply both the clock and calendar fields, you must separate them with a space.

/PRIORITY:n

Changes the queue priority of a batch job. Nonprivileged users may set priorities up through 150. Privileged users may set priorities up through 250.

/[NO]RESTART

Changes the restartability of your job. If you specify /RESTART, your job will start again from the beginning if it is interrupted while ACTIVE. If you specify /NORESTART (the default) your job will pick up where it left off, if it is interrupted while ACTIVE.

File Qualifiers

/FILE_POSITION:n

Changes the operation of /[NO]DELETE as it applies to a file contained in a job in queue. The number n refers to the file position in the job. Use SHOW QUEUE to determine its position.

/[NO]DELETE

Changes the delete status of a single file contained in a batch job.

Examples

```
$ SUBMIT MEDSURG.BAT RET
```

The SHOW QUEUE/FULL command output might look like this:

```
** PRINT QUEUES **
PRINT  => LPO
LPO    => LPO
** BATCH QUEUES **
BATCH  => BAPO
  [303,3]   MEDSURG   ENTRY:43             ACTIVE ON BAPO
```

```
       PRI:50  NORESTART  LOG  PRINT:PRINT
         >    1 DUO:[303,3]MEDSURG.BAT;1                    NODELETE
$
```

To change the priority of the file MEDSURG.BAT from a priority of 50 to 100, you would type the following SET QUEUE command on your terminal:

```
$ SET QUEUE/ENTRY:43/PRIORITY:100 RET
```

The SHOW QUEUE/FULL command output on your terminal would look like this:

```
** PRINT QUEUES **
PRINT  => LPO
LPO    => LPO
** BATCH QUEUES **
BATCH  => BAPO
  [303,3]   MEDSURG   ENTRY:43              ACTIVE ON BAPO
       PRI:100 NORESTART  LOG  PRINT:PRINT
         >    1 DUO:[303,3]MEDSURG.BAT;1                     NODELETE
$
```

Note that the file MEDSURG.BAT now has a priority of 100 instead of 50.

Notes

When you issue the SUBMIT command, you specify attributes of the QMG job through command qualifiers. SET QUEUE command qualifiers change the matching attributes.

You cannot change the attributes of an active job.

You can delete files by specifying the /FILE_POSTION:n qualifier in your SET QUEUE command line.

8.6.5 Holding and Releasing Jobs

You can specify that a batch job be HELD in queue when you issue the SUBMIT command.

You can release such jobs with the RELEASE command.

8.6.5.1 HOLD

HOLD blocks a job in its queue until it is explicitly released.

Format

HOLD/ENTRY:nnn

Remember, jobs may share the same name but never the same entry number.

8.6.5.2 RELEASE

RELEASE unblocks a job that has been held in queue.

Format

RELEASE/ENTRY:nnn

Examples

```
$ SUBMIT/HOLD LAYLA.BAT RET
```

The SHOW QUEUE/FULL command output might look like this:

```
** PRINT QUEUES **
PRINT  => LP0
LP0    => LP0
** BATCH QUEUES **
BATCH  => BAP0
  [303,3]   LAYLA     ENTRY:25             HELD
      PRI:50  NORESTART  LOG  PRINT:PRINT
       >    1 DU0:[303,3]LAYLA.BAT;1                 NODELETE
$
```

To unblock LAYLA.BAT, you would type the following RELEASE command on your terminal:

```
$ RELEASE/ENTRY:25 RET
```

The SHOW QUEUE/FULL command output would look like this:

```
** PRINT QUEUES **
PRINT  => LP0
LP0    => LP0
** BATCH QUEUES **
BATCH  => BAP0
  [303,3]   LAYLA     ENTRY:25             ACTIVE ON BAP0
      PRI:50  NORESTART  LOG  PRINT:PRINT
       >    1 DU0:[303,3]LAYLA.BAT;1                     NODELETE
$
```

Note, the HELD status of the job in the previous example was first changed to WAITING, and then, since there were no jobs in queue, became ACTIVE.

8.7 The Batch Log File

A batch job creates a batch log file that contains a record of activity on the virtual terminal. Unless otherwise specified, the batch log is automatically printed on your system's line printer after batch processing is complete. Note that if you have your batch log file spooled to a printer, the temporary file is deleted after batch processing is complete.

The batch log contains the commands in the batch chain, the time the commands were executed, the error messages received, any comments that you placed in your batch job and, optionally, login messages and the data blocks for the job. The log also includes all system output that would normally appear on your terminal during an interactive session. Any spooled output is also appended to the print job that contains the batch log file. This is called the log file print job.

If you specify the /NOLOG qualifier, the batch log is not spooled to the printer, but your system output is spooled as separate print jobs.

The batch log file is divided into three parts: the log file heading, the user job flag information, and a printout of the user batch job activity.

The log file heading contains the following:

- Title "QMG Batch Job"
- Batch job name
- Version number of the batch processor
- Day, month, year, and time the batch job began
- Batch log page number
- Task name of the batch processor

The user job flag displays the following:

- User job name
- Device name of the logged-in virtual terminal
- UIC under which the batch job was logged in

Note that the previous information is followed by system messages from a successful login.

The remaining part of the batch log is a record of activity which includes the following:

- Command lines and comments preceded by a time stamp
- Output from commands and tasks (normally written to your terminal)
- Data for commands and tasks
- Lines skipped by a GOTO statement
- Exit status messages

TERM, DATA, and SKIP are line identifiers that appear in the spaces between the time stamps and the command lines in the log file. TERM identifies the line as output from a task to the virtual terminal, DATA identifies the line as input data to a task, and SKIP indicates that the line was skipped because of a GOTO command.

Periods that appear below an identifier indicate a continuation of that particular line identifier.

Your batch log file may also contain the status code that an ON command has met or exceeded when a task exited. A display is also included in the batch log of each task's exit status.

Note that when data appears on the same line as a prompt, it causes the line to be labeled TERM rather than DATA. If a GOTO is specified, with the ON and IF commands enabled, the lines skipped are labeled as SKIP.

Example

```
$JOB ORWELL
$!Generate and print calendars for year 1985
$ON ERROR THEN GOTO BADCAL
$RUN CALENDAR
1985
$EOD
$GOTO SPOOL
$BADCAL: ON ERROR THEN GOTO NOFILE
$RUN NEWCAL
L
1985
$SPOOL:
$PRINT BIGSIS:=1985.CAL/CO:10
$IF ERROR THEN GOTO DIFFQ
$DIFFQ: PRINT/COPIES:10 1985.CAL
```

```
$STOP
$NOFILE:
$!Could not create calendar file
$EOJ
```

In this example, a user batch job is created called ORWELL. The following text shows the batch log file that the user batch job ORWELL produces:

```
QMCBatch Job - CALENDAR    BPR V1.0   23-MAY-85  11:03  Page 1
Processor BAPO
11:03          $JOB ORWELL
      ==================================
      User Job - ORWELL    Terminal VT2:
               UIC = [301,31]
      ==================================
        TERM
         .
         .   Micro/RSX  V3.0
         .
         .
11:03:54       $!Generate and print calendars for year 1985
11:03:54       $ON ERROR THEN GOTO BADCAL
11:03:55       $RUN CALENDAR
        TERM
         .     ** Calendar generator program - Version 1.0
         .
         .     Enter year:1985
         .     Sorry, year must not exceed 1983
        'ERROR' Exit status returned - enabling action "ON" command
         SKIP $EOD
          .    $GOTO SPOOL
11:03:56       $BADCAL: ON ERROR THEN GOTO NOFILE
11:03:56       $RUN NEWCAL
         TERM
          .    ** Calendar generator program - Version 1.5
          .
          .    Enter 'S' for small format, 'L' for large format: 1
          .    Enter year: 1985
          .
          .    File 1985.CAL has been created
          .
11:04:01       $SPOOL:
11:04:01       $PRINT BIGSIS:1985.CAL/CO:10
         TERM PRI - Extraneous input
11:04:03       $IF ERROR THEN GOTO DIFFQ
         SKIP $STOP
11:04:04       $DIFFQ: PRINT/COPIES:10 1985.CAL
         TERM PRI -Job 13, name "1985   ",submitted to queue "LPO  "
```

```
              $STOP
11:04:08 TERM Connect time:    0 hrs  1 mins  0 secs
            . CPU time used:   0 hrs  0 mins  4 secs
            . Task total:     16
```

8.8 Error Messages

The following sections describe error messages that occur from batch-specific commands. Commands in batch jobs can also create error messages that appear in your batch log.

In addition, the batch processors themselves can send error messages concerning your batch jobs. Some of these messages result from system problems and are returned to the operator's console. Others reflect difficulties in processing your batch job and are returned to your batch log. Section 8.8.1 describes error messages which appear in your batch log. Section 8.8.2 describes messages that are sent to the operator's console.

8.8.1 Error Messages in Batch Logs

BPR—Batch file already open

Explanation: BPR (batch processor) attempted to open the specified file when it was already open.

User Action: System problem. See your operator or system manager.

BPR—Batch file close failure

Explanation: BPR failed to close the specified file.

User Action: The batch job should not be affected. Check the directory to see if the file is locked. If so, issue the DCL UNLOCK command.

BPR—Batch file deletion failure

Explanation: The /DELETE qualifier to the SUBMIT command failed to execute.

User Action: Check the directory. The file may not be in the directory, or the batch job may lack delete access to the file. Delete the file with the DCL DELETE command.

BPR—CPU time limit exceeded, user job terminated

Explanation: The batch job ran longer than the /TIME qualifier to the JOB command permitted.

User Action: Retry the batch job and increase the amount of CPU time using the /TIME qualifier.

BPR—I/O error

Explanation: The batch processor failed to read from or write to the virtual terminal, or the batch processor was unable to open the batch file.

User Action: Retry the batch job. This may indicate a hardware error. See your operator or system manager.

BPR—Virtual terminal I/O was aborted

Explanation: The task sending I/O to the virtual terminal was aborted.

User Action: None.

BPR—Label undefined

Explanation: The batch job specified a label that was not in the job.

User Action: Check the job to see that the label is present in proper form. The label must begin with a dollar sign ($) and end with a colon (:).

BPR—Log file directory not found - aborted batch job

Explanation: The SUBMIT command included the /NOPRINT qualifier, but the batch job cannot find the directory for the batch log.

User Action: See your system manager.

BPR—Logon privilege violation

Explanation: The $JOB command gave an incorrect UIC.

User Action: Check the command for proper syntax and retry the job. For nonprivileged users, the UIC in $JOB must match the login UIC in effect when the SUBMIT command was entered.

BPR—Output request from incorrect virtual terminal

Explanation: System error.

User Action: See your operator or system manager.

BPR—Spawn failure

Explanation: System error.

User Action: See your operator or system manager.

BPR—Specified maximum CPU time too large

Explanation: The largest amount of CPU time that may be specified is 65535 minutes (1092 hours, 15 minutes).

User Action: Specify less CPU time.

BPR—Syntax error

Explanation: A command line in the batch job does not start with a dollar sign ($), or special batch commands have improper syntax.

User Action: Check the batch file for proper syntax and retry the job.

BPR—Syntax error—$JOB does not appear first

Explanation: $JOB logs in the user batch job on the virtual terminal and must appear first in the batch file.

User Action: Edit the batch file so that $JOB appears first.

BPR—Virtual terminal output too long for buffer

Explanation: A stream of characters from the virtual terminal was too long to fit in a storage buffer.

User Action: Make sure that the task outputs RETURN and LINEFEED characters when performing terminal I/O.

8.8.2 Error Messages to the Operator's Console

BPR—Batch file input error

Explanation: BPR could not read from the batch file.

User Action: Check the batch file status. The file may not be in the directory.

BPR—Batch job jobname still in progress

Explanation: System error. QMG attempted to start the batch job while another job was in progress.

User Action: Inform the system manager.

BPR—Error during send to QMG

Explanation: System error.

User Action: Inform the system manager.

BPR—Illegal error—severity code n

Explanation: System error.

User Action: Inform the system manager.

BPR—Incorrect Emit Status Block (ESB) address returned by spawned task

Explanation: System error.

User Action: Inform the system manager.

BPR—Log file close failure

Explanation: BPR failed to close the log file.

User Action: Check the log file status. The device may not be available, or the file may not be in the directory.

BPR—Log file open error

Explanation: BPR failed to open the log file.

User Action: Check the log file destination. The device may be write-locked.

BPR—Log file output error

Explanation: BPR failed to write to the log file.

User Action: Check the log file status. The device may not be available, or the file may not be in the directory.

BPR—Output request from incorrect virtual terminal

Explanation: System error.

User Action: Inform the system manager.

Chapter 9
The Indirect Command Processor

This chapter describes indirect command files and the Indirect Command Processor (Indirect). Also included are descriptions of the processor directives and symbols that control the execution of Indirect.

9.1 Indirect Command Files

Indirect command files can be used for many different things—from doing simple tasks to performing complex system-control and programming functions.

An indirect command file is a text file containing DCL command lines and special directives that allow you to control command file processing. The Indirect Command Processor (which usually runs under the task name AT.) reads the indirect command file, interprets the directives, and passes the DCL commands to DCL.

For example, an indirect command file could contain the following command lines:

```
.ENABLE SUBSTITUTION
.ASKS COMMAN Enter command name
HELP 'COMMAN'
```

With this file, Indirect processes the first two command lines and DCL executes the HELP command line.

To initiate an indirect command file, type in the file specification preceded by an at sign (@). For example:

```
$ @COMMANDS.CMD RET
```

The default file type for indirect command files is .CMD. Thus, the command line in the previous example could also be input as follows:

```
$ @COMMANDS RET
```

The name of the indirect command file can also be a logical name assignment that translates into a vaild Files Control Services (FCS) file specification. For example,if you have assigned the logical name TEST to the string DU1:[USER]COMMANDS.CMD, the command @TEST invokes the file COMMANDS.CMD.

If the catchall task (TDX) is installed on your system, you can give the command file a 3-character name and execute the file without using the at sign. For example:

```
$ ABC RET
```

Indirect searches your directory for a command file called ABC.CMD.

See Chapter 2 of this manual and the *Micro/RSX System Manager's Guide* for more information on the catchall task (TDX).

Indirect command files can also be nested. The maximum level of nesting is four unless this value has been changed by your system manager. A maximum level of four means that you can run one command file, which can run another file, which can run a third file, which can run a fourth file, which can run a fifth file.

For example, the following command file executes a DCL command line and then invokes another command file (COOKIE.CMD). When Indirect is finished with COOKIE.CMD, it returns to the first file, which executes more DCL commands.

```
SET TERMINAL/LOWER/SCOPE/WIDTH:80
@COOKIE
SHOW DEVICES
SHOW USERS
SHOW TIME
```

For DCL commands and for questions displayed by the .ASKx directives, the first character displayed is the DCL prompt. The default prompt is a dollar sign followed by a space ($).

9.2 The Indirect Command Processor

When processing an indirect command file, Indirect first reads the command file and interprets each command line either as a command to be passed directly to DCL or as a request for action by Indirect. The directives for Indirect are distinguished by a period (.) as their beginning character.

The Indirect directives allow you to perform the following functions:

- Define and assign values to logical, numeric, and string symbols (see Section 9.4 for more information on symbols)
- Substitute a symbols's value into any line of the command file
- Perform arithmetic
- Manipulate strings
- Display text on the user's terminal
- Ask questions of a user
- Control the sequence of execution of a command file
- Call subroutines
- Detect error conditions
- Test symbols and conditions
- Create and access data files
- Parse commands and data
- Enable or disable any of several operating modes
- Control time-based and parallel task execution
- Expand logical name assignments

These functions are described throughout Section 9.6, Description of Indirect Directives.

Two directives, .BEGIN and .END, allow you to block-structure the command file and create Begin-End blocks. Modular, block-structured command files are easier to debug and maintain. More importantly, Begin-End blocks isolate local symbol definitions as well as labels and thus conserve symbol table space.

When you define a symbol, Indirect creates an entry for the definition in an internal symbol table. Generally, symbol table entries retain their definitions under the following conditions:

- If defined locally, throughout the execution of the command file.
- If defined globally, throughout the execution of all levels of nested command files (a dollar sign ($) at the beginning of the symbol indicates a global symbol).

When defined within a Begin-End block, however, local symbols retain their definitions only throughout the execution of the commands within that block. The symbols are erased from the symbol table when Indirect encounters the .END directive at the end of the block.

One Indirect directive, .ENABLE GLOBAL (see Section 9.6.12), and a switch, /LO (see Section 9.5), allow the definition of some symbols as global to all file levels. If symbols are *not* global, each time Indirect enters a deeper level, it masks out of the symbol table all symbols defined by the previous level so that only the symbols defined in the current level are available for use by that level. When control returns to a previous level, the symbols defined in that level become available once again and the ones from the lower level(s) are lost.

When Indirect reaches the end of the highest-level indirect command file, it displays the message

```
$ @ <EOF>
```

and then exits. (The message is not displayed if the .DISABLE DISPLAY directive is in effect. See Sections 9.6.11 and 9.6.12.)

Indirect displays on the requesting terminal every DCL command line as it is executed. However, if Indirect is activated by @filename/NOCLI, the DCL command lines are displayed but not executed. (See Section 9.5 for information on the /[NO]CLI switch.)

A command file can also include comments. Comments can be placed at different locations in the file and require different preceding characters depending on how you want Indirect and DCL to treat them. Following are the three formats for comments:

;comment	Comments at beginning of line to be displayed by DCL
!comment	Comments after the start of a DCL command line
.;comment	Comments that will not be displayed

Indirect attaches the terminal while processing contiguous comment lines that begin with a semicolon. This allows you to type CTRL/O and suppress a lengthy comment. Output is resumed by typing another CTRL/O or is resumed at the next DCL command line or Indirect directive statement in the command file.

Note that no command or comment lines are displayed if .ENABLE QUIET is in effect (see Section 9.6.12).

When Indirect processes a .ENABLE QUIET statement, it forces a detachment (if detach mode is enable, which is the default) because it no longer needs the terminal for processing. Once quiet mode has been established, no attempts are made to reattach the terminal.

Any DCL command line issued by Indirect also causes an unconditional detachment. This action prevents a task, which may need the terminal, from suspending activity because the terminal is attached by Indirect.

A .DISABLE QUIET statement establishes terminal I/O but does not attempt to detach the terminal. See Section 9.6.12 for more information.

If you do not specify a file name in the initial command line, Indirect can construct the name of a default file to be opened. The default file is named INDINIxxx.CMD, where xxx either is null or is the 3-character task name under which Indirect is installed. Note, however, that this facility is usually disabled. To enable it, the value in the build file for the Indirect task must be changed.

If a specified command file cannot be found in the current directory, Indirect can also search for the file in another directory. However, to enable this facility, the value D$CUIC in the build file for the Indirect task must be changed to be nonzero. If the new value is 1, Indirect searches for the file in LB:[libuic]. If the new value is greater than 377(8), Indirect considers it to be the octal equivalent of the UIC (on LB:) to be searched. For example, if you issue the command @ABC.CMD but Indirect cannot find the file in the current directory, then, if the value of D$CUIC is set to [303,54], Indirect searches that directory for the file.

9.3 Summary of Indirect Directives

The Indirect directives described later in this chapter are listed here by category. A detailed description of each directive is given in alphabetical order in Section 9.6.

Category	Function
Label Definition	
.label:	Assigns a name to a line in the command file so that the line may be referenced elsewhere within the file by a .GOTO or .GOSUB directive.
Symbol Definition	
.ASK	Prompts for user input to define or redefine a logical symbol and assign the symbol a true or false value.
.ASKN	Prompts for user input to define or redefine a numeric symbol and assign the symbol a numeric value.
.ASKS	Prompts for user input to define or redefine a string symbol and assign the symbol a character string value.
.ERASE	Deletes all local or global symbol definitions or a single global symbol definition.
.SETT .SETF	Defines or redefines a logical symbol and assigns the symbol a true or false value.
.SETN	Defines or redefines a numeric symbol and assigns the symbol a numeric value.
.SETD .SETO	Redefines the radix of a numeric symbol.
.SETL	Defines or redefines a logical symbol and assigns the symbol a true or false value.
.SETS	Defines or redefines a string symbol and assigns the symbol a character string value.
.TRANSLATE	Expands a logical name translation into the special symbol <EXSTRI> .

Category	Function
File Access	
.CHAIN	Closes the current indirect command file and begins executing commands from another file.
.CLOSE	Closes a user data file.
.DATA	Specifies a single line of data to be output to a file.
.OPEN	Creates and opens an output data file (if the file exists, creates a new version and opens it).
.OPENA	Opens an existing data file and appends subsequent text to it (does not create a new version). Defaults to .OPEN if the file does not exist.
.OPENR	Opens a data file for reading with the .READ directive.
.PARSE	Parses (divides) strings into substrings.
.READ	Reads a line from a file into a specified string variable.
Logical Control	
.BEGIN	Marks the beginning of a Begin-End block.
.END	Marks the end of a Begin-End block.
.EXIT	Terminates processing of either Indirect or the current command file, returns control to the invoking terminal or to the previous Indirect file level, and optionally sets the value for the special symbol <EXSTAT> .
.GOSUB	Calls a subroutine within the command file.
.GOTO	Branches to a label within the command file.
/	Defines logical end-of-file. Terminates file processing and exits. This directive is equivalent to the .STOP directive. It is the only directive that does not begin with a period and does not consist of alphabetic characters.

Category	Function
.ONERR	Branches to a label upon detecting a specific Indirect error condition.
.RETURN	Effects an exit from a subroutine and returns to the line immediately following the subroutine call.
.STOP	Terminates indirect command file processing and optionally sets Indirect exit status. This directive is equivalent to the logical end-of-file (/) directive.
Logical Tests	
.IF	Determines whether or not a symbol satisfies a condition.
.IFACT .IFNACT	Determines whether or not a task is active.
.IFDF .IFNDF	Determines whether or not a symbol is defined.
.IFENABLED .IFDISABLED	Tests the .ENABLE or .DISABLE options.
.IFINS .IFNINS	Determines whether or not a task is installed in the system.
.IFLOA .IFNLOA	Determines whether or not a device driver is loaded.
.IFT .IFF	Determines whether a logical symbol is true or false.
.TEST	Tests the length of a string symbol or locates a substring.
.TESTDEVICE	Returns information about a device in the system.
.TESTFILE	Determines if a specified file exists and determines the physical device associated with a logical device name (performs device translation).
.TESTPARTITION	Returns information about a memory partition in the system.

Category	Function
Enable or Disable an Operating Mode	
.ENABLE .DISABLE	Enables or disables control of the following modes: Substitution (SUBSTITUTION) Time-out parameter (TIMEOUT) Lowercase-character processing (LOWERCASE) Terminal attachment (ATTACH, DETACH) Output of data to data files (DATA) File deletion (DELETE) Global symbols (GLOBAL) Symbol radix (DECIMAL) Command line echo (QUIET) Command display (TRACE) Field display (DISPLAY) Passing commands to CLI (CLI) Input truncation error suppression (TRUNCATE) Escape recognition (ESCAPE) Escape-sequence processing (ESCAPE-SEQ) Control-Z recognition (CONTROL-Z) Numeric overflow (OVERFLOW)
Increment or Decrement Numeric Symbols	
.DEC	Decrements the value of a numeric symbol by one.
.INC	Increments the value of a numeric symbol by one.
Execution Control	
.DELAY	Delays the execution of an indirect command file for a specified period of time.
.PAUSE	Temporarily suspends the execution of an indirect command file to allow user action.
.WAIT	Waits for a specified task to complete execution and sets the special symbol <EXSTAT> with the completed task's exit status.
.XQT	Initiates a task, passes a command line to it, and continues Indirect processing without waiting for the task to complete.

9.4 Symbols

Indirect allows you to define symbols. These symbols can then be tested or compared to control flow through the indirect command file. Their values may also be inserted into DCL commands, data records for data files, or comments to be displayed on the terminal.

Symbol names are ASCII strings from one through six characters in length. They must start with a letter (A through Z) or a dollar sign ($). The remaining characters must be alphanumeric or a dollar sign.

There are three symbol types:

- Logical
- Numeric
- String

A logical symbol has a value of either true or false.

A numeric symbol can have a numeric value in the range of 0 through 177777(8) (65535 decimal). The symbol can be defined to have either a decimal or octal radix. The radix is relevant only when the symbol is substituted (see Section 9.4.2).

A string symbol has as its value a string of ASCII characters, with a length of 0 through 132(10) characters.

A symbol's type (logical, numeric, or string) is defined by the first directive that assigns a value to the symbol. Assignment directives can assign the following:

- A true or false value to define a logical symbol (defined by .ASK, .SETL, .SETT, or .SETF)
- An octal or decimal number to define a numeric symbol (defined by .ASKN or .SETN)
- A character string to define a string symbol (defined by .ASKS, .READ, or .SETS)

9.4.1 Special Symbols

Indirect defines certain special symbols automatically. These symbols are dependent on specific system characteristics and the replies to queries given during command file execution. Special symbols can be compared, tested, or substituted and are of three types: logical, numeric, or string. All special symbols have a common format: angle brackets (< >) enclose the special symbol name.

Sections 9.4.1.1 through 9.4.3 give brief descriptions of the special logical, numeric, and string symbols, and discuss the use of numeric and string symbols and expressions. Section 9.4.4 explains reserved symbols, and Sections 9.4.5 and 9.4.5.1 discuss symbol-value substitution.

9.4.1.1 Special Logical Symbols

The special logical symbols are assigned a true or false value based on the following conditions:

Symbol	Value
<ALPHAN>	Set to true if last string entered in response to a .ASKS directive or tested with a .TEST directive contains only alphanumeric characters. An empty string also sets <ALPHAN> to true.
<ALTMOD>	Set to true if last question was answered with an ALTMODE or ESCAPE. Otherwise, <ALTMOD> is set to false.
<DEFAUL>	Set to true if the answer to last query was defaulted (the RETURN key was pressed once) or a time-out occurred.
<EOF>	Set to true if the last .READ or .ASKx directive resulted in reading past the end of the file. Otherwise, <EOF> is set to false.

Symbol	Value
<ERSEEN>	Set to true if any of the following conditions are true (<ERRNUM> , <EXSTAT> , and <FILERR> are described in Section 9.4.1.2): • <FILERR> is less than 0 (that is, if a negative error code was returned). • An exit status (<EXSTAT>) value worse than <WARNING> was returned • <EOF> is set to true • <ERRNUM> is not 0 • You used the command line .SETT <ERSEEN> The command line .SETF <ERSEEN> sets the following conditions: • <FILERR> is set to 0 • <EXSTAT> is set to 0 • <EOF> is set to false • <ERRNUM> is set to 0
<ESCAPE>	Set to true if last question was answered with an ALTMODE or ESCAPE. Otherwise, <ESCAPE> is set to false. <ESCAPE> is a read-only symbol.
<FALSE>	Logical constant used for comparisons with the .IF directive or as a default for the .ASK directive.
<IAS>	Set to true if the current operating system is IAS. Always false on a Micro/RSX system.
<LOCAL>	Set to true if the terminal from which Indirect is executing (TI:) is a local terminal. If the terminal is remote, <LOCAL> is set to false.
<MAPPED>	Set to true if the system on which Indirect is running is mapped; set to false if the system is unmapped. Always true on a Micro/RSX system.

Symbol	Value
<NUMBER>	Set to true if the last string entered in response to a .ASKS directive or tested with a .TEST directive, contains only alphanumeric characters. An empty string also sets <NUMBER> to true.
<OCTAL>	Set to true if the answer to the last .ASKN directive or the radix of the numeric symbol tested in the last .TEST directive is octal, or if the last string tested with a .TEST directive contained all numeric characters in the range 0 through 7.
<PRIVIL>	Set to true if the current user is privileged. Its value is determined from the flag contained in the terminal data base. The symbol is set when Indirect is started and remains unchanged during execution. The next time Indirect is started, <PRIVIL> is reset if a command to change the user's privilege (for example, DCL SET TERM /NOPRIV) was issued during the previous execution.
<RAD50>	Set to true if the last string entered in response to a .ASKS directive or tested with a .TEST directive contains only Radix–50 characters. Radix–50 characters are the uppercase alphanumeric characters plus period (.) and dollar sign ($). A blank is not a Radix–50 character in this context. An empty string also sets <RAD50> to true.
<RSX11D>	Always false on a Micro/RSX system.
<TIMOUT>	Set to true if time-out mode is enabled and the last .ASKx directive timed out waiting for a user response.
<TRUE>	Logical constant used for comparisons with the .IF directive or as a default for the .ASK directive.

9.4.1.2 Special Numeric Symbols

The special numeric symbols are assigned the following values:

Symbol	Value
<ERRCTL>	Controls the way in which Indirect processes errors. The symbol is treated as an eight-bit mask. For each class of error that a user's .ONERR target routine processes (see Section 9.6.21), the appropriate bit is set in the mask. If the bit is cleared, Indirect exits after printing the error information. If the eighth bit, which is the sign bit or 200(8), is set, Indirect does not print any information about the error. The initial default value for <ERRCTL> is 1, which implies that only class 1 errors can be handled with a .ONERR address and that error messages will be printed. To cover class 1 and 2 errors, the value for <ERRCTL> must be 3. **Note** If you attempt to trap errors other than default class 1, processing cannot continue in most cases. The error service routine is limited to returning a fatal error message and executing the .EXIT directive. The internal state of Indirect is indeterminate in all but class 1 error cases. If you receive an error that is not class 1, clean up what you are doing as much as possible and exit from Indirect. See Section 9.8 for a list of the error messages and their assigned class values.
<ERRNUM>	Assigned the class number of an error that Indirect has finished processing. This value can be used for processing specific error types with a .ONERR routine. See Section 9.8 for a list of the error messages and their assigned class values.

Symbol	Value
<ERRSEV>	Assigned the error severity mask associated with the error that Indirect has finished processing. This bit mask corresponds to the bit mask <ERRCTL> used to control the processing.
<EXSTAT>	Assigned the value of 0, 1, 2, 4, or 17, depending on the exit status from the last DCL command line executed or from the last ".WAIT taskname" directive, where taskname was activated by the .XQT directive. This special numeric symbol is modified at the completion of a synchronous DCL command line or at the completion of a .WAIT directive. The .EXIT directive can also modify <EXSTAT> . The value is returned from a task that has completed if the task exits with status. Otherwise, the value is returned from DCL. The values 0, 1, 2, 4, and 17 and their corresponding special symbols indicate the following:

0	<WARNIN>	Warning.
1	<SUCCES>	Success.
2	<ERROR>	Error.
4	<SEVERE>	Severe error.
17	<NOSTAT>	The task could not return exit status.

Symbol	Value
<FILERR>	Assigned the FCS-11 (I/O error or driver) or directive (DSW) status code resulting from a .TESTFILE, .OPENx, or .READ directive operation. <FILERR> contains the contents of offset F.ERR and F.ERR+1 from the File Descriptor Block (FDB) associated with the file. If F.ERR+1 (the high byte of the word) contains zero, F.ERR (the low byte of the word) contains an I/O error code. If F.ERR+1 contains -1, F.ERR contains a directive status code.

The following lists give the codes (in octal words) and their meanings.

I/O Error Codes (F.ERR+1, the high byte, contains 0):

Error Number		Meaning
Decimal	Octal	
-1	000377	Bad parameters
-2	000376	Invalid function code
-3	000375	Device not ready
-4	000374	Parity error on device
-5	000373	Hardware option not present
-6	000372	Illegal user buffer
-7	000371	Device not attached
-8	000370	Device already attached
-9	000367	Device not attachable
-10	000366	End-of-file detected
-11	000365	End-of-volume detected
-12	000364	Write attempted to locked unit
-13	000363	Data overrun
-14	000362	Send/receive failure

Error Number		Meaning
–15	000361	Request terminated
–16	000360	Privilege violation
–17	000357	Shareable resource in use
–18	000356	Illegal overlay request
–19	000355	Odd byte count (or virtual address)
–20	000354	Logical block number too large
–21	000353	Invalid UDC module number
–22	000352	UDC connect error
–23	000351	Caller's nodes exhausted
–24	000350	Device full
–25	000347	Index file full
–26	000346	No such file
–27	000345	Locked from read/write access
–28	000344	File header full
–29	000343	Accessed for write
–30	000342	File header checksum failure
–31	000341	Attribute control list format error
–32	000340	File processor device read error
–33	000337	File processor device write error
–34	000336	File already accessed on LUN
–35	000335	File ID, file number check
–36	000334	File ID, sequence number check
–37	000333	No file accessed on LUN
–38	000332	File was not properly closed

Error Number		Meaning
–39	000331	No buffer space available for file
–40	000330	Illegal record size
–41	000327	File exceeds space allocated, no blocks
–42	000326	Illegal operation on File Descriptor Block
–43	000325	Bad record type
–44	000324	Illegal record-access bits set
–45	000323	Illegal record-attribute bits set
–46	000322	Illegal record number—too large
–47	000321	Internal consistency error
–48	000320	Rename—two different devices
–49	000317	Rename—a new file name already in use
–50	000316	Bad directory file
–51	000315	Cannot rename old file system
–52	000314	Bad directory syntax
–53	000313	File already open
–54	000312	Bad file name
–55	000311	Bad device name
–56	000310	Bad block on device
–57	000307	Enter—duplicate entry in directory
–58	000306	Not enough stack space (FCS or FCP)
–59	000305	Fatal hardware error on device
–60	000304	File ID was not specified

Error Number		Meaning
–61	000303	Illegal sequential operation
–62	000302	End-of-tape detected
–63	000301	Bad version number
–64	000300	Bad file header
–65	000277	Device off line
–66	000276	Block check, CRC, or framing error
–67	000275	Device on line
–68	000274	No such node
–69	000273	Path lost to partner
–70	000272	Bad logical buffer
–71	000271	Too many outstanding messages
–72	000270	No dynamic space available
–73	000267	Connection rejected by user
–74	000266	Connection rejected by network
–75	000265	File expiration date not reached
–76	000264	Bad tape format
–77	000263	Not ANSI "D" format byte count
–78	000262	No data available
–79	000261	Task not linked to specified ICS /ICR interrupts
–80	000260	Specified task not installed (.NST)
–80	000260	No AST specified in connect (.AST)
–81	000257	Device off line when off-line request was issued

Error Number		Meaning
–82	000256	Invalid escape sequence
–83	000255	Partial escape sequence
–84	000254	Allocation failure
–85	000253	Unlock error
–86	000252	Write check failure
–87	000251	Task not triggered
–88	000250	Transfer rejected by receiving CPU
–89	000247	Event flag already specified
–90	000246	Disk quota exceeded
–91	000245	Inconsistent qualifier usage
–92	000244	Circuit reset during operation
–93	000243	Too many links to task
–94	000242	Not a network task
–95	000241	Time-out on request
–96	000240	Connection rejected
–97	000237	Unknown name
–98	000236	Unable to size device
–99	000235	Media inserted incorrectly
–100	000234	Spindown ignored

Directive Status Codes (F.ERR+1, the high byte, contains –1):

Error Number		Meaning
Decimal	Octal	
–1	177777	Insufficient dynamic storage
–2	177776	Specified task not installed
–3	177775	Partition too small for task
–4	177774	Insufficient dynamic storage for send
–5	177773	Unassigned LUN
–6	177772	Device handler not resident
–7	177771	Task not active
–8	177770	Directive inconsistent with task state
–9	177767	Task already fixed/unfixed
–10	177766	Issuing task not checkpointable
–11	177765	Task is checkpointable
–15	177761	Receive buffer is too small
–16	177760	Privilege violation
–17	177757	Resource in use
–18	177756	No swap space available
–19	177755	Illegal vector specified
–20	177754	Invalid table number
–21	177753	Logical name not found
–80	177660	Directive issued/not issued from AST

Error Number		Meaning
-81	177657	Illegal mapping specified
-83	177655	Window has I/O in progress
-84	177654	Alignment error
-85	177653	Address window allocation overflow
-86	177652	Invalid region ID
-87	177651	Invalid address window ID
-88	177650	Invalid TI parameter
-89	177647	Invalid send buffer size (greater than 255.)
-90	177646	LUN locked in use
-91	177645	Invalid UIC
-92	177644	Invalid device or unit
-93	177643	Invalid time parameters
-94	177642	Partition/region not in system
-95	177641	Invalid priority (greater than 250.)
-96	177640	Invalid LUN
-97	177637	Invalid event flag (greater than 64.)
-98	177636	Part of DPB out of user's space
-99	177635	DIC or DPB size invalid

See the *RSX-11M/M-PLUS and Micro/RSX I/O Operations Manual* and the *Micro/RSX I/O Drivers Reference Manual* for more information.

<FORATT> Assigned the octal value of the file attributes that were used to open the data files.

<MEMSIZ> Assigned the value of the current system memory size in K words (K is 1024(decimal)).

Symbol	Value
<SPACE>	Assigned the number, in octal, of free bytes in the internal symbol table for Indirect. The number does not reflect the amount of space that could be gained by the automatic extension of the Indirect task.
<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. The symbol is also set when a command file is invoked (<STRLEN> contains the octal number of variables used in the command line) and as the result of a .PARSE statement (<STRLEN> contains the octal number of substrings produced by the directive).
<SYMTYP>	Assigned the numeric code for the type of symbol tested with a .TEST directive. The symbol types have the following code numbers: Logical — 0 Numeric — 2 String — 4
<SYSTEM>	Assigned an octal number to represent the operating system on which Indirect is running. For a Micro/RSX system, the value is 6.
<SYUNIT>	Assigned the unit number of the user's default device (SY:).
<TICLPP>	Assigned the current page length setting for the terminal. When you first invoke Indirect, it attempts to determine the length. If the information is not available, <TICLPP> defaults to 24(decimal).
<TICWID>	Assigned the current page width setting for the terminal. When you first invoke Indirect, it attempts to determine the width. If the information is not available, <TICWID> defaults to 80(decimal).

Symbol	Value
<TISPED>	Assigned the baud rate for transmitting characters from the host system to the terminal. When you first invoke Indirect, it attempts to determine the baud rate. The baud rate information is useful for determining the quality and quantity of information to be transmitted. The following list gives the octal value that corresponds to the baud rates:

1	0	13	1200
2	50	14	1800
3	75	15	2000
4	100	16	2400
5	110	17	3600
6	134	20	4800
7	150	21	7200
10	200	22	9600
11	300	23	EXTA
12	600	24	EXTB

<TITYPE>	Assigned the terminal type of the terminal from which Indirect is running. If the terminal type is changed from within an indirect command file, <TITYPE> is set to the latest terminal type. If Indirect cannot determine the terminal type, <TITYPE> is set to zero (0). The following list gives the octal value that corresponds to the terminal types:

0 Unknown	11 VT52	23 LA38
1 ASR33	12 VT55	24 VT101
2 KSR33	13 VT61	25 VT102
3 ASR35	14 LA180S	26 VT105
4 LA30S	15 VT100	27 VT125
5 LA30P	16 LA120	30 VT131
6 LA36	20 LA12	31 VT132

Symbol	Value		
	7 VT05	21 LA100	35 PC3xx-series
	10 VT50	22 LA34	36 VT200-series
	See the *Micro/RSX I/O Drivers Reference Manual* (the chapter on the full-duplex terminal driver) for more information.		

9.4.1.3 Special String Symbols

The special string symbols are assigned the following string values:

Symbol	Value	
<ACCOUN>	Assigned certain accounting information from a user's accounting block (UAB). If Resource Accounting is not running on the system, the fields of <ACCOUN> are null. The information is in the following format (note the trailing comma):	
	username,sessionid,accountnumber,CPU,DIR,QIO,TAS, activetasks,	
	username	The first 14(decimal) characters of the user name (as it appears in the system account file) followed by the first initial.
	sessionid	The 3-letter session-ID code followed by the unique login number.
	accountnumber	The user's account number as it appears in the system account file.
	CPU	The number of CPU ticks used since login.
	DIR	The number of system directives issued since login.
	QIO	The number of QIO directives issued since login.

Symbol	Value
	TAS — The number of tasks run since login.
	activetasks — The current number of the user's active tasks.
	The individual fields can be isolated with the .PARSE directive: `.PARSE <ACCOUN> "," NAME SID ACNT CPU DIR QIO TAS ACT JUNK`
	Note that because double-precision arithmetic is not available in Indirect, the numeric <ACCOUN> parameters cannot be converted to numeric form and manipulated in arithmetic expressions.
<CLI>	Assigned the acronym (3 through 6 letters) of the current command line interpreter (for example, DCL).
<CONFIG>	Contains the parameter defaults specified when the current Indirect task was built. If you have the Advanced Programmer's Kit, see the module INDCFG in the system procedure library LB:[1,2]INDSYS.CLB on the system disk for more details.
<DATE>	Assigned the current date; format is dd-mmm-yy.

Symbol	Value
<DIRECT>	Contains a user's current default directory string; format is [name]. The contents of <DIRECT> are different depending on the directory mode you are in and the kind of directory you are using. If you are not in named directory mode ("nonamed" mode), <DIRECT> contains a null directory string ([]). If you are in named directory mode and using a named directory, <DIRECT> contains your default directory string in the form ddddddddd. If you are in named directory and using a numeric directory, <DIRECT> contains your default directory string in the form [gggmmm]. If <DIRECT> contains the null string ([]), use the special symbol <UIC> for the location of your current default directory. If <DIRECT> contains a directory string, use it as the current default directory location.
<EXSTRI>	When Indirect is first initiated, contains build-time information about the Indirect task. The information includes the version number of the task and the time the task was built. Afterwards, it can contain such information as the string results from a more deeply nested indirect command file or the results of a .TESTDEVICE statement. The results are sent to the calling command file. This symbol can be redefined with a .SETS <EXSTRI> xxxx command.

Symbol	Value
<FILATR>	Contains eight fields of file-attribute information obtained from offsets for the File Descriptor Block (FDB). The information is from the FDB used in the last .OPENx operation and is in the following format (note the trailing comma): rtyp,ratt,rsiz,hibk,efbk,ffby,racc,rctl, The attributes are as follows: F.RTYP Record type (byte, octal). Set as follows to indicate the type of records for the file: 1—fixed-length records (R.FIX) 2—variable-length records (R.VAR) 3—sequenced records (R.SEQ) F.RATT Record attribute (byte, octal). Bits 0 through 3 are set as follows to indicate record attributes: Bit 0 If 1, first byte of record contains a FORTRAN carriage control character (FD.FTN); otherwise, 0. Bit 1 If 1, for a carriage control device, a line feed is to occur before the line is printed and a carriage return is to occur after the line is printed (FD.CR); otherwise, 0. Bit 2 If 1, indicates print file format (FD.PRN); FCS allows this attribute but does not interpret the format word; otherwise, 0. Bit 3 If 1, the records cannot cross block boundaries (FD.BLK); otherwise, 0.

Symbol	Value	
	F.RSIZ	Record size (word, decimal). Contains the size of fixed-length records or indicates the size of the largest record that currently exists in a file of variable-length records.
	F.HIBK	Highest virtual block number allocated (double word with F.EFBK, decimal).
	F.EFBK	End-of-file block number (double word with F.HIBK, decimal).
	F.FFBY	First free byte in the last block or the maximum block size for magnetic tape (word, octal).
	F.RACC	Record access (byte, octal). Bits 0 through 3 define as follows the record access modes: Bit 0 — If 1, READ$/WRITE$ mode (FD.RWM); if 0, GET$/PUT$ mode. Bit 1 — If 1, random access mode (FD.RAN) for GET$/PUT$ record I/O; if 0, sequential access mode. Bit 2 — If 1, locate mode (FD.PLC) for GET$ /PUT$ record I/O; if 0, move mode. Bit 3 — If 1, PUT$ operation in sequential mode does not truncate the file (FD.INS); if 0, PUT$ operation in sequential mode truncates the file.

Symbol	Value		
	F.RCTL	Device characteristics (byte, octal). Bits 0 through 5 define as follows the characteristics of the device associated with the file:	
		Bit 0	If 1, record-oriented device (FD.REC); if 0, block-structured device.
		Bit 1	If 1, carriage control device (FD.CCL); otherwise, 0.
		Bit 2	If 1, teleprinter device (FD.TTY); otherwise, 0.
		Bit 3	If 1, directory device (FD.DIR); otherwise, 0.
		Bit 4	If 1, single-directory device (FD> SDI; an MFD is used, but no directories are present).
		Bit 5	If 1, block-structured device inherently sequential in nature (FD.SQD), such as a magnetic tape. Record-oriented devices are assumed to be sequential in nature, so this bit is not set for them.
	If no file is currently open, a fatal error occurs.		
<FILSPC>	Assigned the specification for the file referred to with the last .OPEN, .OPENA, .OPENR, or TESTFILE directive operation, or in the last specification for a nested command file.		
<FMASK>	Contains octal values representing answers to some of the system generation questions. If you have the Advanced Programmer's Kit, refer to the module INDSFN in the system procedure library LB:[1,2]INDSYS.CLB on the system disk for an explanation of the values.		
<LIBUIC>	Assigned the UIC of the current nonprivileged task library; format is [g,m], where g is the group number of the UIC and m is the member number of the UIC (leading zeros are not included).		

Symbol	Value
<LOGDEV>	Assigned the device name and unit number of the user's login account.
<LOGUIC>	Assigned the login UIC of the current user; format is [ggg,mmm].
<NETUIC>	If the system has DECnet, assigned the UIC in which DECnet-related tasks are stored on the system volume; format is [ggg,mmm]. <NETUIC> is used with <SYSUIC> and <LIBUIC> to separate the components of the system.
<NETNOD>	Assigned the DECnet node name of the system. If the system is not on the DECnet network, <NETNOD> is assigned MICRO.
<SYDISK>	Assigned the device mnemonic (two letters) of the user's default device (SY:); format is dd (for example, DU).
<SYSDEV>	Assigned the physical name of the system disk. The device name is in the form ddn (for example, DU0).
<SYSID>	Assigned the operating system's baselevel number.
<SYSUIC>	Assigned the system UIC; format is [ggg,mmm].
<SYTYP>	Contains a string consisting of up to 12 ASCII characters that identifies the system; only three identifications are valid: "Micro/RSX," "RSX-11M-PLUS," and "RSX-11M."
<TIME>	Assigned the current time; format is hh:mm:ss.

Symbol	Value
<UIC>	Assigned the current UIC; format is [ggg,mmm]. The contents of <UIC> are different depending on the directory mode you are in and the kind of directory you are using. If you are not in named directory mode ("nonamed" mode), <UIC> contains your default UIC in the form ggg,mmm. If you are in named directory mode and using a named directory, <UIC> contains your protection UIC in the form ggg,mmm. In this case, there is no default. If you are in named directory mode and using a numeric directory, <UIC> contains the default UIC corresponding to the default directory string in <DIRECT> . The default UIC is in the form ggg,mmm. <UIC> follows the numbered default directory string in named directory mode so that all command files and tasks created previous to Version 3.0 will still work from named mode in a numeric directory. If you need to obtain the protection UIC from within the indirect command file, use one of the following procedures: • If you are a privileged user, use the <UIC> symbol. • If you are a nonprivileged user and your system has Resource Accounting, use the <LOGUIC> symbol.
<VERSN>	Contains a string consisting of up to four ASCII characters that identifies the version number of the system; format is n.n (for example, 3.0).

9.4.2 Numeric Symbols and Expressions

A numeric symbol is a string of digits representing a value in the range of 0 through 177777(octal) (0 through 65535(decimal), if immediately followed by a period, or if decimal mode has been enabled). If an arithmetic operation yields a result outside of this range, or one that crosses the boundaries, a fatal error occurs and the following message is displayed (unless turned off by .ENABLE OVERFLOW):

```
AT. -- Numeric under- or overflow
```

A numeric symbol or constant may be combined with another numeric symbol or constant by a logical or arithmetic operator to form a numeric expression. Arithmetic operators are used to add (+), subtract (-), multiply (*), and divide (/). Logical operators are the inclusive OR (!), logical AND (&), and NOT (#). Embedded spaces and tabs are not permitted in front of operators. If a space precedes an operator, particularly the plus sign (+), the operator will not function correctly.

Numeric expressions are evaluated from left to right unless parentheses are used to form subexpressions, which are evaluated first. For example, the directive statements

```
.SETN N1 2
.SETN N2 3
.SETN N3 N1+N2*4
```

assign numeric symbol N3 the value 24(octal), whereas the directive statements

```
.SETN N1 2
.SETN N2 3
.SETN N3 N1+(N2*4)
```

assign numeric symbol N3 the value 16(octal).

Numeric expressions are permitted as second operands in numeric .IF and .SETN directives. They are also permitted as range and default arguments in .ASKN and .ASKS directives. The directives .EXIT and .STOP allow numeric expressions to represent exit status.

Indirect associates a radix, either octal or decimal, with each numeric symbol. The radix of a numeric symbol changes each time the symbol is assigned a new value. If you use a numeric expression to assign a new value to a symbol and all operands in the expression are octal, then the symbol is set to octal. If any operand in the expression is decimal, the symbol is set to decimal. For example:

```
.SETN N1 2                 ! N1 is octal
.SETN N2 3.                ! N2 is decimal
.SETN N3 N1+3              ! N3 is octal
.SETN N3 N1+3.             ! N3 is decimal
.SETN N3 N1+N2             ! N3 is decimal
```

You can also assign a new value to a symbol with the .ASKN directive. See Section 9.6.3 for more information.

The .SETO and .SETD directives allow you to change the radix of a numeric symbol without changing the value of the symbol. For example:

```
.SETN N1 10.                  ! N1 = 10 decimal
.SETO N1                      ! N1 = 12 octal
```

See Section 9.6.31 for more information on .SETO and .SETD.

The radix of a numeric symbol does not affect arithmetic operations or comparisons. The radix is important only when substituting a numeric symbol into a string. If the radix of the symbol is octal, the value of the symbol is substituted into the string as an octal number. If the radix is decimal, the value is substituted as a decimal number. For example:

```
SETN N1 10.                   ! N1 = 10 decimal
; N1 = 'N1'                   ! Displayed as ; N1 = 10
.SETO N1                      ! Make N1 octal
; N1 = 'N1'                   ! Displayed as ; N1 = 12
```

If you substitute a numeric symbol into a string and the substituted number is decimal, a period (.) following the symbol name causes a trailing period to be included in the string (following the substituted number). For example:

```
.SETN N1 10.                  ! N1 = decimal
; N1 = 'N1'                   ! Displayed as ; N1 = 10
; N1 = 'N1.'                  ! Displayed as ; N1 = 10.
.SETO N1                      ! Make N1 octal
; N1 + 'N1.'                  ! Displayed as ; N1 = 12
```

You can also force a numeric symbol to be substituted as an octal or decimal number by using a substitution format control string. For example:

```
.SETN N1 10.                  ! N1 = 10 decimal
; N1 = 'N1%D'                 ! Displayed as ; N1 = 10
; N1 = 'N1%O'                 ! Displayed as ; N1 = 12
```

See Section 9.4.5.1 for more information on substitution format control strings.

9.4.3 String Symbols, Substrings, and Expressions

A string constant is a string of any printable characters enclosed by quotation marks or a pound sign (#). (if you begin a string with one of these delimiters, you must end it with the same delimiter.) Using pound signs is helpful when you want to include quotation marks in the string. Empty strings are also permitted. The number of characters cannot exceed 132(decimal). For example:

```
"ABCDEF"
#HITHERE#
#HI"THERE"#
   ""
   ##
```

String symbols may have the value of any string constant. The value is assigned by a .SETS or .ASKS directive. For example, the directive statements

```
.SETS   S1   "ABCDEF"
.SETS   S2    S1
```

assign string symbol S2 the value of string symbol S1 (that is, ABCDEF).

A substring facilitates the extraction of a segment from the value of a string symbol. You can use substrings only in second operands of .SETS and .IF directives. For example, the directive statements

```
.SETS   S1   "ABCDEF"
.SETS  S2    S1[1:3]
```

assign string symbol S2 the value of string symbol S1 beginning at character one and ending at character three (that is, ABC).

You can also use the syntax [n:*] to extract the characters from position n to the end of the string. For example, the directive statements

```
.SETS  S1  "ABCDEF:
.SETS  S2  S1[3:*]
```

assign string symbol S2 the value CDEF.

You can combine a string constant, symbol, or substring with another string constant, symbol, or substring by the string concatenation operator (+) to form a string expression.

String expressions are permitted as second operands in .SETS and .IF directives where the first operand is a string symbol. For example, the directive statements

```
.SETS   S1   "A"
.SETS   S2   "CDEF"
.SETS   S3   S1+"B"+S2[1:3]
```

assign string symbol S3 the value of the concatenation of string symbol S1, string constant "B," and the first three characters of string symbol S2 (that is, ABCDE).

9.4.4 Reserved Symbols

Parameters for a command file can be passed to Indirect for processing. The parameters are stored in the following reserved local symbols:

P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, COMMAN

The symbol COMMAN contains everything in the issuing command line, including the specification for the command file.

The symbols P0 through P9 contain individual elements of the command line. The elements are delimited by a single space or tab character in between each one. (This is not true for a .CHAIN command line, however.) Two delimiting characters between elements represent a null parameter. (See the description of the .PARSE directive in Section 9.6.25 for an example of this behavior.)

With the .GOSUB directive (see Section 9.6.16), any parameters to the right of the label and to the left of a comment are transferred to the symbol COMMAN. The value of COMMAN can then be parsed to obtain formal call parameters.

9.4.5 Symbol Value Substitution

Substitution can occur in any line. Indirect can use the values assigned to logical, numeric, string, or special symbols by replacing a normal parameter (for example, a device unit) with the symbol name enclosed in apostrophes (for example, 'DEVICE'). When a previous directive has enabled substitution mode (.ENABLE SUBSTITUTION), Indirect replaces the symbol name enclosed in apostrophes with the value assigned to the symbol.

When Indirect encounters an apostrophe, it treats the subsequent text, up to a second apostrophe, as a symbol name. Indirect then searches the table of symbols for the corresponding symbol and substitutes the value of the symbol in place of the symbol name and surrounding apostrophes in the command line.

For example, the first three lines in the following example appear in an indirect command file. When Indirect executes these lines, it displays the last two lines at the entering terminal.

```
.ENABLE SUBSTITUTION
.ASKS DEVICE Device to mount?
MOUNT 'DEVICE'
```

```
$ * Device to mount? [S]: DU1: RET
$ MOUNT DU1:
```

DU1: was entered in response to the displayed question. This reply assigned the string value DU1: to string symbol DEVICE. Then, when Indirect read

```
MOUNT 'DEVICE'
```

it substituted for 'DEVICE' the value assigned to DEVICE (that is, DU1:). If substitution mode was not enabled, Indirect would simply have passed the line to DCL as it appeared in the command file (that is, MOUNT 'DEVICE').

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 contiguous apostrophes (''). If substitution mode is enabled, Indirect displays the command file line

```
;DON''T PANIC
```

as

```
;DON'T PANIC
```

9.4.5.1 Substitution Format Control

The conversion of numeric values to strings and the placement of string and logical values in a substitution operation can be controlled with a format control string. The control string is in the following form:

'symbol%controlstring'

The control string begins with the percent sign (%) and ends with the second of the two apostrophes that denote the substitution operation. The control string consists of one or more of the following characters:

C	Compress leading, embedded, and trailing blanks, and remove embedded nulls (leave one space between characters, but strip all leading and trailing spaces).
D	Force the conversion of a numeric symbol to decimal.
O	Force the conversion of a numeric symbol to octal.
S	Perform signed conversion for a numeric symbol.
M	Perform magnitude conversion for a numeric symbol.
Z	Return leading zeros for a positive numeric value.
Rn	Right-justify the resulting string, truncating to 'n' decimal characters if necessary.
Ln	Left-justify the resulting string, truncating to 'n' decimal characters if necessary.
X	Convert the variable to Radix–50 characters.
V	If the symbol being substituted is numeric, convert the low byte to its equivalent ASCII character and substitute it. If the symbol being substituted is a string, convert the first character to its octal representation and substitute it.

As an example, the following command file shows various control strings being used and the results of using the control strings:

```
.ENABLE SUBSTITUTION
.SETS STRING "  A B  CD  "
; STRING = 'STRING%C'     ! Compress spaces
.SETS STRING "ABCD"
; STRING = 'STRING%R5'    ! Right-justify string,
;                         ! truncating to 5 characters
; STRING = 'STRING%3'     ! Right-justify string,
;                         ! truncating to 3 characters
.SETN NUMBER 10.
; NUMBER = 'NUMBER%D'     ! Convert numeric symbol to decimal
; NUMBER = 'NUMBER%O'     ! Convert numeric symbol to octal
; NUMBER = 'NUMBER%ZO'    ! Return leading zeros for positive
;                         ! numeric value, convert to octal
; NUMBER = 'NUMBER%ZOR4'  ! Return leading zeros; convert to
;                         ! octal; right-justify,
;                         ! truncating to 4 characters
```

When the command file is executed, Indirect displays the text as follows:

```
$ @CONTROL [RET]
$ ; STRING = A B CD     ! Compress spaces
$ ; STRING =  ABCD      ! Right-justify string, truncating to 5
$ ;                     ! characters
$ ; STRING = ABC        ! Right-justify string, truncating to 3
$ ;                     ! characters
$ ; NUMBER = 10         ! Convert numeric symbol to decimal
$ ; NUMBER = 12         ! Convert numeric symbol to octal
$ ; NUMBER = 000012     ! Return leading zeros for positive
$ ;                     ! numeric value, convert to octal
$ ; NUMBER = 0012       ! Return leading zeros; convert to
$ ;                     ! octal; right-justify, truncating to 4
$ ;                     ! characters
```

Indirect does not perform a consistency check on the control string. If you specify conflicting format characters, Indirect uses the last one specified.

9.5 Switches

Indirect accepts the following switches: /TR, /CLI, /LB, /LO, and /DE.

Switch	Function
/[NO]TR	Displays a trace of the indirect command file on the terminal from which the file is being run. This function is useful for debugging an indirect command file. Each command line, including Indirect directive statements, is displayed. As each command line is processed, a number representing the nesting depth of the command file is displayed, followed by an exclamation point and the command line. If the command line causes some action to occur, the next displayed line indicates the action; usually, this line consists of the DCL commands issued as a result of the previous directive. The default is /NOTR.
/[NO]CLI	Passes commands not processed by Indirect to the default command line interpreter (DCL) for your system. The default is /CLI.
/LB	Indicates that the specified file is a universal library of command procedures and that the specified module is the procedure to be executed. When command procedures, which are indirect command files, are inserted into a universal library with the DCL LIBRARY/INSERT command, you can then reference them with /LB:module. Command libraries are built by creating a universal library and inserting command files into it. You can then reference the procedures in the library with the following command line: `@commandlibrary/LB:module` The default file type for a command library is .CLB. If you do not specify a module (@commandlibrary/LB), Indirect attempts to locate a module called .MAIN..

Switch	Function
	If you do not specify a library name (@/LB:module), the following actions occur:

- If the command is issued from the terminal or from a file that is not in the library, Indirect ignores the /LB switch and treats the command line as though you had used @module.CMD. Note that if the command is issued from a command file, the default device and directory of the specified module are the same as those for the current file, not necessarily the same as those for the terminal.
- If the command was issued from within a library, the specified module is searched for in the current library.

These default actions for an unspecified library allow a collection of procedures to be developed in a given directory with the @/LB:module or .CHAIN /LB:module commands. When the procedures are then placed in a library, no source changes are required.

Example

The command file PARAM.CMD contains parameter definitions for the .SETN directive and the command file SYSPRC.CMD contains system-specific procedures. The following command lines create the command library and enter the command files into it:

```
$ LIBRARY/CREATE/UNIVERSAL:CMD SYSTART.CLB
$ LIBRARY/INSERT SYSTART.CLB PARAM,SYSPRC
```

You can then use the following command lines to reference the command library modules:

```
$ @SYSTART/LB:PARAM      !Define global symbols
$ @SYSTART/LB:SYSPRC     !Run init procedure
```

If you have the Advanced Programmer's Kit, DIGITAL supplies a library of command procedures on the system disk. The library is LB:[1,2]INDSYS.CLB and it contains the following procedures:

INDCFG	Displays the current build parameters for the running Indirect task

Switch	Function
	INDDMP Dumps to the terminal the contents of the Indirect symbol table
	INDPRF A sample procedure to fully parse file-name strings
	INDSFN Returns system-configuration information
	INDVFY Displays the values of all of the special symbols
	QIOERR Returns a string expansion of the <FILERR> error codes
	.INDEX Displays an index of the procedures in the library
	The following command line shows the format for invoking a command procedure in the library: `@LB:[1,2]INDSYS/LB:procedurename`
	Before you attempt to access a command procedure, make sure that INDSYS.CLB is in LB:[1,2]. If it is not in this directory, your system manager must copy the library from the Advanced Programmer's Kit.
/[NO]LO	Indicates that when a new command file is executing, it can have access to the local symbols created by its calling command file and that any local symbols created by the new command file will be defined as local symbols for the calling command file. The default is /NOLO.
/[NO]DE	Indicates that the indirect command file is to be deleted when its processing is complete unless a logical end-of-file (/) or .STOP directive is encountered before the end of the file. The default is /NODE.

You may use any combination of the switches in the following command line:

```
@filespec/switch(es)
```

or in the following directive statement:

```
.CHAIN filespec/switch(es).
```

Except for /LB and /LO, the switches you specify in the command line that initiates Indirect processing are used as defaults when executing those commands.

9.6 Description of Indirect Directives

Directives must be separated from their arguments and from DCL-specific commands by at least one space. Unless you are using the .IF directive, only one directive per command line is allowed.

You can insert any number of blanks and horizontal tabs in three places in a command line:

1. At the start of the command line
2. Immediately following the colon (:) of a label
3. At the end of the command line

This allows you to format the command files for readability. The recommended procedure is to begin labels in the first column and everything else in the ninth column (after one horizontal tab).

An important exception is the lines processed between .ENABLE and .DISABLE DATA directives; no blanks or tabs are removed from these lines. For example:

```
        .IFT Z .GOTO 10
    .
    .
    .
.10:    .OPEN DATFIL
        .DATA XXXXX
.ENABLE DATA
This is data
that goes into
the data file.
.DISABLE DATA
        .GOTO 20
```

Note that the .DISABLE DATA statement must begin in the first column or Indirect will place it in the data file. You can also use the .CLOSE directive in place of .DISABLE DATA. It too must begin in the first column.

9.6.1 Define a Label (.label)

Labels always appear at the beginning of the line. They may be on a line with additional directives and/or a DCL command, on a line with a comment, or on a line by themselves. When control passes to a line with a label, the line is processed from the first character after the colon.

Commands do not have to be separated from the label by a space. Only one label is permitted per line. Labels are one through six characters in length and must be preceded by a period and terminated with a colon. A label may contain only alphanumeric characters and/or dollar signs ($).

It is also possible to define a label as a direct-access label; once the label is found, its position in the command file is saved. This allows subsequent jumps to frequently called labels or subroutines to be effected quickly. The first statement processed after a jump to a direct-access label is the one on the next line.

If you define more than the maximum number of labels allowed, the subsequent direct-access labels replace the earliest, and so on. The smaller the number of direct-access labels, the larger the amount of free space in the symbol table.

If you have a large command file that branches from a line to a label before that line, using direct-access labels can result in a substantial saving of processing time. Normally, Indirect searches for the label in every line below the one where the branch occurred. If the label is not found, Indirect wraps around to the top of the file to continue the search. With direct-access labels, however, Indirect can go immediately to the label.

To declare a label for direct access, leave the line following the colon blank.

Example

```
.100:   .ASK A Do you want to continue
        .IFT A .GOSUB 200
          .
          .
          .
.200:
        .;THIS IS THE START OF A SUBROUTINE
          .
          .
          .
        .RETURN
```

In this example, .200: is a direct-access label while .100: is not.

The target label of a .GOTO branch from within a Begin-End block must be contained in that block because the .GOTO directive cannot branch into another block. The target label of a .ONERR directive must also be contained on the same Begin-End level. The target label of a .GOSUB call from within a Begin-End block, however, can be outside the current block because program control returns to the block from which the .GOSUB call was made. For more information, see the descriptions of the .BEGIN, .GOSUB, .GOTO, and .ONERR directives (Sections 9.6.5, 9.6.16, 9.6.17, and 9.6.21, respectively.)

9.6.2 Ask a Question and Wait for a Reply (.ASK)

The .ASK directive diplays a question on the terminal, waits for a reply, and sets a specified logical symbol to the value of true or false, depending on the reply. If the symbol has not already been defined, Indirect makes an entry in the symbol table. If the symbol has been defined, Indirect resets its value (true or false) in accordance with the reply. Indirect exits with a fatal error if the symbol was previously defined as a string or numeric symbol.

Formats (brackets are required syntax)

```
.ASK   ssssss txt-strng
.ASK   [default:timeout] ssssss txt-strng
.ASK   [:timeout] ssssss txt-strng
```

Parameters

ssssss

The 1- through 6-character symbol to be assigned a true or false value.

txt-strng

The question or prompt that Indirect displays.

default

The default response; used if the question is answered with an empty line (null) or if time-out occurs. The default can be <TRUE> or <FALSE> or another logical variable.

timeout

The time-out count. Indirect waits this long for a response, then applies the default answer. The format for time-out is nnu, where nn is the decimal number of time units to wait and u is T (ticks), S (seconds), M (minutes), or H (hours). The time-out count is valid only if time-out mode is enabled (.ENABLE TIMEOUT; see Section 9.6.12).

The entire .ASK statement must fit on one command line.

Note that if you omit the default value but specify a timeout count, the colon is required for positional identification.

When executing a .ASK directive, Indirect displays (unless .DISABLE DISPLAY is in effect) txt-strng prefixed by an asterisk (*) and suffixed with "?[Y/N]:". Indirect recognizes five answers:

Y RET	Set symbol ssssss to true.
N RET	Set symbol ssssss to false.
RET	Set symbol to false or to user-specified default value. The RET symbol indicates the RETURN key.
ESC	Set symbol ssssss to true and set the special logical symbol <ESCAPE> to true only if escape recognition has been enabled. The ESC symbol indicates the ESCAPE or ALTMODE key.
CTRL/Z	If Control-Z mode is enabled, set <EOF> to true and proceed, else exit immediately.

Example

The directive statement

```
.ASK PRINT Do you want to print the file
```

displays

```
$ * Do you want to print the file? [Y/N]:
```

on the terminal. Symbol PRINT will be set to true or false after you type Y, N, the RETURN key, or the ESCAPE key (if escape recognition is enabled).

9.6.3 Ask for Definition of a Numeric Symbol (.ASKN)

The .ASKN directive displays on the terminal a request for a numeric value, waits for it to be entered, optionally tests the range for the numeric response and/or applies a default value, and sets the specified symbol accordingly. If the symbol has not previously been defined, Indirect makes an entry in the symbol table. If the symbol has already been defined, Indirect resets its value in accordance with the reply. Indirect exits with a fatal error if the symbol was previously defined as a logical or string symbol.

Formats (brackets are required syntax)

```
.ASKN   ssssss txt-strng
.ASKN   [low:high:default:timeout] ssssss txt-strng
```

Parameters

ssssss

The 1- through 6-character symbol to be assigned a numeric value.

txt-strng

The question or prompt that Indirect displays.

low:high

A numeric expression or symbol giving the range for the response.

default

A numeric expression or symbol giving the default value.

timeout

The time-out count. Indirect waits this long for a response, then applies the default answer. The format for time-out is nnu, where nn is the decimal number of time units to wait and u is T (ticks), S (seconds), M (minutes), or H (hours). The time-out count is valid only if time-out mode is enabled (.ENABLE TIMEOUT; see Section 9.6.12).

The entire .ASKN statement must fit on one command line.

Note that if you omit any of the parameters within the square brackets, any preceding colons are required for positonal identification.

The command line cannot exceed 132(decimal) characters in length. When executing a .ASKN directive, Indirect displays (unless .DISABLE DISPLAY is in effect) txt-strng prefixed by an asterisk (*) and suffixed either with [O]: to indicate that the response will be taken as octal, or with [D]: to indicate that the response will be taken as decimal. The reply must be a number either within the specified range or in the range 0 through 177777(octal) (by default) or 0 through 65535(decimal).

If the response is outside the specified range, the following message is displayed:

```
AT. -- Value not in range
```

Indirect then repeats the query.

If an arithmetic operation yields a result greater than 177777(octal) when computing the actual value of any of the arguments low, high, or default, a fatal error occurs and the following message is displayed:

```
AT. -- Numeric under- or overflow
```

If the response is an empty line (null) and a default value (default) was not specified, Indirect applies a default of 0. Note that in this case, the range, if specified, must include 0.

The response may be either octal or decimal; a leading pound sign (#) forces octal, a trailing period (.) forces decimal. In the absence of either, Indirect applies a default radix. The default radix is decimal if either the range or default values are decimal expressions (followed by a period). Otherwise, the default radix is octal (unless decimal mode has been enabled). Indirect displays the default type as either [O] or [D].

To force a default decimal radix without specifying a range argument, use the following construction:

```
.ASKN [::0.] A Enter value
```

or

```
.ENABLE DECIMAL
.ASKN A Enter value
```

Examples

The directive statement

```
.ASKN SYM Define numeric symbol A
```

displays the following on the terminal:

```
$ * Define numeric symbol A [O]:
```

In this example, [O] is the default radix (octal).

Indirect then defines symbol SYM according to the reply entered.

In this next example, the directive statement

```
.ASKN [2:35:16:20S] NUMSYM Define numeric symbol A
```

displays the following on the terminal:

```
$ * Define numeric symbol A [O R:2-35 D:16 T:20S]:
```

The format used in this displayed is as follows:

```
    [x R:low-high D:default T:timeout],
```

Where:

x	O if the default radix is octal or D if it is decimal.
R:low-high	The specified range.
D:default	The specified default.
T:timeout	The specified time-out count before the default answer is used.

Indirect then checks whether the response string is in the specified range.

In the last example, the directive statement

```
.ASKN [NUMSYM+10:45:NUMSYM+10] SYM Define numeric symbol B
```

displays the following on the terminal (assuming the value of 16(octal) for NUMSYM):

```
$ * Define numeric symbol B [O R:26-45 D:26]:
```

9.6.4 Ask for Definition of a String Symbol (.ASKS)

The .ASKS directive displays on the terminal a request for a string value to define a specified symbol and optionally tests whether the number of characters in the response string falls within the specified range. If the symbol has not previously been defined, Indirect makes an entry in the symbol table. If the symbol has already been defined, Indirect resets its value in accordance with the reply. Indirect exits with a fatal error if the symbol was defined previously as a logical or numeric symbol. If the number of characters is out of the specified range, the following message is displayed:

```
AT. -- String length not in range
```

Indirect then repeats the query.

Formats (brackets are required syntax)

.ASKS ssssss txt-strng

.ASKS [low:high:default:timeout] ssssss txt-strng

Parameters

ssssss

The 1- through 6-character symbol to be assigned a string value.

txt-strng

The question or prompt that Indirect displays.

low:high

A numeric expression giving the range for the number of characters permitted in the response string.

default

A string expression or symbol giving the default value.

timeout

The time-out count. Indirect waits this long for a response, then applies the default answer. The format for timeout is nnu, where nn is the decimal number of time units to wait and u is T (ticks), S (seconds), M (minutes), or H (hours). The time-out count is valid only if time-out mode is enabled (.ENABLE TIMEOUT; see Section 9.6.12).

The entire .ASKS statement must fit on one command line.

Note that if you omit any of the parameters within the square brackets, any preceding colons are required for positional identification.

When executing a .ASKS directive, Indirect displays (unless .DISABLE DISPLAY is in effect) txt-strng prefixed by an asterisk (*) and suffixes it with [S]:. the reply must be an ASCII character string.

Examples

The directive statement

```
.ASKS NAME Please enter your name
```

displays the following on the terminal:

```
$ * Please enter your name [S]:
```

Indirect then defines symbol NAME according to the string reply entered.

In the next example, the directive statement

```
.ASKS [1:15::10S] MIDNAM Please enter your middle name
```

displays the following on the terminal:

```
$ * Please enter your middle name [S R:1-15 T:10S]:
```

The format used in this displayed is as follows:

```
    [S R:low-high T:timeout]
```

Where:

S	The symbol type (string).
R:low-high	The specified range for the number of characters.
T:timeout	The specified timeout count.

9.6.5 Begin Block (.BEGIN)

The .BEGIN directive marks the beginning of a Begin-End block. The block must be terminated with a .END directive.

Labels and local symbols defined following the .BEGIN directive are local to the block instead of being used throughout the entire command file. Therefore, labels and local symbols defined inside a block lose definition outside the block.

Symbols defined outside a block retain definition throughout the file. Symbols defined outside a block and then modified within the block, however, assume and retain the value assigned in the block.

Labels defined outside a block are not accessible by a .GOTO directive from within the block. They are, however, accessible by a .GOSUB directive because program control returns to the next line within the block.

Labels and local symbols defined within a block lose definition with an .ERASE LOCAL directive statement (see Section 9.6.14) or with the .END directive.

The .BEGIN directive must be the only directive on a command line. For example, it cannot appear on the same line as an .IF directive.

Format

```
.BEGIN
```

9.6.6 Continue Processing Using Another File (.CHAIN)

The .CHAIN directive, which must be the last command in the file, closes the current file, erases all local symbols, clears any .ONERR arguments, empties the direct-access label cache, and continues processing using command lines from another file. The .CHAIN directive does not close data files, pass parameters, or change the nested file level.

Format (brackets not part of syntax)

```
.CHAIN   filespec[/switch(es)]
```

Parameters

filespec

The specification (including a directory, if desired) of the file that contains the new command lines. This parameter can also be a logical name assignment that translates into a valid FCS file specification.

/switch(es)

Any of the optional switches described in Section 9.5.

Example

```
.CHAIN OUTPUT
```

This directive statement transfers control to the file OUTPUT.CMD.

```
.CHAIN TEMP
```

This directive statement transfers control to the command file specified by the logical translation of TEMP.

9.6.7 Close Secondary File (.CLOSE)

The .CLOSE directive closes the secondary file opened by a .OPEN directive (see Section 9.6.22).

Format (brackets not part of syntax)

```
.CLOSE  [#n]
```

Parameter

n

An optional file number in the range 0 to x-1, where x is the number of file-open FDBs specified in the build file for the Indirect task. (The value x is the maximum number of files that can be open simultaneously.) The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

9.6.8 Output Data to Secondary File (.DATA)

The .DATA directive specifies text that is to be output to a secondary file previously opened by a .OPEN directive.

When Indirect processes the text string that follows the .DATA directive, it ignores the leading space (if present), assuming it to be a separator between the directive and the text string. Any other spaces are transferred to the data file. If a tab follows the directive, it is transferred to the file. If no other characters follow the directive, a blank line is transferred to the file. This processing has the following results:

Command File	Open File
.DATA foo RET	foo RET
.DATA foo RET	foo RET

Command File	Open File
.DATA[TAB] foo[RET]	[TAB] foo[RET]
.DATA[TAB] foo[RET]	[TAB] foo[RET]
.DATA[RET]	null line

Note that if a comment follows a .DATA statement (that is, .DATA data !comment), Indirect also outputs the comment to the secondary file because it cannot tell if the comment pertains to the .DATA statement itself or to the data being output to the file.

Format (brackets not part of syntax)

.DATA [#n] txt-strng

Parameters

n

An optional file number in the range 0 to x-1, where x is the number of file-open FDBs specified in the build file for the Indirect task. (The value x is the maximum number of files that can be open simultaneously.) The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

txt-strng

The text to be output to the secondary file.

The command line cannot exceed 132(decimal) characters and the specified text string cannot continue onto the next line. If a secondary file is not open, an error condition exists; Indirect issues an error message and begins error processing.

Example

```
.SETS SEND "This is data"
.OPEN TEMP
.DATA 'SEND'
.CLOSE
```

These directives output THIS IS DATA to the secondary file TEMP.DAT (.DAT is the default file type for a data file).

9.6.9 Decrement Numeric Symbol (.DEC)

The .DEC directive decrements a numeric symbol by 1. Indirect exits with a fatal error if the symbol was defined previously as a logical or string symbol.

Format

.DEC ssssss

Parameter

ssssss

The 1- through 6-character numeric symbol.

Example

```
.DEC X
```

This directive decrements by 1 the value assigned to the numeric symbol X. If X crosses the zero boundary (goes from positive to negative), decrementing it will cause an underflow error.

9.6.10 Delay Execution for a Specified Period of Time (.DELAY)

The .DELAY directive delays further processing of the file for a specified period of time.

Format

.DELAY nnu

Parameters

nn

The decimal number of time units to delay.

u

T	—	Ticks
S	—	Seconds
M	—	Minutes
H	—	Hours

The parameter nn is decimal by default, or octal if preceded by a pound sign (#). For example:

10S	=	10(decimal) seconds
#10S	=	10(octal) seconds

If quiet mode is disabled when the .DELAY directive is executed, Indirect issues the following message:

```
AT. -- Delaying
```

When the time period expires and the task resumes, Indirect issues this message:

```
AT. -- Continuing
```

Example

```
.DELAY 20M
```

This directive statement delays processing for 20(10) minutes.

9.6.11 Disable Option (.DISABLE)

The .DISABLE directive disables a specified operating mode previously activated by an .ENABLE directive. See Section 9.6.12 for information on the operating modes.

Format

.DISABLE option[,option...]

Parameter

option

One or more of the operating modes described in Section 9.6.12.

The following is a list of the operating modes that can be disabled:

ATTACH	DELETE	GLOBAL	SUBSTITUTION
CLI	DETACH	LOWERCASE	TIMEOUT
CONTROL-Z	DISPLAY	OVERFLOW	TRACE
DATA	ESCAPE	QUIET	TRUNCATE
DECIMAL	ESCAPE-SEQ		

Note that when you disable DETACH mode from a command file and then request a task or DCL command to display information, the command file may not be able to continue executing. The task or DCL command may need to attach to the terminal to display the information, but will not be able to do so because Indirect cannot detach from the terminal.

9.6.12 Enable Option (.ENABLE)

The .ENABLE directive is used to invoke several operating modes. Each mode is independent of the others; all of them can be active simultaneously. When Indirect starts to process the highest-level command file, the initial settings are as follows:

ATTACH	enabled (R)	ESCAPE-SEQ	disabled (R)
CLI	enabled (R)	GLOBAL	enabled (I)
CONTROL-Z	disabled (I)	LOWERCASE	enabled (I)
DATA	disabled (I)	OVERFLOW	disabled (R)
DECIMAL	disabled (R)	QUIET	disabled (R)
DELETE	disabled (I)	SUBSTITUTION	disabled (I)
DETACH	enabled (R)	TIMEOUT	enabled (R)
DISPLAY	enabled (R)	TRACE	disabled (I)
ESCAPE	disabled (I)	TRUNCATE	disabled (R)

However, when Indirect passes control to a lower-level command file by means of a .CHAIN filename or @filename statement, only the following modes are reset to their initial (denoted by "I" in the previous list) settings: CONTROL-Z, DATA, DELETE, ESCAPE, GLOBAL, LOWERCASE, SUBSTITUTION, and TRACE. The remaining operating modes retain (denoted by "R" in the previous list) their new settings in the lower-level file.

In **ATTACH** mode, Indirect attaches to a terminal when displaying comment lines. In **DETACH** mode, it detaches from the terminal when processing command lines. Enabling both of these modes allows you to type CTRL/O to suppress a lengthy comment.

Attach and detach modes perform conditional IO.ATT and to IO.DET terminal-QIO functions, depending on the setting of the ATTACH or DETACH attribute bits in an internal flag word that controls the operating modes. However, disabling the detach mode always attaches the terminal until a DCL command is issued or a .ENABLE QUIET statement is encountered (see the following description). Thus, if you want the terminal to remain detached while quiet mode is in effect, enable both attach and detach modes at the beginning of your indirect command file or interactive terminal session. Also, if you are going to toggle in the attach and detach operating modes during the execution of the command file, follow the .ENABLE/.DISABLE statement with a .ENABLE QUIET statement.

Enabling **CONTROL-Z** mode allows a command file to detect a CTRL/Z response to a question and continue processing. If Control-Z mode is disabled and you type CTRL/Z in response to a .ASKx question, Indirect exits. If Control-Z mode is enabled, the special symbol <EOF> is set to true and Indirect continues processing the command file.

In **DATA** mode, Indirect outputs lines that follow a .ENABLE DATA directive statement to a secondary file. (The .DATA directive sends a single line of text to a secondary file.) To disable data mode, the .DISABLE DATA (or .CLOSE) statement must begin in the first column. Otherwise, Indirect copies the statement itself into the data file. The .ENABLE DATA directive also has an optional argument (#n) that specifies which file the data is to go into. See the description of the .DATA directive (Section 9.6.8) for more information.

In **GLOBAL** symbol mode, symbol names that begin with a dollar sign ($) are defined as global to all levels of indirect files; once such a symbol has been defined, all levels recognize it. Symbols that do not begin with a dollar sign are recognized only within the level that defines them.

In **DECIMAL** mode, all numeric symbols are created or redefined by default as decimal instead of octal.

In **DELETE** mode, the current command file is deleted when Indirect processes the last command line in the file.

In **DISPLAY** mode, Indirect displays the current fields for the .ASKx directive and @ <EOF> . If display mode is disabled, Indirect displays only the text string for the .ASKx directive and suppresses @ <EOF> .

In **CLI** mode, commands not processed by Indirect are passed to your CLI. The default CLI for Micro/RSX is DCL. CLI mode is equivalent to the function of the /CLI switch (see Section 9.5).

In **LOWERCASE** mode, characters read from the terminal in response to .ASKS directives are stored in the string symbol without lower- to uppercase conversion. The representation of characters is significant when comparing strings (see Section 9.6.18) because the .IF directive distinguishes between lowercase and uppercase characters.

In **SUBSTITUTION** mode, Indirect substitutes a string for a symbol. The symbol must begin and end in apostrophes ('symbol'). For example, if the symbol A has been assigned the string value THIS IS A TEST, then every 'A' will be replaced by THIS IS A TEST. When substitution mode is enabled, Indirect performs substitutions in each line before scanning the line for directives and DCL commands. (While obeying a .GOTO label directive, however, Indirect ignores any undefined symbols encountered before the target line, that is, the line containing the specified label.)

ESCAPE recognition permits the response to a .ASK, .ASKN, or .ASKS directive to be an escape character. A question answered with a single escape character sets the special logical symbol <ESCAPE> to true. The escape character must be used only as an immediate terminator to the question; if one or more characters precede the escape character, an error condition exists. In this case, the following message is displayed:

```
AT. -- Invalid answer or terminator
```

Indirect then repeats the question. Note that if you press the ESCAPE key in response to a .ASK directive, the specified logical symbol (ssssss of .ASK ssssss txt-strng) is also set to true.

ESCAPE-SEQuence processing forces Indirect to attach to the terminal for escape-sequence recognition, using the IO.ATT!TF.SEQ I/O function. In this mode, the result of a .ASKx or .READ statement from the terminal will contain the terminating escape character and escape sequence as documented in the full-duplex terminal driver chapter of the *Micro/RSX I/O Drivers Reference Manual.*

OVERFLOW mode allows signed arithmetic in numeric expressions. When you enable overflow mode, Indirect evaluates numeric expressions as signed integers rather than as unsigned integers. Enabling this mode provides for numeric expressions and operations that otherwise would result in the "Numeric under- or overflow" error message.

In **QUIET** mode, Indirect does not echo DCL command lines or comments. The command lines are executed normally and, if they return a message or display, the message or display is shown on the terminal.

In **TIMEOUT** mode, Indirect uses the time-out parameters specified with the .ASKx directives. Indirect waits for the time-out count to elapse and then applies the default answer to the directives. Time-out mode must be enabled (the default) to use the time-out counts for the .ASKx directives.

In **TRACE** mode, command lines that Indirect has processed are displayed on the terminal. As each line is processed, it is displayed with its nesting level and an exclamation point (!). Trace mode is equivalent to the function of the /TR switch (see Section 9.5).

In **TRUNCATE** mode, Indirect ignores any truncation errors on a .READ directive. A truncation error occurs when a line in a file is too long. If the full record cannot fit within the 132(decimal)-character limit of the symbol, the record is truncated.

Formats (brackets not part of syntax)

```
.ENABLE   option[,option...]
.ENABLE DATA   [#n]
```

Parameters

option

One or more of the operating modes previously described.

#n

An optional file number in the range 0 to x-1, where x is the number of file-open FDBs specified in the build file for the Indirect task. (The value x is the maximum number of files that can be open simultaneously.) The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

Examples

SUBSTITUTION mode:

```
.ENABLE SUBSTITUTION
.ASKS FILE Specify next file
PRINT 'FILE'
```

When the command file is executing, the corresponding lines displayed at the terminal are:

```
$ * Specify next file [S]: SOURCES RET
$ PRINT SOURCES
```

GLOBAL symbol mode:

The following two lines appear in an indirect command file called TEST1:

```
.ENABLE GLOBAL
.SETS $X "TEST"
```

A file called TEST2.CMD contains the following lines:

```
.ENABLE GLOBAL
.ENABLE SUBSTITUTION
@TEST1
RUN '$X'
```

DCL displays the following lines when the file TEST2.CMD is run:

```
$ RUN TEST
$ @ <EOF>
```

ESCAPE-recognition mode:

```
         ;If you want a list of options, type <ESC>.
         .ENABLE ESCAPE
         .ASKS A Enter option
         .IFT <ESCAPE> .GOTO LIST
              .
              .
              .
 .LIST:  ;Options are: A (ADD), S (SUBTRACT), M (MULTIPLY)
         .ASKS A Enter option
```

If you press the ESCAPE key in response to ENTER OPTION, the lines displayed at the terminal are:

```
$ ;If you want a list of options, type <ESC>.
$ * Enter option [S]: <ESC>
$ ;Options are: A (ADD), S (SUBTRACT), M (MULTIPLY)
$ * Enter option [S]:
```

QUIET mode:

```
.ASK QUIET Do you want command lines suppressed
.IFT QUIET .ENABLE QUIET
.IFF QUIET .DISABLE QUIET
SHOW TASKS/ACTIVE
```

If the response is affirmative, Indirect displays the active tasks but not the SHOW TASKS/ACTIVE command. For example:

```
$ * Do you want command lines suppressed? [Y/N]: Y
DCL
SHOT14
AT.T14
```

CONTROL-Z mode:

```
.ENABLE CONTROL-Z
.ASK RESP Do you want to continue
```

```
.IFT <EOF> .GOTO CLENUP
.IFF RESP .GOTO CLENUP
```

If you press CTRL/Z in response to the question, <EOF> is set to true and Indirect transfers to label CLENUP.

9.6.13 End Block (.END)

The .END directive marks the end of a Begin-End block. If Indirect encounters more .END directives than .BEGIN directives, command processing terminates and the following message is displayed:

```
AT. -- Illegal nesting
```

Format

.END

As with .BEGIN, the .END directive must be the only directive on the command line.

9.6.14 Delete Symbols (.ERASE)

The .ERASE directive deletes all local or global symbol definitions, or a specific global symbol definition.

When you define a symbol either locally (by defining a symbol value) or globally (by enabling global symbol mode and preceding the symbol name with a dollar sign ($)), Indirect creates an entry in the symbol table. The .ERASE directive erases either all local or all global entries, or a specific global entry, in the table.

Following a .ERASE directive, you can redefine a symbol's value as well as its type.

Format

.ERASE LOCAL

.ERASE GLOBAL

.ERASE SYMBOL global-symbol

An .ERASE LOCAL directive outside of a Begin-End block erases all local symbols defined within the current file.

An .ERASE LOCAL directive within a Begin-End block erases only those local symbols defined within the block.

However, note that the following actions also occur:

1. Local symbols defined within a nested file are erased when that file exits.
2. Local symbols defined within a Begin-End block are erased by .END.
3. Local symbols defined outside of Begin-End blocks are visible, modifiable, and not erasable within a Begin-End block.

Also note that .ERASE LOCAL statement will not work if it is included in a command file invoked with the /LO (local) switch, a command file invoked with the at sign (@file), or by a .CHAIN directive with /LO, or a file called by the .CHAIN directive with /LO. This restriction occurs because Indirect uses its internal stack for local symbols, nested command file context, and Begin-End block context storage. When a command file is invoked with the at sign or is called by .CHAIN with /LO, the context of the calling command file is placed on the internal stack for later use. But when Indirect processes the .ERASE LOCAL statement in the invoked file, it removes this context along with any local symbols defined by the command files. This behavior causes Indirect to abort with an error when it later attempts to remove the context of the calling file (or the Begin-End context, which is also placed on the stack) from the internal stack.

An .ERASE GLOBAL directive, either outside of or within a Begin-End block, erases all global symbols.

An .ERASE SYMBOL global-symbol directive erases the specified global symbol. (Individual local symbols are not erasable.)

Examples

```
.ERASE LOCAL
```

This directive erases all local symbol definitions used in the indirect command file.

```
.ERASE SYMBOL $SWITC
```

This directive erases the single global symbol "$SWITC."

9.6.15 Exit Current Command File (.EXIT)

The .EXIT directive terminates processing of the current command file or Begin-End block and returns control to the previous-level command file or, if the directive is executed within a block, to the line following the .END directive. If the directive is encountered at the uppermost indirect nesting level, Indirect exits and passes control to DCL (see the .STOP directive, Section 9.6.33).

The .EXIT directive also allows you to optionally specify a value to copy into the special symbol <EXSTAT> .

Format (brackets not part of syntax)

```
.EXIT   [value]
```

Parameter

value

An optional numeric expression to be copied to the special symbol <EXSTAT> .

Example

The following line appears in an indirect command file called TEST1:

```
@TEST2
```

The file TEST2.CMD contains the following line:

```
.EXIT
```

When Indirect encounters the .EXIT directive in TEST2, control returns to TEST1.CMD.

If the .EXIT directive in TEST2.CMD includes a numeric expression (for example, .EXIT N2), Indirect evaluates the expression and copies the value into <EXSTAT> .

9.6.16 Call a Subroutine (.GOSUB)

The .GOSUB directive saves the current position in an indirect command file and then branches to a label. The label identifies an entry point to a subroutine that is terminated by a .RETURN directive.

When you issue a .GOSUB directive from within a Begin-End block, Indirect saves the current block context and then scans the file for the first occurrence of the subroutine label. Note that during the scan, Indirect ignores any intervening .BEGIN or .END directives. The .RETURN directive restores previous block context. Thus, the subroutine can be contained within a Begin-End block.

The maximum nesting depth for subroutine calls depends on the number specified in the build file for the Indirect task.

Format

.GOSUB label parameters

Parameter

label

The label that designates the first line of a subroutine, but without the leading period and trailing colon. Any parameters to the right of the label and to the left of a comment are transferred to the reserved local symbol COMMAN. The value of COMMAN can then be parsed with the .PARSE directive (see Section 9.6.25) to obtain formal call parameters.

Example

```
.GOSUB EVAL
```

This directive statement transfers control to the subroutine labeled .EVAL:.

9.6.17 Branch to a Label (.GOTO)

The .GOTO directive causes a branch from one line in an indirect command file to another. All commands between the .GOTO directive and the specified label are ignored. Branches can go forward or backward in the file.

The target of a .GOTO branch from within a Begin-End block must be contained in that block. The .GOTO directive cannot branch into another block. When Indirect encounters a .GOTO directive within a Begin-End block, it searches for the specified label in that block. Since Indirect only searches the one Begin-End block, you can use the same label more than once in a command file.

See Section 9.6.1 for more information on labels and direct-access labels.

Format

.GOTO label

Parameter

label

The name of the label, but without the leading period and trailing colon.

Example

```
.GOTO  100
```

This directive statement transfers control to the line containing the label .100:.

9.6.18 Logical Test (.IF)

A number of directives make tests. If the result of the test is true, Indirect processes the remainder of the command line. Logical tests can be combined into a compound logical test by using the .AND and .OR directives.

9.6.18.1 Test if Symbol Meets Specified Condition (.IF)

The .IF directive compares a numeric or string symbol with another expression of the same type to determine if one of several possible conditions is true. If the condition is satisfied, Indirect executes the remainder of the command line.

When comparing a string symbol with a string expression, Indirect compares the ASCII values of each operand's characters (from left to right) one by one. An operand is considered greater if the first nonequal character has a greater value than the corresponding character in the other operand.

Numeric symbols are compared strictly on the basis of magnitude. If overflow mode is enabled (see Section 9.6.12), Indirect evaluates numeric expressions as signed integers rather than as unsigned.

Format

```
.IF  symbol relop expr directive-statement
```

Parameters

symbol

The 1- through 6-character logical, numeric, or string symbol.

relop

One of the following relational operators:

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

expr

An expression of the same type as symbol.

directive-statement

The Indirect command line to be processed if the condition is satisfied.

Examples

```
.SETS X "A"
.SETS Y "a"
.IF X LT Y  .GOTO 200
```

The ASCII value of string symbol X is less than the ASCII value of string symbol Y, which satisfies the less-than condition. Thus, control passes to the line containing the label .200:.

```
.SETN N1 2
.SETN N2 7
.IF N1 <= N2  DIR
```

With the condition satisfied (numeric symbol N1 less than or equal to numeric symbol N2), the DIRECTORY command is executed.

```
.SETS S1 "AAb"
.SETS S2 "AA"
.SETS S3 "BBBB"
.IF S1 >= S2+S3[1:1]  .INC A
```

Because string symbol S1 is greater than or equal to string symbol S2 concatenated with the first character of string symbol S3 (AAb > = AAB), that condition is satisfied and Indirect increments numeric symbol A.

9.6.18.2 Test if Task Is Active or Dormant (.IFACT/.IFNACT)

The .IFACT and .IFNACT directives test whether a task is active (.IFACT) or dormant (.IFNACT). If the result of the test is true, the remainder of the command line is processed. If the specified task is not installed, Indirect assumes the dormant condition.

Formats

.IFACT taskname directive-statement

.IFNACT taskname directive-statement

Parameters

taskname

A 1- through 6-character legal task name.

directive-statement

The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFACT   REPORT  .GOTO 350
.IFNACT  REPORT  RUN REPORT
```

9.6.18.3 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 result of the test is true, the remainder of the command line is processed. These directives do not test the value of the symbol.

Formats

.IFDF ssssss directive-statement

.IFNDF ssssss directive-statement

Parameters

ssssss

The 1- through 6-character symbol being tested. The symbol can be local, global, or an Indirect special symbol.

directive-statement

The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFDF  A  .GOTO 100
.IFNDF  A  .ASK A Do you want to set the time
```

9.6.18.4 Test if Task is Installed or Not Installed (.IFINS/.IFNINS)

The .IFINS and .IFNINS directives test whether a task is installed (.IFINS) or not installed (.IFNINS) in the system. If the result of the test is true, the remainder of the command line is processed.

Formats

.IFINS taskname directive-statement

.IFNINS taskname directive-statement

Parameters

taskname

A 1- through 6-character task name.

directive-statement

The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFINS  LP1  .GOTO 250
.IFNINS  LP1  INS $LP1
```

9.6.18.5 Test if Mode is Enabled or Disabled (.IFENABLED/.IFDISABLED)

The .IFENABLED and .IFDISABLED directives test whether an operating mode has been enabled with the .ENABLE directive or disabled with the .DISABLE directive. See the description of the .ENABLE directive in Section 9.6.12 for the list of operating modes.

Formats

.IFENABLED option directive-statement

.IFDISABLED option directive-statement

Parameters

option

The same operating mode option (with the exception of DATA) used with the .ENABLE or .DISABLE directive, or one of the following options:

FULL-DUPLEX	The full-duplex terminal driver is present in the system; default is enabled.
LOCAL	The /LO switch was specified in the initial command line; default is enabled.
POTASK	Parent-offspring tasking support is included in the current system; default is enabled.

directive-statement

The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFENABLED CLI .GOTO SHOW
.IFDISABLED DECIMAL .ENABLE DECIMAL
```

9.6.18.6 Test if Driver is Loaded or Not Loaded (.IFLOA/.IFNLOA)

The .IFLOA and .IFNLOA directives test whether a driver is loaded (.IFLOA) or not loaded (.IFNLOA) in the system. If the result of the test is true, the remainder of the command line is processed. Note that for the purposes of these directives, resident drivers are assumed to be loaded.

Formats

.IFLOA dd: directive-statement

.IFNLOA dd: directive-statement

Parameters

dd:

A device driver

directive-statement

The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFLOA DU: .GOTO 250
.IFNLOA DU: LOA DU:
```

9.6.18.7 Test If Symbol Is True or False (.IFT/.IFF)

The .IFT and .IFF directives test whether a logical symbol is true (.IFT) or false (.IFF). If the result of the test is true, Indirect processes the remainder of the command line.

Indirect exits with a fatal error if the symbol being tested was previously defined as a numeric or string symbol.

Formats

.IFT ssssss directive-statement

.IFF ssssss directive-statement

Parameters

ssssss

The 1- through 6-character logical symbol being tested.

directive-statement

The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFT  A  .GOTO  100
.IFF  B  .GOTO  200
```

9.6.18.8 Compound Tests

You can combine .IF tests by using the .AND and .OR directives. In addition, an implied .AND is effected when more than one .IF appears on the same line without being separated by an .AND directive.

The .AND directive takes precedence over the .OR directive as shown in the following example:

```
.IFT A .OR .IFT B .AND .IFT C .GOTO D
```

That is, Indirect reads the line as:

```
.IFT A .OR (.IFT B .AND .IFT C) .GOTO D
```

Examples

```
.IFT A .AND .IFF B  .GOTO HELP
```

If the logical symbol A is true and the logical symbol B is false, control passes to the line containing the label .HELP:.

```
.IFT A .IFF B  .GOTO HELP
```

This has the same effect as the previous directive (.AND implied).

```
.IFT A .OR  .IFF B RUN WAY
```

If the logical symbol A is true or if the logical symbol B is false, the RUN command is issued.

9.6.19 Increment Numeric Symbol (.INC)

The .INC directive increments a numeric symbol by 1. Indirect exits with a fatal error if the symbol was previously defined as a logical or string symbol.

Format

.INC ssssss

Parameter

ssssss

The 1- through 6-character numeric symbol being incremented.

Example

```
.INC B
```

This increments by 1 the value assigned to the numeric symbol B. If B crosses the zero boundary (goes from negative to positive), incrementing it will cause an overflow error.

9.6.20 Define Logical End-of-File (/)

The logical end-of-file directive (/) terminates file processing at all levels, closes all open data files, and exits. Indirect then displays (if display mode has not been disabled) the following message:

```
   $ @ <EOF>
```

Format

```
   /
```

The directive is the first nonblank character of the line.

You can use this directive at any location in the command file to quickly terminate file processing, but care should be taken to avoid an inadvertent exit.

Example

```
        .ASK CONT Do you wish to continue
         .IFT CONT .GOTO 100
         /
.100:
```

9.6.21 Branch to Label on Detecting an Error (.ONERR)

If Indirect detects one of the following errors:

- Task not installed in system (.XQT, .WAIT)
- Undefined symbol
- Bad syntax (.XQT, .WAIT, .DELAY)
- Unrecognized command
- String substitution error
- Symbol type error (.IF, .IFT, .IFF, .INC, .DEC)

- Redefinition of a symbol to a different type (.ASK, .ASKN, .ASKS, .SETT, .SETF, .SETL, .SETN, .SETD, .SETO, .SETS)
- Data file error (.OPEN, .OPENA, .OPENR, .DATA, .CLOSE, or .READ between .ENABLE DATA and .DISABLE DATA)

control passes to the line containing the specified label. This feature provides you with a means of gaining control to terminate command file processing in an orderly manner.

Note that the .ONERR directive applies only to the error conditions listed; errors returned from a task external to Indirect (for example, a DCL syntax error) are not processed by the .ONERR directive.

Format

```
.ONERR   [label]
```

Parameter

label

The name of the label, but without the leading period and trailing colon. The brackets are not part of the required syntax.

Upon detecting an error, Indirect passes control to the line starting with .label:. The .ONERR directive must be issued before Indirect encounters the error condition. If the directive is executed (one of the listed errors is encountered), error processing passes to the specified label. If the label specified by the .ONERR directive does not exist and an error condition has occurred, command processing terminates.

If you do not specify the optional label, Indirect disables processing for the previous .ONERR directive.

If you want to have .ONERR processing and Begin-End blocks in a program, the label you specify must be located on the same block level as the .ONERR directive. When Indirect detects an error, it passes control to the most recently defined .ONERR label in the current block level or in a previous, lower block level.

Once a .ONERR condition has occurred, another .ONERR directive must be issued to trap a future error.

The .ONERR directive works with the special symbol <ERRCTL> (see Section 9.4.1.2). For each class of error that a .ONERR target routine processes, the appropriate bit is set in the symbol. The initial default value for <ERRCTL> is 1, which implies that only class 1 errors can be handled with a .ONERR routine. (Note that if you attempt to process errors other than default class 1, Indirect cannot continue in most cases. The error service routine is limited to a fatal error message and .EXIT. The internal state of Indirect is indeterminate in all but class 1 error cases.) After Indirect has processed the .ONERR directive, <ERRCTL> is reset to 1. See Section 9.8 for a list of error messages and their assigned class values.

Example

```
.ONERR  100
```

Upon detecting one of the error conditions, Indirect passes control to the line labeled .100:.

9.6.22 Open Secondary File (.OPEN)

The .OPEN directive opens a specified secondary file as an output file. The .DATA directive is used to place data in this secondary file.

Format (brackets not part of syntax)

.OPEN [#n] filespec

Parameters

#n

An optional file number in the range 0 to x-1, where x is the number of file-open FDBs specified in the build file for the Indirect task. (The value x is the maximum number of files that can be open simultaneously.) The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

filespec

A file to be opened as an output file. The default file type is .DAT.

Indirect sets the owner UIC of the file being opened to be the current protection UIC of the user. All FCS protection and privilege checks are still in effect.

For nonprivileged users, the protection UIC is always the same as their login UIC. If you are not in named directory mode, you can change only your default UIC with the SET UIC or SET DEFAULT command. If you are in named directory mode, the SET UIC command is illegal and SET DEFAULT changes only your default UIC.

For privileged users, the protection UIC can change. If you are not in named directory mode, you can use SET UIC and SET DEFAULT to change both your protection UIC and your default UIC. If you are in named dirctory mode, the SET UIC command changes only your protection UIC. Your default UIC remains the same. The SET DEFAULT command changes only your default UIC. Your protection UIC remains the same.

For more information on the SET UIC and SET DEFAULT commands, see Volume 2, Chapter 12.

The parameter filespec can also be a logical name assignment that translates into a valid FCS file specification.

You cannot specify a fixed-length record file with the .OPEN directive. If you do, Indirect changes the attribute of the file from fixed-length to variable-length when it closes the file.

Note that you cannot include a comment that begins with a semicolon (;comment) in an .OPEN statement. Doing so results in a syntax error. Comments that begin with an exclamation point (!comment) are accepted.

Examples

```
.OPEN SECOUT
```

This directive opens the file SECOUT.DAT as an output file.

```
.OPEN TEMP
```

This directive opens the file specified by the logical translation of TEMP.

The command file HIHO.CMD contains the following directive statement:

```
.OPEN GRUMPY
```

When the following command lines are executed:

```
$ ASSIGN DU2:[DWARVES]GRUMPY.DAT GRUMPY [RET]
$ @HIHO [RET]
```

Indirect opens the file DU2:[DWARVES]GRUMPY.DAT.

9.6.23 Open Secondary File for Append (.OPENA)

The .OPENA directive opens a secondary file and appends all subsequent data to the file.

Format (brackets not part of syntax)

.OPENA [#n] filespec

Parameters

n

An optional file number in the range 0 to x-1, where x is the number of file-open FDBs specified in the build file for the Indirect task. (The value x is the maximum number of files that can be open for append simultaneously.) The default is 0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

filespec

A secondary file to be opened with subsequent data appended to it. The default file type is .DAT.

Indirect sets the owner UIC of the file being opened to the current protection UIC of the user. See the description of the .OPEN directive for more information.

The parameter filespec can also be a logical name assignment that translates into a valid FCS file specification.

You cannot specify a fixed-length record file with the .OPENA directive. If you do, Indirect changes the attribute of the file from fixed-length to variable-length record when it closes the file.

Note that you cannot include a comment that begins with a semicolon (;comment) in an .OPENA statement. Doing so results in a syntax error. Comments that begin with an exclamation point (!comment) are accepted.

If the specified file does not already exist, .OPENA becomes the .OPEN directive by default.

Examples

```
.OPENA SECOUT
```

This directive opens the file SECOUT.DAT as an output file and appends subsequent data to it.

```
.OPENA TEMP
```

This directive opens the file specified by the logical translation of TEMP as an output file and appends subsequent data to it.

The command file BEAUTY.CMD contains the following directive statement:

```
.OPENA BEAST
```

When the following command lines are executed:

```
$ ASSIGN DU2:[TALES]BEAST.DAT BEAST RET
$ @BEAUTY RET
```

Indirect opens the file DU2:[TALES]BEAST.DAT and appends subsequent data to it.

9.6.24 Open File for Reading (.OPENR)

The .OPENR directive opens a file for reading with the .READ directive.

Format (brackets not part of syntax)

.OPENR   [#n] filespec

Parameters

n

An optional file number in the range 0 to x-1, where x is the number of file-open FDBs specified in the build file for the Indirect task. (The value x is the maximum number of files that can be open for reading simultaneously.) The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

filespec

A file to be opened for reading. The default file type is .DAT. Indirect sets the owner UIC of the file being opened to the current protection UIC of the user. See the description of the .OPEN directive for more information.

The parameter filespec can also be a logical name assignment that translates into a valid FCS file specification.

You cannot specify a fixed-length record file with the .OPENR directive. If you do, Indirect changes the attribute of the file from fixed-length to variable-length record when it closes the file.

Note that you cannot include a comment that begins with a semicolon (;comment) in a .OPENR statement. Doing so results in a syntax error. Comments that begin with an exclamation point (!comment) are accepted.

Examples

```
.OPENR INDADD
```

This directive opens the file INDADD.DAT for reading with the .READ directive.

```
.OPENR DATLIB.ULB/LB:DATINP
```

This directive opens for reading the library module DATINP that is contained in the univeral library DATLIB.

```
.OPENR TEMP
```

This directive opens for reading the file specified by the logical translation of TEMP.

The command file HANSEL.CMD contains the following directive statement:

```
.OPENR GRETEL
```

When the following command lines are executed:

```
$ASSIGN DU2:[WITCH]GRETEL.DAT GRETEL RET
$@HANSEL RET
```

Indirect opens the file DU2:[WITCH]GRETEL.DAT for reading.

9.6.25 Parse Strings into Substrings (.PARSE)

The .PARSE directive parses strings in a command line into substrings.

Format

.PARSE <STRING> <CONTROLSTRING> <VAL1> <VAR2> ...<VARN>

The string is broken up into substrings as specified by the control string. The substrings are stored in the specified variables. The first character of the control string delimits the first substring, the second character of the control string delimits the second substring, and so on. The last character of the control string is repeated if the number of variables exceeds the length of the control string. If you specify more variables than substrings, the additional variables are set to null strings. If you specify fewer variables than the number of substrings that can be parsed, the last variable contains the unparsed fragment of <STRING>. If the last character of <STRING> is a delimiter (for example, a comma (,) or a right angle bracket (>)), Indirect assumes that there is a null substring after it. If you do not specify a symbol for this substring for this substring to be parsed into, the delimiter and the substring are parsed into the last symbol specified.

The symbol <STRLEN> contains the actual number of substrings that Indirect processed (including explicit null substrings).

Examples

A command file, PARSE.CMD, contains the following command lines:

```
.ENABLE SUBSTITUTION
.PARSE COMMAN " ," FILE A1 A2 A3 A4 A5
;FILE = 'FILE'
;A1 = 'A1'
;A2 = 'A2'
;A3 = 'A3'
;A4 = 'A4'
;A5 = 'A5'
;<STRLEN> = '<STRLEN>'
```

The command file is invoked with the following command line:

```
$ @PARSE ARG1,ARG2,,ARG3 RET
```

When the file is executed, COMMAN contains "PARSE ARG1, ARG2,,ARG3" and Indirect displays the following information:

```
$ ;FILE = PARSE
$ ;A1 = ARG1
$ ;A2 = ARG2
$ ;A3 =
$ ;A4 = ARG3
$ ;A5 =
$ ;<STRLEN> = 5
```

The following example is from an interactive terminal session:

```
$ @TI: RET
AT.>ENABLE SUBSTITUTION RET
AT.>SETS A "1" . RET
AT.>PARSE A "," B C RET
AT.>;'B' RET
$ ;1
AT.>;'C' RET
$ ;                           (null substring)
AT.>PARSE A "," B RET
AT.>;'B' RET
$ ;1,
AT.>CTRL/Z
$ @ <EOF>
$
```

In this example, the first time string A is parsed, there are enough variables to contain the substring 1 and the implied null substring following it. The second time A is parsed, however, there are not enough variables to contain the substring, so the entire string (1,) is parsed into the specified variable.

9.6.26 Pause for Operator Action (.PAUSE)

The .PAUSE directive interrupts processing of an indirect command file to wait for user action. A .PAUSE directive causes Indirect to stop itself, after which you can perform some operations and subsequently cause the task to resume.

Format

.PAUSE

When Indirect stops itself, it displays the following message on the entering terminal:

```
AT. -- Pausing, To continue type "command taskname"
```

Where:

command

The command line to be issued to resume the task.

taskname

The name of the Indirect task.

You then type the appropriate command line to resume the task. Indirect displays the following message:

```
AT. -- Continuing
```

and continues processing where it left off.

Note that the .PAUSE directive is legal only if the command line interpreter for your terminal is DCL.

9.6.27 Read Next Record (.READ)

The .READ directive reads the next record into a specified string variable. The entire record is read into the variable. If the record is longer than 132(decimal) characters, an error occurs.

After every .READ operation, the special symbol <FILERR> contains the FCS-11 file code for the read and the special symbol <EOF> reflects whether an end-of-file was found. (Note that .OPENR does not clear <EOF> .) If an error or end-of-file occurs, the string variable remains unchanged from its previous state.

Format (brackets not part of syntax)

.READ [#n] ssssss

Parameters

n

An optional file number that specifies the file from which the record is to be read. The file number must be one of the numbers used in a previous .OPENR statement.

ssssss

The string variable into which the record will be read.

Example

```
          .ENABLE SUBSTITUTION
          .OPENR FILE
          .IF <FILERR> NE 1 .GOTO ERROR
.LOOP:
          .READ RECORD
          .IFT <EOF> .GOTO DONE
          .IF <FILERR> NE 1 .GOTO ERROR
          ; 'RECORD'
          .GOTO LOOP
            .
.ERROR:     .
            .
.DONE:    .CLOSE
            .
            .
            .
```

These directives open the file FILE.DAT for reading, read each record into the string variable RECORD, display each record on the terminal, and close the file.

9.6.28 Return from a Subroutine (.RETURN)

The .RETURN directive signifies the end of a subroutine and returns control to the line immediately following the .GOSUB directive that initiated the subroutine.

Format

.RETURN

9.6.29 Set Symbol to True or False (.SETT/.SETF/.SETL)

The .SETT, .SETF, and .SETL directives define or change the value of a specified logical symbol. If the symbol has not been defined, Indirect makes an entry in the symbol table and sets the logical symbol to the value specified. If the symbol has already been defined, Indirect resets the symbol accordingly. Indirect exits with a fatal error if the logical symbol was defined previously as a numeric or string symbol.

Formats

.SETT ssssss

.SETF ssssss

.SETL ssssss llllll

Parameters

ssssss

The 1- through 6-character logical symbol to be assigned a true or false value.

llllll

A logical or numeric expression. The symbol ssssss is assigned the value of llllll when the logical expression is evaluated.

Examples

```
.SETT X
```

This directive sets the logical symbol X to true.

```
.SETF ABCDE
```

This directive sets the logical symbol ABCDE to false.

```
.SETL TEST SWITCHA!SWITCHB
```

This directive sets the logical symbol TEST to true if SWITCHA or SWITCHB is true.

9.6.30 Set Symbol to Numeric Value (.SETN)

The .SETN directive defines or changes the value of a specified numeric symbol. If the symbol has not been defined, Indirect makes an entry in the symbol table and sets the symbol to the numeric value specified. If the symbol has already been defined, Indirect resets the symbol accordingly. Indirect exits with a fatal error if the numeric symbol was previously defined as a logical or string symbol.

Format

.SETN   ssssss numexp

Parameters

ssssss

The 1- through 6-character numeric symbol.

numexp

A numeric expression. (See Section 9.4.2.)

When specifying a numeric value to assign to a symbol, you may combine a numeric symbol or constant with another numeric symbol or constant to form a numeric expression. If numeric expressions are used, no embedded blanks or tabs are permitted. Evaluation is done from left to right unless parentheses are used to form subexpressions, which are evaluated first. The radix of an expression is octal if all the operands are octal and decimal mode has not been enabled; otherwise, the radix is decimal.

Examples

```
.SETN NUMBER 27
```

This directive assigns to the numeric symbol NUMBER the value 27(8).

```
.SETN A1 3*(A2-5)
```

This directive assigns the numeric symbol A1 the value of symbol A2 minus 5, multiplied by 3.

9.6.31 Set Symbol to Octal or Decimal (.SETO/.SETD)

The .SETO and .SETD directives redefine the radix of a specified numeric symbol. If the symbol has not been defined, Indirect makes an entry in the symbol table and sets the symbol to the specified radix with a value of 0. If the symbol has already been defined, Indirect resets the symbol accordingly. Indirect exits with a fatal error if the symbol was previously defined as a logical or string symbol.

Formats

.SETO   ssssss

.SETD   ssssss

Parameter

ssssss

The 1- through 6-character numeric symbol to be assigned an octal or decimal radix.

Example

```
.SETN A 10     ; Sets symbol A to 10(octal)
.SETD A        ; Defines A as a decimal-radix symbol with a value of
                ; 8(octal).
.SETO A        ; Defines A back to original radix with a value of
                ; 10(octal).
```

9.6.32 Set Symbol to String Value (.SETS)

The .SETS directive defines or changes the string value of a specified string symbol. If the symbol has not been defined, Indirect makes an entry in the symbol table and sets the symbol to the specified string value. If the symbol has been defined, Indirect resets the symbol accordingly. Indirect exits with a fatal error if the symbol was defined previously as a logical or numeric symbol.

Format

.SETS   ssssss strexp

Parameters

ssssss

The 1- through 6-character string symbol.

strexp

Any string expression. (See Section 9.4.3.)

Indirect 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 quotation marks ("constant" or pound sign (#constant#)).

You can combine a string symbol, constant, or substring with another string symbol or substring by the string concatenation operator (+) to form a string expression.

Examples

```
.SETS A "ABCDEF"
```

This directive assigns to string symbol A the string value ABCDEF.

```
.SETS STR2 "ZZZ"
```

This directive assigns to string symbol STR2 the value ZZZ.

```
.SETS S1 #123"456#
```

This directive assigns to string symbol S1 the value 123"456.

```
.SETS X STR2+"ABC"
```

This directive assigns to string symbol X the value of symbol STR2 plus ABC (that is, ZZZABC).

```
.SETS X STR2+A[1:3]
```

This directive is equivalent to the previous directive. It assigns to string symbol X the string value of STR2 plus the first three characters of string A (that is, ZZZABC).

```
.SETS MYFILE <UIC>+"MYFILE.TXT"
```

This directive assigns the string symbol MYFILE the string value of the current UIC and the string contained within the quotation marks (for example, if the current UIC is [303,23], MYFILE is assigned the string value [303,23]MYFILE.TXT).

9.6.33 Terminate Command File Processing (.STOP)

The .STOP directive immediately terminates command file processing at all levels, closes all open data files, and exits. The following message is then displayed (unless .DISABLE DISPLAY is in effect):

```
$ @ <EOF>
```

The .STOP directive allows you to optionally set the exit status for Indirect execution.

Format (brackets not part of syntax)

.STOP [value]

Parameter

value

An optional numeric expression to serve as the exit status for Indirect. If you do not specify an exit status value, the .STOP directive is identical to the logical end-of-file directive (/).

Example

```
.STOP 0
```

This directive terminates command file processing and sets the exit status for Indirect to 0 (Warning).

9.6.34 Test Symbol (.TEST)

The .TEST directive has two different functions. It tests a variable and sets various special symbols accordingly, and it does substring searches and sets the special symbol <STRLEN> accordingly.

Format 1

.TEST ssssss

Parameter

ssssss

The 1- through 6-character symbol to be tested.

The results of the test are as follows:

- If variable is a string, <SYMTYP> is set to 4 and <STRLEN> contains the length of the string. Also, the special symbols <ALPHAN> , <NUMBER> , <RAD50> , and <OCTAL> are set based on a scan of the characters of variable.
- If variable is numeric, <SYMTYP> is set to 2.
- If variable is octal, <SYMTYP> is set to 2 and <OCTAL> is set to TRUE.
- If variable is logical, <SYMTYP> is set to 0.

Format 2

.TEST string substring

Parameters

string

A string symbol or constant.

substring

A string expression.

In this case, the substring is searched for in the specified string. If the substring is present, <STRLEN> is set to the position of the starting character of the substring within the string. If substring is not present, <STRLEN> is set to 0.

If a string constant is used in string or substring, the constant must be enclosed by quotation marks ("constant") or by pound signs (#constant#).

Examples

If SUM is a string symbol, the directive statement

```
.TEST SUM
```

sets <SYMTYP> to 4 and places the number of characters represented by the symbol SUM into <STRLEN> .

The directive statements

```
.SETS MAIN "ABCDEF"
.TEST MAIN "C"
```

set <STRLEN> to 3, the position of C in the string ABCDEF.

The directive statements

```
.SETS S1 #AB"CDE#
.TEST S1 #D#
```

set <STRLEN> to 5, the position of D in the string AB″CDE.

9.6.35 Test Device (.TESTDEVICE)

The .TESTDEVICE directive allows a command file to acquire information about any device in the system. The information, including error indications, is contained in the string symbol <EXSTRI> . Each device attribute in the string is separated by a comma (which allows processing by the .PARSE and .TEST directives). The first field of the string is the full physical name of the device. The next four fields are octal representations of the device-characteristics words (U.CW1 through U.CW4 of the Unit Control Block). Additional fields contain more information about the device.

Format

.TESTDEVICE dd[nn]:

Parameter

dd[nn]:

The device about which the command file is requesting information.

The device name can be a logical name assignment that translates into a valid device specification.

The information stored in <EXSTRI> is in the following form:

```
ddnn:,xx,xx,xx,xx,atr,atr...,atr,
```

Where:

ddnn:

The physical device name for the device specified in the command line.

xx,xx,xx,xx

The four device-characteristics words in octal notation. (See the *RSX-11M-PLUS Guide to Writing an I/O Driver* for more information.)

atr

One or more of the following device attributes:

NSD	"No such device" is configured into this system.
LOD	The device driver is loaded.
UNL	The device driver is not loaded.
ONL	The device is on line.
OFL	The device is off line.
MTD	The device is a mountable volume and is mounted.
NMT	The device is not a mountable volume or is not mounted.
FOR	The device is a mountable volume and is mounted foreign.
NFO	The device is not a mountable volume or is not mounted foreign.
PUB	The device is a public device.
NPU	The device is not a public device.
ATT	The device is attached to another task.
ATU	The device is attached to this copy of Indirect.
NAT	The device is not attached.
ALO	The device is allocated to another terminal.
ALU	The device is allocated to this terminal.
NAL	The device is not allocated.

<EXSTRI> contains the value "NSD," (no such device) if the device is not present in the current system configuration.

<EXSTRI> contains the comma delimiters along with the device information so that the device information can be extracted from <EXSTRI> with the .PARSE directive.

Examples

```
.TESTDEVICE SY:
```

This directive statement acquires information about user logical device SY: and stores it in <EXSTRI> .

```
.TESTDEVICE TEMP
```

This directive acquires information about the device specified by the logical translation of TEMP.

The command file PROCESS.CMD contains the following directive statement:

```
.TESTDEVICE DATA$DISK
```

When the following command lines are executed:

```
$ ASSIGN DU2: DATA$DISK [RET]
$ @PROCESS [RET]
```

Indirect acquires information about the device DU2:.

9.6.36 Test a File (.TESTFILE)

The .TESTFILE directive determines if a specified file exists or it determines the physical device associated with a logical name (that is, it performs device translation).

If you specify a file in the command line, the results of a .TESTFILE operation are contained in the symbols <FILSPC> and <FILERR> . <FILSPC> contains the file specification (including device, directory, file name, file type, and version number) and <FILERR> contains the FCS status code resulting from the search for the file.

If you do not specify a file in the command line, Indirect performs device translation only.

Formats

.TESTFILE filespec

.TESTFILE ll:

Parameters

filespec

The file to be tested.

The parameter filespec can be a logical name assignment that translates into a vaild FCS file specification.

ll:

The logical name assigned to be translated to a physical device.

Examples

```
.TESTFILE MP:IND.MAP
```

If the file exists, this directive assigns the following values:

<FILERR>	=	1
<FILSPC>	=	DU1:[101,300]IND.MAP;4 (MP: is the logical name assigned to the physical device DU1:)

If the file does not exist, the directive assigns the following values:

<FILERR>	=	346 (230(decimal))
<FILSPC>	=	DU1:[101,300]IND.MAP;0

The following directive translates the logical name TI: into its physical device name:

```
.TESTFILE TI:
```

The directive assigns the symbol values as follows:

<FILERR>	=	1
<FILSPC>	=	TT23:.DAT;0

```
.TESTFILE TEMP
```

This directive assigns the symbol values as follows:

<FILERR>	=	1
<FILSPC>	=	file specified by the logical translation of TEMP

The command file SWAN.CMD contains the following directive statement:

```
.TESTFILE EGG
.TESTFILE SYS$TALES
```

When the following command lines are executed:

```
$ ASSIGN DU2:[UGLY]DUCKLING.DAT SYS$TALES [RET]
$ ASSIGN TALES.CMD EGG [RET]
$ @SWAN [RET]
```

Indirect tests the file DU2:[UGLY]DUCKLING.DAT and then for the file TALES.CMD.

9.6.37 Test a Partition (.TESTPARTITION)

The .TESTPARTITION directive allows a command file to acquire information about a partition in the system. The partition can be the one in which Indirect is running or any other partition. You can use the directive to verify that a partition is large enough before installing a task in it or that the partition is present before loading a special system. Indirect returns the information (in the special symbol <EXSTRI>) in the following format:

partitionname,base,size,type,

where base and size are in 64-byte blocks and type is SYS for system-controlled partitions, USR for user-controlled partitions, or NSP for an unknown partition name. If the partition is not found, Indirect returns a "No such partition" error in the form:

partitionname,,,NSP,

Format

.TESTPARTITION partitionname

Parameter

partition-name

A 1- through 6-character legal partition name. If you use the wildcard (*) instead of a partition name, Indirect assumes you are testing the same partition in which the current version of Indirect is executing.

Example

```
.TESTPARTITION GEN
;GEN,1500,2303,SYS,
```

This directive acquires information about the partition named GEN. The partition has a starting address of 150000(octal), it is 230300(octal) bytes long, and it is a system-controlled partition.

9.6.38 Translate a Logical Name Assignment (.TRANSLATE)

The .TRANSLATE directive allows a command file to expand a local or global logical name assignment. The expanded assignment is contained in the string <EXSTRI> .

Format

.TRANSLATE logical

Parameter

logical

The logical name assignment to be expanded.

<EXSTRI> contains the original string if no assignment exists for the specified logical name.

For more information on logical names, especially logical names containing colons, see the description of the ASSIGN and DEFINE command in the *Micro/RSX Guide to Advanced Programming,* which is available separately or as part of the Advanced Programmer's Kit.

Examples

```
.TRANSLATE TEMP
```

where TEMP is assigned the string DU0:[USER]LOGIN.CMD.

This directive assigns to the symbol <EXSTRI> the value DU0:[USER]LOGIN.CMD.

```
.TRANSLATE SYS$LOGIN
```

where SYS$LOGIN is assigned the string DU2:[MYDIR].

This directive assigns to the symbol <EXSTRI> the value DU2:[MYDIR].

9.6.39 Wait for a Task to Finish Execution (.WAIT)

The .WAIT directive suspends processing of an indirect command file until a particular task has terminated.

Format

.WAIT taskname

Parameter

taskname

A 1- through 6-character legal task name.

If the task name is omitted, Indirect assumes the task name applied by the last "RUN task" command. This name is specified as

TTnn

Where:

TT

The invoking terminal.

nn

The terminal number.

The .WAIT directive also sets the symbol <EXSTAT> with the exit status of the completed task.

If the specified (or default) task is not installed, Indirect ignores the .WAIT directive. The .WAIT directive performs no function if the /NOCLI switch is in effect.

Example

```
.WAIT RUN
```

This directive discontinues processing of the command file until the terminal-initiated task RUN exits.

9.6.40 Initiate Parallel Task Execution (.XQT)

Indirect usually passes a command to DCL and waits until the command's execution has completed. However, it is possible for Indirect to initiate a task and not wait for it to complete before executing the next directive. The .XQT directive allows you to start a task, to pass a command line to it, and to continue processing in parallel with the initiated task. (In DCL, you can also use the RUN /COMMAND:"command-line" command to pass command lines to another task.) The maximum number of successive .XQT directives allowed depends on the parameter specified in the build file for the Indirect task.

Format

.XQT   taskname command-line

Parameters

taskname

The name of the task (for example, RUN).

command-line

The command line to be executed.

The .XQT directive allows you to initiate parallel processing of tasks. The .WAIT directive is used to synchronize their execution.

The .WAIT directive must also be used to clear out the status block of a .XQT. If the block is not cleared out, it is never reused, which could possibly (after enough .XQTs) produce the error message, "Too many concurrent .XQTs."

9.7 Examples

The following sections contain examples showing different uses for Indirect. The longer examples are followed by detailed explanations.

9.7.1 Invoking Indirect Interactively and Displaying Symbols

```
$ @TI: RET
AT.>
```

Specifying @TI: allows you to work with Indirect interactively. When Indirect issues the AT.> prompt, you can enter directive statements, invoke command files, or display the values of special symbols. To display a symbol, use the .ENABLE SUBSTITUTION directive, and then request the symbol in the following format:

AT.> ;' <symbol> '

9.7.2 Using an Indirect Command File

A file named PRINTER.CMD contains the following command lines:

```
.ENABLE SUBSTITUTION
;'<TIME>'
PRINT LISTINGS.MEM
.EXIT
```

To execute the command file, use the following command line:

```
$ @PRINTER RET
```

9.7.3 Asking for a Device Specification

```
.;
.; This command file asks for a device specification.
.; You may enter the device name with or without a colon
.; and the unit number does not have to be entered for
.; unit 0.  The output produced is the proper device name
.; with a unit number and a colon.

.ENABLE SUBSTITUTION                                               ❶
.DISABLE LOWERCASE                                                 ❷
.ASKS DEVICE What is the device name?                              ❸
.SETN TEMPN 2                                                      ❹
.SETS TEMPS ":"                                                    ❺
.TEST DEVICE                                                       ❻
.IF TEMPN EQ <STRLEN> .SETS DEVICE DEVICE+"0"                      ❼
.IF TEMPS NE DEVICE[<STRLEN>:<STRLEN>] .SETS DEVICE DEVICE+":"    ❽
.DISABLE DISPLAY                                                   ❾
; The full device specification is 'DEVICE'                        ❿
.ENABLE DISPLAY                                                    ⓫
.EXIT                                                              ⓬
```

When you execute this command file, Indirect asks for the name of a device and then displays the complete device specification on the terminal. For example:

```
$ @DEVICE [RET]
$ * What is the device name? [S:]: du1 [RET]
 The full device specification is DU1:
$ @ <EOF>
$
```

The following commentary gives a line-by-line explanation of the command file:

❶ Substitution mode enabled.

❷ Lowercase mode disabled, which means that all input characters are converted to uppercase regardless of how they were typed in.

❸ Asks for the device name (that is, the mnemonic and unit number) and assigns it to the string symbol DEVICE.

❹ Sets numeric symbol TEMPN to the value 2, which is the number of characters for the device mnemonic.

❺ Sets string symbol TEMPS to contain a colon. The colon is a string constant, so it must be enclosed in quotation marks.

❻ Tests the symbol DEVICE (which contains the specified device name). As a result, the following special symbols are set:

<SYMTYP>	=	4 (because DEVICE is a string symbol)
<STRLEN>	=	the length of the string (the number of characters typed in response to the question)

❼ Performs a conditional test. If the value of TEMPN (2) equals the value of <STRLEN> , set DEVICE to be the current contents of DEVICE plus 0. That is, if <STRLEN> equals 2, that means the user typed in only the device mnemonic without a unit number. Therefore, the unit number of the device should be 0. DEVICE becomes dd0.

❽ Performs another conditional test. If the value of TEMPS (:) does not equal the last character of DEVICE, add a colon to DEVICE (set the string symbol DEVICE to be equal to DEVICE plus colon; DEVICE becomes ddn:).

❾ Display mode disabled, which means that Indirect displays only the text string for the .ASKS directive and suppresses @ <EOF> upon exiting.

❿ Displays this text, with the full device name substituted for 'DEVICE,' on the terminal.

⓫ Displays mode reenabled, which means that @ <EOF> will be displayed after all when Indirect exits.

⓬ Exits from the file and Indirect.

9.7.4 Asking for the Type and Unit Number of the Terminal

```
.ENABLE SUBSTITUTION            ❶
.ENABLE GLOBAL                  ❷
.TESTFILE TI:                   ❸
.SETN TYPE <TITYPE>             ❹
.SETN $TRM TYPE                 ❺
; '$TRM'                        ❻
.SETS UNIT <FILSPC>[1:4]        ❼
.SETS $TT UNIT                  ❽
; '$TT'                         ❾
```

When you execute this command file, Indirect retrieves the type and unit number of the terminal from which the file is executing, and displays this information on the terminal. For example:

```
$ @TERMINAL [RET]
$ ; 15
$ ; TT14
$ @ <EOF>
$
```

The first number in this display, 15, means that the terminal from which the file is running is a VT100. The second number in the display, 14, is the unit number for the terminal.

The following commentary gives a line-by-line explanation of the command file:

❶ Substitution mode enabled.

❷ Global symbol mode enabled, which means that symbol names that begin with a dollar sign ($) are defined as global for all levels of command files. Once such a symbol has been defined, all levels recognize it.

❸ Translates logical name TI: into its physical device name (for example, TT14:) and puts the name in the special symbol <FILSPC>. The device name is turned into a file specification, so the contents of <FILSPC> would be TT14:.DAT;0.

❹ Sets numeric symbol TYPE to the contents of special symbol <TITYPE>, which contains the octal code number for the type of terminal (for example, 15 for a VT100).

❺ Sets numeric global symbol $TRM to the contents of TYPE.

❻ Displays the value of $TRM (the octal code number for the terminal type) on the terminal.

❼ Sets string symbol UNIT to the first four characters of <FILSPC> (for example, TT14; if the unit number is less than 10, Indirect also displays the colon).

❽ Sets string global symbol $TT to the contents of UNIT (TT14).

❾ Displays the value of $TT (the unit number of the terminal).

9.7.5 Initializing and Mounting a Volume, and Copying Files to That Volume

```
        .ENABLE SUBSTITUTION                                       ❶
.GETDEV:                                                           ❷
        .ASKS DEVICE Enter device (DU1 or DU2)                     ❸
        .IF DEVICE EQ "DU0" .GOTO GETDEV                           ❹
        .ASKS DIR What directory (include square brackets)?        ❺
.INIT:                                                             ❻
        .ASK INIT Initialize device                                ❼
        .IFF INIT .GOTO COPY                                       ❽
        ALLOCATE 'DEVICE':                                         ❾
        MOUNT/FOREIGN 'DEVICE'"                                    ❿
        .ASKS LABEL What volume label?                             ⓫
        INITIALIZE 'DEVICE':'LABEL'                                ⓬
        DISMOUNT/NOUNLOAD 'DEVICE':                                ⓭
        MOUNT/NOSHAREABLE 'DEVICE':'LABEL'                         ⓮
        CREATE/DIRECTORY 'DEVICE':'DIR'                            ⓯
.COPY:                                                             ⓰
        .ASKS FILES Enter names of files (file1,file2,...)         ⓱
        COPY 'FILES' 'DEVICE':'DIR'                                ⓲
        .ASK MORE More files                                       ⓳
        .IFT MORE .GOTO COPY                                       ⓴
        .ASK LIST List directory                                   ㉑
        .IFF LIST .GOTO END                                        ㉒
        DIRECTORY 'DEVICE':'DIR'                                   ㉓
.END:                                                              ㉔
        DISMOUNT 'DEVICE':                                         ㉕
        DEALLOCATE 'DEVICE':                                       ㉖
        .EXIT                                                      ㉗
```

The following commentary gives a line-by-line explanation of the command file:

❶ Substitution mode enabled.

❷ Line for .GETDEV: label. It is a direct-access label, so it is the only element on the command line.

❸ Asks for the name of the device to which the files will be copied.

❹ Performs a conditional test. If DEVICE = DU0 (an illegal device), returns to .GETDEV: and asks the question again.

❺ Asks for the directory to which the files will be copied.

❻ Line for the .INIT: label (also a direct-access label).

❼ Asks if the device should be initialized.

❽ If the device should not be initialized, proceeds with the copy operation.

❾ Allocates the specified device.

❿ Mounts the device foreign, which is necessary for initializing a device.

⓫ Asks for the label for the volume.

⓬ Initializes the volume and gives it the specified label.

⓭ Dismounts the device without spinning it down.

⓮ Remounts the device as a private, Files–11 volume.

⓯ Creates the specified directory on the volume.

⓰ Line for the .COPY: label (also a direct-access label).

⓱ Asks for the specifications of the files to be copied.

⓲ Copies the files to the device.

⓳ Asks if there are more files to be copied.

⓴ If there are more files, returns to the .COPY: label.

㉑ If there are no more files, asks if you would like a directory of the copied files.

㉒ If you do not want a directory, goes to the end of the file (.END:).

㉓ If you do want a directory, displays the names of the copied files on the terminal.

㉔ Line for the .END: label (also a direct-access label).

㉕ Dismounts the device.

㉖ Deallocates the device.

㉗ Exits from the file and Indirect.

9.7.6 Editing, Purging, Printing, and Formatting Files

```
        .ENABLE QUIET                                     ❶
        .ENABLE SUBSTITUTION                              ❷
        .ASKS FILNAM What is the file name?               ❸
        .ASKS FILTYP What is the file type?               ❹
        EDIT 'FILNAM'.'FILTYP'                            ❺
        .ASK A Do you want to purge this file?            ❻
        .IFT A PURGE/KEEP:2 'FILNAM'.'FILTYP'             ❼
        SET FILE /TRUNCATE 'FILNAM'.'FILTYP';*            ❽
        .ASK DSR Do you want to invoke DSR?               ❾
        .IFT DSR .GOSUB PROC                              ❿
        .ASK B Do you want a listing?                     ⓫
        .IFF B .GOTO 100                                  ⓬
        .GOSUB LIST                                       ⓭
        PRINT/FORMS:'C'/COPIES:'D' 'FILNAM'.'FILTYP'      ⓮
.100:                                                     ⓯
        .EXIT                                             ⓰
.PROC:                                                    ⓱
        RNO 'FILNAM'='FILNAM'                             ⓲
        .SETS FILTYP "MEM"                                ⓳
        .ASK F Do you want to purge the .MEM files?       ⓴
        .IFT F PURGE/KEEP:2 'FILNAM'.MEM                  ㉑
        SET FILE /TRUNCATE 'FILNAM'.MEM;*                 ㉒
        .RETURN                                           ㉓
.LIST:                                                    ㉔
        .ASKN C What form number?                         ㉕
        .ASKN [::1.] D How many copies?                   ㉖
        .RETURN                                           ㉗
```

The following commentary gives a line-by-line explanation of the command file:

❶ Quiet mode enabled, which means that Indirect does not echo (display on the terminal) DCL command lines or comments. The command lines are executed normally and, if they return a message or display, those are shown on the terminal.

❷ Substitution mode enabled.

❸ Asks for the name of the file (for example, MYFILE).

❹ Asks for the type of the file (for example, CMD).

❺ Invokes EDT so that you can edit the specified file.

❻ When you are done with EDT (using the EXIT or QUIT command), asks if you want to purge the versions of the file.

❼ If you want to purge the files, DCL does this, keeping the two latest versions of the file.

❽ Truncates the files to free up blocks that are allocated to the files but not used.

❾ Asks if you want to use DIGITAL Standard Runoff (DSR) to format the file.

❿ If you do want to use DSR, Indirect goes to the subroutine for file processing (.PROC:).

⓫ After returning from the processing subroutine, asks if you want a listing of the file.

⓬ If you do not want a listing, exits from the file and Indirect.

⓭ If you do want a listing, goes to the subroutine for listing files (.LIST:).

⓮ After returning from the listing subroutine, prints the specified number of copies of the file on the designated printer.

⓯ Line for label .100: (a direct-access label).

⓰ Exits from the file and Indirect.

⓱ Line for .PROC: label (label for the processing subroutine).

⓲ DSR formats the file (which must be a .RNO file) and creates (by default) a .MEM file.

⓳ Sets string symbol FILTYP equal to type MEM.

⓴ Asks if you want to purge the .MEM files.

㉑ If you do want to purge the files, DCL does this, keeping the two latest versions of the file.

㉒ Truncates the files to free up blocks that are allocated to the files but not used.

㉓ Returns to the line after .GOSUB PROC (.ASK B Do you want a listing).

㉔ Line for .LIST: label (label for the listing subroutine).

㉕ Asks for the form number for the line printer. Sets the numeric symbol C to this value, which is used in the PRINT command line.

㉖ Asks for the number of copies to be printed (the default is 1). Sets numeric symbol D to this value, which is also used in the PRINT command line.

(27) Returns to the line after .GOSUB LIST (the PRINT command line).

9.8 Indirect Messages

When Indirect encounters an error, it displays the appropriate error message and the command line in which the error occurred. If the line contained a substitution, the line as it appeared before the substitution took place is also displayed. Indirect also closes all open data files before exiting.

Section 9.8.1 explains the information-only messages and Section 9.8.2 explains the error messages. The error messages are divided into classes, depending on the level of severity. Class 2 errors can be handled with the <ERRCTL> symbol (see Section 9.4) and class 1 errors can be handled with the .ONERR directive (see Section 9.6.21). Class 0 errors must be corrected outside of Indirect.

9.8.1 Information-Only Messages

@ < EOF>

Explanation: (Class 0) Indirect has reached the end-of-file for the outermost command file and is terminating execution.

AT.—Continuing

Explanation: Indirect is resuming execution after a .PAUSE or .DELAY directive.

AT.—Delaying

Explanation: A .DELAY directive was just executed, halting the processing of an indirect command file for a specified period of time.

AT.—Invalid answer or terminator

Explanation: In response to a question from .ASK, you entered something other than Y, N, or null, followed by a RETURN; or you did not enter a numeric value in response to an .ASKN question; or you pressed the ESCAPE key either without escape recognition enabled or as a character other than the first one following the question. The question will be repeated.

AT.—Pausing. To continue type "command taskname"

Explanation: Indirect just executed a .PAUSE directive, interrupting processing of an indirect command file to wait for user action.

AT.—Value not in range

Explanation: The response to an .ASKN or .ASKS question was not within the specified range. Indirect repeats the question.

9.8.2 Error Messages

AT.—Bad range or default specification

Explanation: (Class 1) An illegal character was specified as a range or default argument. Only numeric expressions are permitted.

AT—Command file open error

Explanation: (Class 2) The file being invoked in an @file or @file /LB:module command line cannot be found or opened.

AT.—Data file error, code x.

Explanation: (Class 1) Indirect encountered an error while processing a .OPEN, .OPENA, .CLOSE, or .DATA directive, or a data-mode access to the secondary file. See the description of <FILERR> (Section 9.4.1.2) for a definition of the numeric code x.

AT.—File already open

Explanation: (Class 1) An .OPEN or .OPENA directive specified a file that was already open.

AT.—File not found

Explanation: (Class 2) An @filename or .CHAIN directive specified an incorrect file name or nonexistent file.

AT.—File not open

Explanation: (Class 1) Indirect encountered a .DATA or .CLOSE directive that did not reference an open file.

AT.—File read error

Explanation: (Class 2) An error was detected in reading the indirect command file. This error is usually caused by records that are more than 132(decimal) bytes long.

AT.—Illegal file number

Explanation: (Class 1) The file number in an .OPEN, .OPENA, .OPENR, *.DATA, .ENABLE DATA, .READ, or .CLOSE directive is not in the range of 0 through 3.

AT.—Illegal nesting

Explanation: (Class 1) Too many Begin-End blocks have been nested in the indirect command file. The maximum nesting depth is limited to the size of the symbol table.

AT.—Initialization error, code x.

Explanation: (Class 0) Indirect failed to complete initialization when you invoked it. The following list gives the meaning of the displayed code number:

1. Unable to acquire system information such as the UIC or device name.
2. Impure area setup failed.
3. Unable to acquire task-specific information.
4. Unable to acquire terminal-type information.
5. Unable to acquire the disk name and other information about the system device (SY:).
6. Unable to allocate enough space for command and data I/O buffers. For privileged Indirect tasks, Indirect was not installed with a large enough increment value. The system manager should remove and reinstall Indirect with a larger increment or in a larger partition. For the nonprivileged Indirect task, the EXTEND TASK directive failed to return sufficient space for Indirect to allocate the buffers.
7. Initialization of allocated buffers failed.
8. Initialization of the DATA file structures failed.
9. Allocation of FCS-11 buffers for data and command lines failed.
10. Symbol table initialization failed.
11. Initialization cleanup failed.
12. Unable to obtain initial command line.

> 12. Error codes greater than 12 are returned by FMS–11 and other special purpose initialization modules.

Error number 6 is the only initialization error that you should encounter. If any other error from 1 through 12 persists, submit a Software Performance Report (SPR) with any other pertinent information.

AT.—Invalid keyword

Explanation: (Class 1) An unrecognized keyword (preceded by a period) was specified.

AT.—Label not at beginning of line

Explanation: (Class 1) The specified label does not start in the first column of the line. All labels must do so.

AT.—Logical name translation error

Explanation: (Class 1) A logical name translation directive error has occurred. Use the SHOW ASSIGNMENTS command to determine the status of your logical name assignments.

AT.—Maximum indirect file depth exceeded

Explanation: (Class 2) An attempt was made to reference an indirect command file at a nested depth greater than the maximum specified in the build file for the Indirect task.

AT.—No pool space

Explanation: (Class 2) The dynamic memory allocation has been exhausted. Either wait for more pool space to become available or use the DCL SET SYSTEM/POOL command.

AT.—Null control string

Explanation: (Class 1) The control string specified with the .PARSE directive was null (there were no characters between the quotation marks or pound sign).

AT.—Numeric under- or overflow

Explanation: (Class 2) The evaluation of a numeric expression yielded a value outside the range 0 through 177777(octal). This means that the value crossed the zero boundary from positive to negative, or from negative to positive.

AT.—Redefining a read-only symbol.

Explanation: (Class 2) An attempt was made to assign a new value to a read-only symbol. Read-only symbols cannot be overwritten.

AT.—Redefining symbol to different type ssssss

Explanation: (Class 1) An .ASK, .ASKN, .ASKS, .READ, .SETT, .SETF, +.SETL, .SETN, or .SETS directive was in an attempt to set the specified, already defined symbol to a different type. The first definition of a symbol determines its type (logical, numeric, or string); subsequent value assignments must conform to the original type.

AT.—.RETURN without .GOSUB

Explanation: (Class 1) A .RETURN directive was specified without a previous call to a subroutine (.GOSUB).

AT.—Spawn failure

Explanation: (Class 1) Indirect could not initiate the execution of a user command task.

AT.—String expression larger than 132. bytes

Explanation: (Class 2) An attempt was made to generate a string expression longer than 132(decimal) characters.

AT.—String substitution error

Explanation: (Class 1) Indirect encountered an error during a substitution operation. A probable cause for the error is either the omission of a second apostrophe or the specification of a symbol that is not defined.

AT.—Subroutine nesting too deep

Explanation: (Class 1) The maximum subroutine nesting level was exceeded. The maximum level is specified in the build file for the Indirect task.

AT.—Symbol table overflow ssssss

Explanation: (Class 2) The symbol table was full and there was no space for symbol ssssss.

AT.—Symbol type error ssssss

Explanation: (Class 1) The symbol ssssss was used out of context for its type; for example, a numeric expression referenced a logical symbol. Only symbols of the same type can be compared.

AT.—Syntax error

Explanation: (Class 2) The format of the specified command line is incorrect.

AT.—Too many concurrent .XQTs

Explanation: (Class 1) More than the maximum number of successive .XQT directives allowed by the build file for the Indirect task were issued.

AT.—Undefined label < .label:>

Explanation: (Class 1) The label <.label:> specified in a .GOTO, .GOSUB, or *.ONERR directive could not be found.

AT.—Undefined symbol ssssss

Explanation: (Class 1) The symbol ssssss was referenced, but it had not been defined.

Chapter 10

Quick Reference

This chapter lists all DCL commands in alphabetical order. Each command entry includes cross-references to the full command description in Volume 2 or to other Micro/RSX manuals. Each command entry also includes a general description of the functions of the command, as well as the full command syntax, sometimes with brief comments preceded by an exclamation point (!).

These command descriptions are intended for quick reference only. If you are not familiar with the functions of a particular command, you should read the main text referenced in this chapter. The comments are in shorthand form and are meant to serve as reminders only.

Comments include examples of input, value ranges, and cautions. Privileged commands and qualifiers are flagged. A number of commands have two qualifiers listed as synonyms. Other qualifiers perform no operation by themselves and are listed in the comments as "no-ops." These synonyms and no-ops are usually included for compatibility with DCL on VAX/VMS systems or for clarity and completeness. See the main text for more information.

For your convenience, some frequently used commands have brief forms. These short forms are not necessarily compatible with other implementations of DCL and are provided for the convenience of Micro/RSX users. They are as follows:

A	ABORT	H	HELP
B	BROADCAST	?	HELP
C	COPY	LO	LOGOUT
D	DIRECTORY	L	LINK
DEAL	DEALLOCATE	M	MACRO
DEAS	DEASSIGN	P	PRINT
E	EDIT	R	RUN
F	FORTRAN	S	SHOW

ABORT Chapter 15

ABORT forces an orderly end to a running task or to the action of a specific command. Nonprivileged users can abort any task running on TI:. Privileged users can abort any task by using the /TERMINAL qualifier.

Formats

```
ABORT[/COMMAND]  [/qualifier[s]] commandname
ABORT/TASK[/qualifier] taskname
```

Command Qualifiers

```
/COMMAND                    !Default
/[NO]POSTMORTEM
/TASK
/TERMINAL:ttnn:             !Privileged
```

ALLOCATE Chapter 12

ALLOCATE declares a specified device to be a private device. The format for the logical device name is the same as for other device names: two characters, an octal unit number, and a colon. If you omit the unit number and colon, the first available device of that class is allocated.

Format

```
ALLOCATE[/qualifier[s]] dd[nn:] [logicalname]
```

Command Qualifiers

```
/TERMINAL:ttnn:              !Privileged
/TYPE:devicetype             !RD51, RX50, RL02, and so on
```

ANALYZE/CRASH_DUMP

RSX-11M/M-PLUS and Micro/RSX Crash Dump Analyzer Reference Manual

The ANALYZE/CRASH_DUMP command helps you determine the cause of system crashes by analyzing and formatting a memory dump created by the Executive Crash Dump Module. You must have the Advanced Programmer's Kit to use this command.

Format

```
ANALYZE/CRASH_DUMP[/qualifier[s]] filespec[/qualifer[s]]
```

Command Qualifiers

```
/LIST[:listfilespec[/qualifiers]]
              /ERROR_LIMIT
              /PAGE_COUNT:n
              /PAGE_LENGTH:n
              /[NO]PRINTER
              /EXIT:n
              /LIMIT:n
              /LINES:n
              /[-]SP
/BINARY:binaryfilespec
/MEMORY_SIZE:n
/SYMBOLS:symbolfilespec
```

Filespec Qualifiers

```
/ACTIVE:(arg[,...])
        DEVICES
        TASKS
/ALL
/DEVICES
/TASKS
/BLOCK:n
/CLOCK_QUEUE
/CONTROLLERS
```

```
/DENSITY:n
/DATA_STRUCTURES:(arg[,...])
                              COMMAND_PARSER
                              DEVICE
                              PARTITION
                              STATUS
                              TASK
                              UNIT
/DUMP[:(START:n,END:n[,ADDRESS:n])]
/HEADERS
/KERNEL:(arg[,...])
              DATA:(START:n,END:n)
              INSTRUCTION:(START:n,END:n)
              REGISTERS
/PARTITION
/POOL:(START:n,END:n)
/SECONDARY_POOL[:(START:n,END:n)]
/[NO]SYSTEM
/TASKS:(arg[,...])
            DIRECTORY
            ADDRESS:(NAME:name[,START:n,END:n])
            DATA:(NAME:name[,START:n,END:n])
            INSTRUCTION:(NAME:name[,START:n,END:n])
```

ANALYZE/ERROR_LOG *Micro/RSX System Manager's Guide*

The ANALYZE/ERROR_LOG command analyzes and formats information about errors and events that occur on system hardware and generates error log reports. Most of these commands (and the reports they generate) will be most useful to DIGITAL Field Service.

Format

```
ANALYZE/ERROR_LOG   [/qualifier[s]] datafile
```

Command Qualifiers

```
/BRIEF
/COMMAND:switchstring         !Invokes predefined switch string
                DAY
                MONTH
                WEEK
                SYSTEM
/DEVICES[:devicelist]
```

```
/ENTRY:[start:end[,...])]         !Specifies error log packet numbers
/FULL
/INCLUDE[:(arg[,...])]            !Specifies the type of errors
                                  !to report
           ALL
           CONTROL
           MEMORY
           PERIPHERAL
           PROCESSOR
           SYSTEM_INFORMATION
/NODETAIL
/OUTPUT[:outputfile]              !Writes report in a file
/PREVIOUS_DAYS:n
/REGISTERS
/SERIAL_NUMBER:(arg[,...])
                  PACK:n
                  DRIVE:n
/SINCE:starttime[/THROUGH:endtime]
/STATISTICS[:(arg[,...])]         !Writes report based on
                                  !disk geometry
             ALL
             ERROR
             DISK_GEOMETRY
             HISTORY
             NONE
/THROUGH:endtime
/TODAY
/VOLUME_LABEL:volumelabel
/[NO]WIDE
/YESTERDAY
```

ANALYZE/MEDIA Chapter 13

The ANALYZE/MEDIA command determines if bad blocks exist on a disk volume and records their locations for use by backup and restore utilities and the INITIALIZE command.

Format

```
ANALYZE/MEDIA[/qualifier[s]] ddnn:
```

Command Qualifiers

```
/ALLOCATE:label               !Prompts for bad block numbers to
                              !put in BADBLOCK.SYS and to enter
                              !in the bad block descriptor file
/BAD_BLOCKS
/BAD_BLOCKS/EXERCISE:(n,m)
/BAD_BLOCKS/NOEXERCISE
/[NO]EXERCISE[:(n,m)]
/OVERRIDE
/RETRY
/SHOW
```

APPEND Chapter 11

APPEND appends to an existing sequential file records from one or more sequential files.

Format

```
APPEND[/qualifier[s]] infile[,s] outfile
```

Command Qualifiers

```
/EXCLUDE:filespec             !Filespec can include wildcards.
/NOWARNINGS                   !Suppresses error messages
/REWIND                       !Tape only. Rewinds tape before
                              !beginning
/SHARED                       !Permits others to access file
                              !while you append it
/DATE:dd-mmm-yy               !Given day only
/SINCE:dd-mmm-yy              !From given day through current day
/THROUGH:dd-mmm-yy            !From beginning through given day
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
                              !From given day through given day
/TODAY                        !Today only
```

APPEND/ERROR_LOG *Micro/RSX System Manager's Guide*

APPEND/ERROR_LOG appends the specified file to the end of the current error log file. Error Logging must be active for this qualifier to work. The default is to append the file to the current Error Log File and to keep the appended file as well.

Format

```
APPEND/ERROR_LOG   filespec[/qualifier]
```

File Qualifier

/DELETE	!Deletes error log file after !appending it

ASSIGN Chapter 13

ASSIGN equates a logical name to a physical device name, to all or part of a file specification, or to another logical name. ASSIGN checks the syntax of an equivalence name that is either a device or file specification. All references to the logical name are resolved by the operating system.

Format

```
ASSIGN[/qualifier[s]] equivalence_name logical_name
```

Command Qualifiers

/FINAL	!Privileged
/GROUP[:g]	!UIC group number
/LOCAL	!Default
/LOGIN	!Privileged
/GLOBAL	!Privileged
/SYSTEM	!Privileged; synonym for /GLOBAL
/TERMINAL:ttnn:	!Privileged
/TRANSLATION:FINAL	!Privileged; synonym for /FINAL

ASSIGN/QUEUE *Micro/RSX System Manager's Guide*

ASSIGN/QUEUE establishes a path between a queue and a processor in the Queue Manager subsystem. Privileged.

Format

```
ASSIGN/QUEUE   queuename processorname
```

ASSIGN/REDIRECT Chapter 13

ASSIGN/REDIRECT redirects output from one physical device to another. You can also redirect output from a physical device to a pseudo device or the reverse. Privileged.

Format

```
ASSIGN/REDIRECT  oldddnn: newddnn:
```

ASSIGN/TASK Chapter 15

ASSIGN/TASK reassigns an installed task's logical unit numbers (LUNs) from one physical device to another. The reassignment overrides the static LUN assignments in the task's disk image file. You cannot change the LUNs of an active task. Privileged.

Format

```
ASSIGN/TASK:taskname ddnn: lun
```

BACKUP *Micro/RSX System Manager's Guide*

BACKUP backs up and restores Files–11 volumes. It transfers files from a volume to a backup volume and retrieves files from the backup volume.

Format

```
BACKUP[/qualifier[s]] sourceddnn:[filespec[s]] destinationddnn:
```

Command Qualifiers

Group 1: Selective Backup and Restore

```
/CREATED:arg
          AFTER:(dd-mmm-yy hh:mm)
          BEFORE:(dd-mmm-yy hh:mm)
/EXCLUDE
/IMAGE:arg                    !For multivolume disk operations.
        SAVE
        RESTORE
/MODIFIED:arg
          AFTER:(dd-mmm-yy hh:mm)
          BEFORE:(dd-mmm-yy hh:mm)
/NEW_VERSION
/[NO]REPLACE
```

Group 2: Initialization

```
/ACCESSED:n                   !n is default number of FCBs per
                              !volume.
/BAD_BLOCKS:arg               !Default
             AUTOMATIC
             MANUAL
             OVERRIDE
/EXTENSION:n                  !Default n=5.
/FILE_PROTECTION:code         !Default is protection of involume.
/HEADERS:n
/INDEX:arg                    !Specifies location of INDEXF.SYS
       BEGINNING              !on volume; default is location of
       MIDDLE                 !file on involume.
       END
       n                      !At logical block n.
/INITIALIZE
/MAXIMUM_FILES:n
/SAVE_SET:name                !Default is name of volume being
                              !backed up.
/WINDOWS:n                    !Default is number of mapping
                              !pointers on involume.
```

Group 3: Tape and Disk Control

```
/APPEND                       !May need /REWIND; see main text.
/DENSITY:arg
         800
         1600                 !Default density is 800 bpi.
/ERROR_LIMIT:n                !Default n=25.
/LABEL:TAPE:fileset-ID
/LENGTH:n                     !Usable length of output tape in
                              !decimal feet; default n = physical
                              !length of the output tape.
/REWIND                       !Rewind first tape of tape set
                              !before executing the command line;
                              !may use with /APPEND.
```

Group 4: Verification

```
/COMPARE
/VERIFY
```

Group 5: Display

```
/LIST
```

/[NO]LOG	!/LOG goes to TI:; /NOLOG is !default.

Group 6: Disk Processing

/APPEND	!May need /IMAGE; see main text
/DIRECTORY	!Can only be used with !/NOINITIALIZE.
/NOINITIALIZE	
/LABEL:arg	!See also /LABEL:TAPE:volumelabel, !Gp 3.
INPUT:volumelabel	
[OUTPUT:]volumelabel	!/LABEL:OUTPUT is default; if the !only volumelabel in command line !is outvolume, /LABEL:volumelabel !will do.
/MOUNTED	
/[NO]PRESERVE	!/PRESERVE is default.

BROADCAST Chapter 11

BROADCAST displays a specified message at one or more terminals.

Formats

```
BROADCAST   ttnn: message
BROADCAST   username message
BROADCAST   @indirectspec
BROADCAST[/qualifier] message
```

Command Qualifiers

/ALL	!Privileged
/LOGGED_IN	!Privileged

CANCEL Chapter 15

CANEL eliminates entries from the clock queue. CANCEL does not affect a currently executing task, just the pending entries in the clock queue.

Format

```
CANCEL   taskname
```

CONTINUE Chapter 15

CONTINUE resumes execution of a previously suspended task. The task name defaults to terminal TTnn:.

Format

CONTINUE[/qualifier] [taskname]

Command Qualifier

/TERMINAL:ttnn:	!Privileged

CONVERT Chapter 12

CONVERT invokes the RMSCNV utility, which moves records from one file to another. RMSCNV reads records from an input file and writes them to an output file. The action of RMSCNV depends on the organization (sequential, relative, or indexed) of the two files and on the qualifiers you include with the CONVERT command.

Format

CONVERT[/qualifier[s]] infile outfile

Command Qualifiers

/[NO]APPEND	!/APPEND conflicts with /REPLACE.
/BLOCK_SIZE:n	!Magtape block size:18 <=n <=8192; !default=512.
/[NO]FIXED_CONTROL	!/NO is default.
/[NO]IDENTIFICATION	!Prints RMS-11 version number; /NO !is default; no filespec required.
/INDEXED	!Outfile is indexed; see main text.
/KEY[:n]	!0 <=n <=9; default=1.
/[NO]LOG_FILE[:filespec]	!/NO is default; no filespec logs !on TI:
/[NO]MASS_INSERT	
/MERGE	!Both files must have same !organization.
/[NO]PAD[:[#]arg]	!Pad infile records to outfile !length; default pad character is !blank (040).
/RELATIVE	!Outfile is relative.
/[NO]REPLACE	!/REPLACE conflicts with /APPEND.
/SEQUENTIAL	!Outfile is sequential.
/[NO]TRUNCATE	!/NO is default.

COPY Chapter 12

COPY copies files. Unless specified otherwise, COPY preserves the file organization of the input file, that is, indexed files are copied as indexed files, and so forth. See also the CONVERT command. If you intend to do multiple file copies, see the COPY command description.

Format

```
COPY   infile[s] outfile
```

Command Qualifiers

```
/ALLOCATION:n                  !Specifies n blocks of contiguous
                               !space
/BLOCK_SIZE:N                  !Defines block size for outfile on
                               !magtape; n is octal unless
                               !terminated with decimal point; no
                               !effect on infile.
/[NO]CONTIGUOUS                !Specifies contiguous outfile.
/EXCLUDE:filespec              !Filespec can include wildcards.
/NONEW_VERSION                 !Suppresses automatic increment
                               !of version numbers.
/NOWARNINGS                    !Suppresses error messages.
/OWN                           !Makes outfile UIC owner of copy.
/OVERLAY                       !Infile written over outfile.
/PRESERVE_DATE                 !Output file takes creation date of
                               !input file.
/REPLACE                       !No magtape.
/REWIND                        !Magtape only. Rewinds tape before
                               !starting operation.
/SHARED                        !Permits others to access file
                               !while you copy it.
/DATE:dd-mmm-yy                !Given day only.
/SINCE:dd-mmm-yy               !From given day through current
                               !day.
/THROUGH:dd-mmm-yy             !From beginning through given day.
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
                               !From given day through given day.
/TODAY                         !Today only.
```

CREATE Chapter 12

CREATE creates a sequential file in a directory on a file-structured device. After you issue the command, you can immediately enter text. To end data entry and close the file, press CTRL/Z. See also CREATE/DIRECTORY.

Format

```
CREATE   filespec
```

CREATE/CFL *Micro/RSX System Manager's Guide*

The CREATE/CFL command invokes the Control File Language compiler (CFL) and allows you to create intermediate form modules (IFORM) to support non-DIGITAL devices in the Error Logging System.

Format

```
CREATE/CFL[/qualifier(s)] source_file [symbol_file]
```

Command Qualifiers

```
/[NO]INTERMEDIATE_FORM[:iform_file]
/[NO]LIST[:listfile]
/[NO]OPTION
/[NO]SYMBOL_FILE[:symbolfile]
```

CREATE/DIRECTORY Chapter 12

CREATE/DIRECTORY creates a directory on a Files-11 volume and enters its name in the volume's Master File Directory (MFD). Nonprivileged users can only create directories on mounted volumes on their own private (allocated) devices.

Format

```
CREATE/DIRECTORY[/qualifier] [ddnn:][directory]
```

Command Qualifiers

```
/ALLOCATION:n              !Entries for n files.
/LABEL:volumelabel         !Compare with volume label
/NOWARNINGS                !Suppresses error messages
/OWNER_UIC                 !Owner of directory; use if
                           !different from creator
/PROTECTION:code
```

DEALLOCATE Chapter 13

DEALLOCATE counteracts ALLOCATE. It frees a private device for access by others. To deallocate a device, the device must not be mounted.

Format

```
DEALLOCATE[/qualifier] ddnn:
```

Command Qualifiers

/ALL	!Frees all devices allocated by !TI:; do not specify ddnn:.
/DEVICE	!Non-operational
/TERMINAL:ttnn:	!Privileged

DEASSIGN Chapter 13

DEASSIGN deletes logical name assignments. DEASSIGN counteracts both the ASSIGN and DEFINE commands.

Format

```
DEASSIGN[/qualifier[s]] logical_name
```

Command Qualifiers

/ALL	!Combine with any other qualifier
/GROUP[:g]	!UIC group number
/LOCAL	!Default
/LOGIN	!Privileged
/GLOBAL	!Privileged
/SYSTEM	!Synonym for /GLOBAL
/TERMINAL:ttnn:	!Default is TI:; otherwise !privileged

DEASSIGN/QUEUE *Micro/RSX System Manager's Guide*

DEASSIGN/QUEUE counteracts ASSIGN/QUEUE. It is used to eliminate the path from a queue to a processor in the Queue Manager subsystem. Privileged.

Format

```
DEASSIGN/QUEUE  queuename processorname
```

DEFINE *Micro/RSX Guide to Advanced Programming*

DEFINE equates a logical name to a physical device name, to all or part of a file specification, or to another logical name. All references to the logical name are resolved by the operating system.

Unlike ASSIGN, DEFINE does not check the syntax of an equivalence name that is either a device or file specification.

Please note that the DEFINE command primarily benefits the applications programmer.

Format

```
DEFINE[/qualifier[s]] equivalence_name logical_name
```

Command Qualifiers

```
/GROUP[:g]
/GLOBAL                    !Privileged
/LOCAL                     !Default
/LOGIN                     !Privileged
/SYSTEM                    !Privileged; synonym for /GLOBAL
/TERMINAL:ttnn:            !Privileged
/TRANSLATION:FINAL         !Privileged
```

DELETE Chapter 12

DELETE deletes specified versions of files and releases the storage space the files occupy. See also other forms of the DELETE command described in the following entries.

Format

```
DELETE[/qualifier[s]]
```

Command Qualifiers

```
/EXCLUDE:filespec          !Filespec can include wildcards.
/[NO]LOG                   !Lists deleted files on TI:
/NOWARNINGS                !Suppresses error messages
/[NO]QUERY                 !Queries before deleting;
                           !/NO is default.
/DATE:dd-mmm-yy            !Given day only
/SINCE:dd-mmm-yy           !From given day through current day
/THROUGH:dd-mmm-yy         !From beginning through given day
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
                           !From given day through given day
```

```
/TODAY                        !Today only
```

DELETE/DIRECTORY   Chapter 12

DELETE/DIRECTORY deletes a directory on a Files–11 volume and removes its name from the volume's Master File Directory (MFD). Nonprivileged users can only delete directories on mounted volumes on their own private (allocated) device.

Format

```
DELETE/DIRECTORY  [ddnn:][directory]
```

DELETE/ENTRY   Chapter 12

DELETE/ENTRY deletes QMG jobs by entry number (n).

Format

```
DELETE/ENTRY:n[/qualifier]
```

Command Qualifiers

/FILE_POSITION:n

DELETE/JOB   Chapter 12

DELETE/JOB deletes QMG jobs by queuename and jobname.

Format

```
DELETE/JOB[/qualifier] queuename [[uic]]jobname
```

Command Qualifier

/FILE_POSITION:n

DELETE/processortype   *Micro/RSX System Manager's Guide*

DELETE/processortype deletes print processors, output despoolers, or batch processors from the Queue Manager subsystem by processor name or device name. This command also sets the device unspooled.

Format

DELETE/processortype processorname

Processortypes

APPLICATIONS_PROCESSOR	
BATCH_PROCESSOR	
CARD_READER	!INPUT is synonym.
DEVICE	!PRINTER is synonym.
INPUT	!CARD_READER is synonym.
PRINTER	!DEVICE is synonym.
PROCESSOR	

DELETE/QUEUE *Micro/RSX System Manager's Guide*

Privileged. DELETE/QUEUE deletes queues in the Queue Manager subsystem by name. When you specify DELETE/QUEUE, you must also specify the /ERASE qualifier. See DELETE/JOB and DELETE/ENTRY to delete jobs from queues.

Format

DIRECTORY Chapter 12

DIRECTORY displays information on files in directories.

Format

DIRECTORY[/format-qual][/destination-qual] [other-qual[s]] [filespec[s]]

Command Qualifiers

Format Qualifiers

/ATTRIBUTES	!RMS–11 attributes
/BRIEF	
/FREE [ddnn:]	!Free blocks on volume; default is !SY:.
/FULL	
/SUMMARY	!Blocks used and allocated only

Destination Qualifiers

/OUTPUT[:filespec]	!Names output file. TI: is default.
/PRINTER	!Output to LP0:

Other Qualifiers

	!Not used with /FREE or !/ATTRIBUTES
/DATE:dd-mmm-yy	!Given day only
/SINCE:dd-mmm-yy	!From given day through current day
/THROUGH:dd-mmm-yy	!From beginning through given day
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy	!From given day through given day
/TODAY	!Today only
/EXCLUDE:filespec	!Filespec can include wildcards.
/NOWARNINGS	!Suppresses error messages
/REWIND	!Magtape only; rewinds tape before !executing command

DISMOUNT Chapter 13

DISMOUNT marks the volume mounted on the specified device as logically off line and disconnected from the file system. If you specify the volume label, the system verifies the label name to ensure that you are dismounting the correct disk.

Format

```
DISMOUNT   ddnn: [label]
```

Command Qualifiers

/ALL	!Dismount all devices mounted by !user.
/PUBLIC	!Privileged; dismounts all users !from volume.
/SAVE	!Privileged; disk keeps spinning; !privileged tasks can access.
/SYSTEM	!Synonym for /PUBLIC.
/TERMINAL:ttnn:	!Privileged; dismounts volumes !mounted from another terminal.
/[NO]UNLOAD	!Affects DB:, DM:, and DU: devices; !see main text.

EDIT[/EDT] Chapter 4

EDIT or EDIT/EDT invokes EDT, the standard DIGITAL editor.

Format

```
EDIT[/EDT][/qualifier[s]] filespec
```

Command Qualifiers

```
/[NO]COMMAND[:filespec]      !Default is /COMMAND:EDTINI.EDT.
/[NO]CREATE
/[NO]JOURNAL[:filespec]
/[NO]OUTPUT[:filespec]
/[NO]READ_ONLY               !Default is /NOREAD_ONLY.
/[NO]RECOVER                 !Default is /NORECOVER.
```

FIX Chapter 15

FIX causes an installed task or region to be loaded and locked into memory. The taskname parameter can be the name of either the task or the region. Privileged.

Format

```
FIX[/qualifier] taskname
```

Command Qualifiers

```
/READONLY_SEGMENT            !Read-only segment of multiuser
                             !task
/REGION                      !Common region
```

HELP Chapter 11

Help displays information about your system. Help files for DCL and some utilities are supplied with the system. Your system may also have help for an alternate CLI or local, group, or other special help. See main text.

Format

HELP[/qualifier[s]] [parameter 1 ...parameter9]

Command Qualifiers

/CLI:cliname	!For alternate CLIs.
/DCL	!DCL help; default for DCL !terminals.
/FILE:filespec	!Names file containing help text.
/filename	!LB:[1,2] filename.HLP
/GROUP	!Help file is in [g,1]; g is your !group.

HOLD/ENTRY Chapter 12

HOLD/ENTRY holds a QMG job in its queue by entry number (n).

Format

HOLD/ENTRY:n

HOLD/JOB Chapter 12

HOLD/JOB holds a QMG job in its queue by queuename and jobname.

Format

HOLD/JOB queuename [[uic]]jobname

INITIALIZE Chapter 13

INITIALIZE produces a volume in Files–11 format. See also INITIALIZE /UPDATE. You must mount the volume /FOREIGN. Nonprivileged users must allocate the device before it can be initialize. See also other forms of the INITIALIZE command described in the following entries.

Format

INITIALIZE[/qualifier[s]] ddnn: volumelabel

Command Qualifiers

/ACCESSED:n	!Number of directories accessed !simultaneously.

/BAD_BLOCKS:arg
    AUTOMATIC
    (AUTOMATIC,MANUAL)

```
                   MANUAL
                   NOAUTOMATIC
                   OVERRIDE
                   (OVERRIDE,MANUAL)
/DENSITY:arg
               800
               1600
               HIGH
               LOW
/EXTENSION:n                        !Extend full files by n blocks;
                                    !default n=5.
/FILE_PROTECTION:(code)             !Default protection for files
                                    !on volume.
/HEADERS:n
/INDEX:arg                          !Locates index file on volume.
          BEGINNING                 !Default for tapes and DECtapes.
          MIDDLE                    !Default for disks.
          END
          n                         !At logical block n.
/LABEL:VOLUME_ACCESSIBILITY:"c"
                                    !Magtape only: limits access.
                                    !c can be A-Z, 0-9,
                                    !and ! " % ' ( ) + , - . / :
                                    != & >  < _ ? or ;
                                    !Default is blank.
/MAXIMUM_FILES:n
/OWNER:[uic]
/PROFESSIONAL                       !Initialize disk for Professional
                                    !300 series.
/PROTECTION:(code)                  !Default protection for volume.
/[NO]SHOW                           !Display volume information on TI:;
                                    !/NO is default.
/WINDOWS:n                          !Mapping pointers to file windows;
                                    !default n=7.
```

INITIALIZE/processortype (Input) *Micro/RSX System Manager's Guide*

This form of the INITIALIZE command creates, names, and starts an input spooler or card-reader processor. Privileged. See next entry to initialize an output despooler.

Format

INITIALIZE/processortype processorname[/qualifier[s]]

Processor Types

CARD_READER	!INPUT is synonym.
INPUT	!CARD_READER is synonym.

Command Qualifiers

/BATCH_QUEUE:queuename
/CONSOLE:ddnn:
/PRINTER_QUEUE:queuename

INITIALIZE/processortype (Output)

Micro/RSX System Manager's Guide

This form of the INITIALIZE command creates, names, and starts an output despooler or batch processor. Privileged. See previous entry to initialize an input spooler.

Format

INITIALIZE/processortype processorname[/qualifier[s]]

Processor Types

APPLICATIONS_PROCESSOR
BATCH_PROCESSOR
DEVICE
PRINTER
PROCESSOR

Command Qualifiers

/FLAG_PAGE:n	!n can be 0–2; default n=1.
/FORMS:n	!n can be 0–255; default n=0.
/[NO]SHAREABLE	!Default is NOSHAREABLE.
/[NO]LOWERCASE	!Default is LOWERCASE.
/[NO]UPPERCASE	!Default is NOUPPERCASE.

INITIALIZE/QUEUE *Micro/RSX System Manager's Guide*

INITIALIZE/QUEUE creates, names, and starts a queue in the Queue Manager subsystem.

Format

```
   INITIALIZE/QUEUE   queuename[/qualifier]
```

Command Qualifiers

```
/BATCH
/PRINTER                 !Default
/NOWARNINGS
```

INITIALIZE/UPDATE Chapter 13

INITIALIZE/UPDATE invokes the HOME utility to alter values in the volume home block without affecting the other data on the volume. INITIALIZE/UPDATE is only for disks and DECtapes in Files–11 format.

You must mount the volume /FOREIGN.

Format

```
   INITIALIZE/UPDATE[/qualifier[s]] ddnn: volumelabel
```

Command Qualifiers

```
/ACCESSED:n
/DENSITY:arg
          800
          1600
/EXTENSION:n             !Extend full files by n blocks.
/FILE_PROTECTION:code    !Default protection for files on
                         !volume.
/LABEL:newvolumelabel    !Changes volume label.
/MAXIMUM_FILES:n         !Maximum number of files on volume.
/OWNER:[uic]
/PROFESSIONAL            !Initialize disk for Professional
                         !300 series.
/PROTECTION:code         !Protection for volume.
/[NO]SHOW
/WINDOWS:n               !Mapping pointers to file windows;
                         !default n=7.
```

INSTALL Chapter 15

INSTALL includes a task in the System Task Directory, thus making it known to the system. Privileged.

Format

```
INSTALL[/qualifer[s]] [$]filespec
```

Command Qualifiers

```
/[NO]CHECKPOINT
/COMMAND:"taskcommand"
/EXTENSION:n                   !n (octal) additional words of
                               !address space.
/[NO]INTERPRETER               !Installing a CLI? Default is NO.
/MULTIUSER_PARTITION:parname
                               !Install read-only portion.
/PARTITION:parname
/[NO]POSTMORTEM
/PRIORITY:n                    !0-250.
/[NO]READONLY_COMMON           !Install common as read-only.
                               !Default is NO.
/[NO]RESIDENT_HEADER           !/NORESIDENT_HEADER is
                               !external header;/RESIDENT is
                               !no external header.
/[NO]SLAVE                     !/NOSLAVE is default.
/TASK_NAME:taskname            !1-6 characters.
/TRANSLATION_ROUTINE:n
/UIC:[uic]
/[NO]WRITEBACK
```

LIBRARY Chapter 14

LIBRARY creates and maintains user-written library files. The command has eight functions, each listed here as a separate command. See main text for more details on all functions and qualifers.

Format

```
LIBRARY[/operation][/qualifer[s]]

LIBRARY  @filespec
```

LIBRARY/COMPRESS Chapter 14

LIBRARY/COMPRESS physically deletes modules that have been logically deleted through LIBRARY/DELETE. You can rename the resulting compressed library. You can also use this command to copy a library and rename it.

Formats

```
LIBRARY/COMPRESS[:(arg[,s])] libspec [newlibspec]
```

Arguments

GLOBAL:n	!Entry-point table entries.
MODULES:n	!Module-name table entries.
BLOCKS:n	!Size in 256-word blocks.

LIBRARY/CREATE Chapter 14

LIBRARY/CREATE creates a library and optionally inserts one or more modules into it.

Formats

```
LIBRARY/CREATE[:(arg[,s])][/qualifier[s]] libspec [infilespec[s]]
```

Arguments

GLOBAL:n	!Entry-point table entries.
MODULES:n	!Module-name table entries.
BLOCKS:n	!Size in 256-word blocks.

Command Qualifiers

/[NO]GLOBALS	!Include globals in entry-point !table.
/MACRO	!Identifies macro library.
/OBJECT	!Default; identifies object !library.
/SELECTIVE_SEARCH	!Object modules only.
/SQUEEZE	
/UNIVERSAL	!Identifies universal library.

LIBRARY/DELETE Chapter 14

LIBRARY/DELETE deletes object modules from a library. You can delete as many as 15 modules with a single command. See LIBRARY/REMOVE for removing global symbols (entry points) from a library.

Format

```
LIBRARY/DELETE   libspec module[,module[,s]
```

LIBRARY/EXTRACT Chapter 14

LIBRARY/EXTRACT reads one or more modules from a library and writes them to a specified output file. You can extract as many as eight modules with a single command. If you extract more than one module, the modules are concatenated in the output file.

Format

```
LIBRARY/EXTRACT[/qualifier] libspec module[,s]
```

Command Qualifier

```
/OUTPUT[:filespec]            !Default output file is TI:; name a
                              !file when extracting object
                              !modules.
```

LIBRARY/INSERT Chapter 14

LIBRARY/INSERT inserts modules from one or more files into a library. You can insert any number of files with a single command.

Format

```
LIBRARY/INSERT   libspec filespec[s]
```

Command Qualifiers

```
/[NO]GLOBALS                  !Include globals in entry-point
                              !table.
/SELECTIVE_SEARCH
/SQUEEZE
```

LIBRARY/LIST Chapter 14

LIBRARY/LIST lists on your terminal or in an output file the names of all modules in a library.

Format

```
LIBRARY/LIST[:filespec] libspec
```

Command Qualifiers

```
/BRIEF
/FULL
/[NO]NAMES                  !Names plus global entry points.
```

LIBRARY/REMOVE Chapter 14

LIBRARY/REMOVE removes global symbols (entry points) from a library. You can remove as many as 15 global symbols with a single command. See LIBRARY/DELETE for deleting object modules from a library.

Format

```
LIBRARY/REMOVE   libspec global[,global[,s]]
```

LIBRARY/REPLACE Chapter 14

LIBRARY/REPLACE replaces a module in a library with a new module of the same name and deletes the old module.

Format

```
LIBRARY/REPLACElibspec filespec[s]
```

Command Qualifiers

```
/[NO]GLOBALS                !Include globals in entry-point
                            !table.
/SELECTIVE_SEARCH
/SQUEEZE
```

LINK Chapter 14

LINK invokes the Task Builder, which links object modules and routines from user and system libraries to form an executable task.

Format

```
LINK[/qualifier[s]] filespec[/qualifier[s]][,filespec[s]]
```

Command Qualifiers

```
/ANCILLARY_PROCESSOR[:n]  !Task is ACP; n is 0,4, or 5.
/[NO]CHECKPOINT:arg
        SYSTEM            !Checkpoints to [0,0]CORIMG.SYS.
        TASK              !Checkpoints to task image file.
/CODE:(arg[,s])
     CLI                  !CLI task.
     DATA_SPACE
     EAE                  !Extended arithmetic element.
     FAST_MAP             !Fast Mapping.
     FPP                  !Floating-point processor.
     PIC                  !Position-independent code.
     POSITION_INDEPENDENT
                          !Synonym for PIC.
/COMPATIBLE
/[NO]CROSS_REFERENCE
/[NO]DEBUG[:filespec]     !Default is ODT.
/[NO]EXECUTABLE:filespec  !Names task file. Synonym for
                          !/TASK.
/ERROR_LIMIT:n            !Stop task build after n errors.
/[NO]EXTERNAL
/FAST                     !Fast TKB.
/FULL_SEARCH
/[NO]HEADER
/[NO]IO_PAGE
/LONG                     !Long map.
/MAP[:filespec]           !Default is /MAP:TI:.
/[NO]MEMORY_MANAGEMENT[:n]
                          !n for unmapped systems; see main
                          !text; default is /MEM; n is 28
                          !(default) or 30.
/OPTIONS[:filespec]       !File contains options. Otherwise
                          !you will be prompted.

/OVERLAY_DESCRIPTION
/POSTMORTEM
```

```
/[NO]PRINT                      !Print map?
/[NO]PRIVILEGED[:n]             !Default is /NOPRIVILEGED.
/[NO]RECEIVE
/[NO]RESIDENT_OVERLAYS
/SAVE                           !Saves indirect file.
/[NO]SEGREGATE                  !/NOSEG is default.
/SEQUENTIAL
/SHAREABLE[:arg]                !Multiuser.
          COMMON
          LIBRARY
          TASK                  !Default.
/SLAVE
/SLOW
/SYMBOL_TABLE[:filespec]        !Output .STB file.
/[NO]SYSTEM_LIBRARY_DISPLAY
                                !Default is /NOSYS.
/[NO]TASK[:filespec]            !Names task image file;
                                !/EXECUTABLE
                                !is synonym; names 1-6 characters;
                                !/NOTASK means no task built.
/TKB                            !Default.
/TRACE
/[NO]WARNINGS                   !/NO suppresses diagnostic
                                !messages; /WARNINGS is default.
/[NO]WIDE                       !Wide map.
```

File Qualifiers

```
/[NO]CONCATENATE
/DEFAULT_LIBRARY                !File to replace [001001]SYSLIB.OLB
/[NO]GLOBALS                    !Default is /GLOBALS; includes
                                !global symbols in map.
/LIBRARY                        !File is object module library.
/INCLUDE:(module1[:...:modulen])
                                !File is object module library;
                                !include named modules in task
                                !image.
/OVERLAY_DESCRIPTION            !File is .ODL; also a command
                                !qualifier.
/SELECTIVE_SEARCH               !Also a command qualifier.
```

LOGIN Chapter 11

LOGIN (or HELLO) grants access to a multiuser protection system and establishes your privileges as a system user.

Format

```
LOGIN   userid/password
```

LOGOUT Chapter 11

LOGOUT counteracts LOGIN. LOGOUT also aborts any nonprivileged tasks running from the terminal and also dismounts any volumes and deallocates any private devices allocated from the terminal.

Format

```
LOGOUT[/qualifier]
```

Command Qualifier

```
/[NO]HOLD                  !Holds remote line after logout;
                           !/NO is default.
```

MESSAGE/ERROR_LOG *Micro/RSX System Manager's Guide*

MESSAGE/ERROR_LOG inserts text into the error log file. The text appears in reports produced by the ANALYZE/ERROR_LOG command. The message can be any text string up to 79 characters long.

Format

```
MESSAGE/ERROR_LOG   messagetext
```

MOUNT Chapter 13

MOUNT declares a volume to be logically known to the system, on line, and available for use. Some qualifiers can be used with any MOUNT command; some are limited to mounting disks (and other random-addressable devices); and others are limited to mounting magnetic tapes.

Format for Disks and Other Random-Addressable Devices

```
MOUNT[/qualifier[s]] ddnn: volumelabel
```

Format for Magnetic Tapes

```
MOUNT[/qualifier[s]] ddnn:[,ddnn:...] fileset-ID
```

Command Qualifiers for Both Disks and Tapes

/DEFAULT:arg	!Sets defaults for dismount; see !main text.
SAVE	
NOUNLOAD	
UNLOAD	
/FILE_PROTECTION:(code)	!Protection for files created !during mount.
/FOREIGN	
/OVERRIDE:IDENTIFICATION	!Privileged; no label needed.
/PARAMETERS:"user parameters"	!Quotes are required syntax.
/PROCESSOR:arg	!Privileged; name ACP for volume.
acpname	
UNIQUE	
/PROTECTION:(code)	!Volume protection during mount.
/PUBLIC	!Privileged; deallocates; sets !public.
/[NO]SHAREABLE	
/[NO]SHOW	!Displays volume information on !TI:.
/SYSTEM	!Synonym for /PUBLIC.
/[NO]WAIT	!Default is /NOWAIT.
/[NO]WRITE	

Command Qualifiers for Files-11 Devices

/ACCESSED:n	!n is number of File Control !Blocks.
/EXTENSION:n	!Extends full files by n blocks.
/OWNER:[uic]	!Coordinates with file and volume !protection.
/UNLOCK	!Main use is with VFY.
/WINDOW:n	

Command Qualifiers for ANSI and Unlabeled Tapes

See main text for these qualifiers.

/BLOCK_SIZE:n

```
/CARRIAGE_CONTROL:arg
                    FORTRAN
                    LIST
                    NONE
/DENSITY:arg
          800
         1600
/[NO]HDR3
/[NO]LABEL
/OVERRIDE:(arg,[,s])
          ACCESSIBILITY
          EXPIRATION_DATE
          IDENTIFICATION
          SET_IDENTIFICATION
/RECORD_SIZE:n
/TRANSLATE:arg
           EBCDIC
           NONE
           UT1
           UT2
           UT3
/VOLUME_IDENTIFICATION:(volume-ID[,volume-ID[,s]])
```

PRINT Chapter 12

PRINT queues files for printing on a line printer. PRINT can also queue jobs for other output devices.

Format

```
PRINT[/qualifier[s]] filespec[/qualifier[s]][,filespec[,s]]
```

Command Qualifiers

```
/AFTER:TOMORROW
/AFTER:(dd-mmm-yy hh:mm)
/COPIES:n                  !Override on filespec.
/[NO]DELETE                !Override on filespec.
/DEVICE:ddnn:
/[NO]FLAG_PAGE             !Flag page on each file; default is
                           !/NOFLAG_PAGE.
/FORMS:n                   !n can be 0–256; default n=0.
/[NO]HOLD                  !Default is /NOHOLD; same effect as
                           !HOLD command.
```

/JOB_COUNT	
/[NO]JOB_PAGE	!Flagpage on job; default !/JOB_PAGE.
/LENGTH:n	!Page length.
/[NO]LOWERCASE	
/NAME:jobname	!1–9 characters.
/[NO]TRANSFER	!Override on filespec.
/PAGE_COUNT:n	!Limits pages in job.
/PRIORITY:n	!n is 1–150 nonprivileged; through !250 privileged; default n=50
/QUEUE:queuename	
/[NO]RESTART	
/[NO]UPPERCASE	
/[NO]WIDE	

File Qualifiers

/COPIES:n
/[NO]DELETE
/[NO]TRANSFER

PURGE Chapter 12

PURGE deletes all but the latest versions of files and releases the storage space the deleted files occupy.

Format

PURGE[/qualifier[s]] filespec[s]

Command Qualifiers

/EXCLUDE:filespec	!Filespec can include wildcards.
/KEEP:n	
/[NO]LOG	!Lists files on TI: as deleted.
/NOWARNINGS	!Suppresses error messages.
/DATE:dd-mmm-yy	!Given day only.
/SINCE:dd-mmm-yy	!From given day through current !day.
/THROUGH:dd-mmm-yy	!From beginning through given day.
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy.	!From given day through given day.
/TODAY	!Today only.

RELEASE/ENTRY Chapter 12

RELEASE/ENTRY releases by entry number (n) a print or batch job that has been held in its queue.

Format

```
RELEASE/ENTRY:n
```

RELEASE/JOB Chapter 12

RELEASE/JOB releases by queuename and jobname a print or batch job that has been held in its queue.

Format

```
RELEASE/JOB  queuename [[uic]]jobname
```

REMOVE Chapter 15

REMOVE counteracts INSTALL. REMOVE takes a task name out of the System Task Directory. Privileged.

Format

```
REMOVE[/qualifier] taskname
```

Command Qualifier

```
/REGION                     !Takes name of region out of Common
                            !Block Directory and partition
                            !list.
/TRANSLATION_ROUTINE:n      ! Removes an ACD
```

RENAME Chapter 12

RENAME changes the name, type, or version number of an existing file. Note that you can specify wildcards in any the file specification fields, but you should be careful when using them so that you do not specify the wrong file. You should also be careful with version numbers.

Format

RENAME[/qualifier[s]] oldfilespec newfilespec

Command Qualifiers

```
/EXCLUDE:filespec             !Filespec can include wildcards.
/NOWARNINGS                   !Suppresses error messages
/DATE:dd-mmm-yy               !Given day only
/SINCE:dd-mmm-yy              !From given day through current day
/THROUGH:dd-mmm-yy            !From beginning through given day
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
                              !From given day through given day
/TODAY                        !Today only
```

REQUEST Chapter 11

REQUEST sends a message to the operator's console (CO:). You can optionally place quotation marks (") around the message.

Format

REQUEST message

RUN Installed Task Chapter 15

RUN initiates the execution of installed tasks. Privileged users can use RUN to initiate the execution of installed tasks on a schedule by creating entries in the system clock queue.

Format

RUN[/qualifier[s]] taskname

Command Qualifiers

```
/DELAY:nu                     !n is the number of units and u is
                              !the time unit: T,ticks; S,seconds;
                              !M,minutes; H,hours. Privileged.
/INTERVAL:nu                  !Privileged.
/SCHEDULE:hh:mm:ss            !Privileged.
/STATUS:arg
        COMMAND               !Return status from RUN command;
                              !default. Nonprivileged.
        TASK                  !Return status from task being run;
                              !see main text on both these
                              !arguments. Nonprivileged.
```

```
/SYNCHRONIZE:u                 !Synchronize on next T,S,M, or H.
                               !Privileged.
/UIC:[uic]                     !Privileged.
```

RUN Uninstalled Task   Chapter 15

When used to run an uninstalled task (from a task image file), RUN is a combination command, encompassing INSTALL, RUN, and REMOVE. See main text.

Format

```
RUN[/qualifier[s]] [$]filespec
```

Command Qualifiers

```
/[NO]CHECKPOINT
/COMMAND:"taskcommand"         !See main text.
/EXTENSION:n                   !n (octal) additional words of
                               !address space.
/PARTITION:parname             !Privileged
/[NO]POSTMORTEM
/PRIORITY:n                    !Privileged
/STATUS:arg
        TASK                   !Return status from RUN command;
                               !the default
        COMMAND                !Return status from task being run;
                               !see main text on both these
                               !arguments. /STATUS can be
                               !used separately or with /UIC
/TASK_NAME:taskname            !1-6 characters
/TIME_LIMIT:n[u]               !Arg is M (minutes) by default; can
                               !also be S (seconds); 3M is
                               !default.
/UIC:[uic]                     !Privileged
```

SET [DAY]TIME   *Micro/RSX System Manager's Guide*

SET [DAY]TIME sets the system date and time. You must specify at least one of the fields, in any order. Privileged.

Format

```
SET [DAY]TIME:[dd-mmm-yy] [hh:mm]
```

SET DEFAULT   Chapter 12

SET DEFAULT establishes your default device or directory, or both. If you are a privileged user and your terminal's default is /NONAMED_DIRECTORY, then SET DEFAULT also establishes your User Identification Code.

Format

```
SET DEFAULT[/qualifier] [ddnn:][[directory]]
```

Command Qualifier

```
/[NO]NAMED_DIRECTORY          !See main text.
```

SET DEVICE   Chapter 13

SET DEVICE establishes certain device attributes. Privileged.

Format

```
SET DEVICE:ddnn:/qualifier[s]
```

Command Qualifiers

```
/CACHE:(option[,s])              !Modifies data caching; see
                                 !RSX-11M/M-PLUS System
                                 !Management Guide.

      PAR=[main_parname:]subparname[:size]
      [NO]DIRECTORY
      [NO]OVERLAY
      [NO]VIRTUAL
      [NO]LOGICAL
      [NO]READ_AHEAD
/NOCACHE
/[NO]CHECKPOINT_FILE[:n]         !n is number (decimal) of blocks in
                                 ![0,0]CORIMG.SYS.
/[NO]LOWERCASE                   !Default is /NOLOWERCASE.
/[NO]PUBLIC                      !Default is /NOPUBLIC.
/[NO]SYSTEM                      !Synonym for /[NO]PUBLIC.
/WIDTH:n                         !Nonprivileged for TI:.
```

SET ERROR_LOG *Micro/RSX System Manager's Guide*

SET ERROR_LOG sets up error-logging operations and manipulates the Error Log file.

Syntax for setting up error logging:

Format

```
SET ERROR_LOG[/qualifier[s]]ddn:[,ddn:[,s]]
```

Command Qualifiers

```
/HARD_LIMIT:n
/SOFT_LIMIT:n
/RESET_COUNTS
```

Syntax for manipulating the Error Log file:

Format

```
SET ERROR_LOG[/qualifier[s]]
```

Command Qualifiers

```
/BACKUP_FILE:filespec
/[NO]LIMITING
/NEW_LOG_FILE:filespec[/qualifier[s]]
                      /DELETE
                      /NEW_VERSION
```

SET FILE Chapter 12

SET FILE establishes certain file attributes.

Format

```
SET FILE[/qualifier[s]] filespec[s]
```

Command Qualifiers

```
/END_OF_FILE:(BLOCK:n,BYTE:n)
                            !See main text
/ENTER:synonym_filespec     !See main text
/NOWARNINGS                 !Suppresses error messages
/REMOVE                     !See main text
/REWIND                     !Magtape only; rewinds tape before
                            !beginning operation
/TRUNCATE                   !Eliminates blocks allocated but
                            !unused; saves disk space
```

SET HOST Chapter 2

SET HOST connects your terminal to a remote system. You issue this command after you have logged in to your current system. Both your current system and the remote system (nodename) must run DECnet software.

Format

```
SET HOST   nodename
```

SET LIBRARY/DIRECTORY Chapter 15

SET LIBRARY/DIRECTORY establishes the directory where the system utilities and other nonprivileged system tasks are kept. Note that this command does not create the specified directory. Privileged.

Format

```
SET LIBRARY/DIRECTORY:[directory]
```

SET [NO]PARTITION Chapter 15

SET [NO]PARTITION creates or eliminates a partition. The partition name must be from one through six characters. Privileged.

Format

```
SET [NO]PARTITION:parname/qualifier[s]
```

Command Qualifiers

```
/BASE:n
/DEVICE                    !Device common.
/DIAGNOSTIC
/SIZE:n
/SYSTEM
/TOP:value
```

SET PASSWORD Chapter 11

SET PASSWORD changes your password.

Format

```
SET PASSWORD
```

SET PRIORITY Chapter 15

SET PRIORITY alters the priority of an active task. The active task's priority (n) can be from 0 through 250. Privileged.

Format

```
SET PRIORITY:n taskname
```

SET PROTECTION Chapter 12

SET PROTECTION changes the protection code of files. The protection code controls who can access files and in what ways. The first format is preferred. The default code is (SY:RWED,OW:RWED,GR:RWED,WO:R).

Format

```
SET PROTECTION:(code)[/qualifier[s]] filespec[s]
SET PROTECTION[/qualifier[s]] filespec[s] (code)
```

Command Qualifiers

```
/DATE:dd-mmm-yy                !Given day only
/SINCE:dd-mmm-yy               !From given day through current day
/THROUGH:dd-mmm-yy             !From beginning through given day
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
                               !From given day through given day
/TODAY                         !Today only
/EXCLUDE:filespec              !Filespec can include wildcards.
```

SET PROTECTION/[NO]DEFAULT Chapter 12

SET PROTECTION/DEFAULT establishes your personal default protection code (for example, (SY:RWED,OW:RWED,GR:R,WO:) for all files that you create after issuing this command. It is recommended that you place this command in your LOGIN.CMD file. SET PROTECTION/NODEFAULT removes your personal default file protection.

Format

```
SET PROTECTION:(code)/DEFAULT
SET PROTECTION/NODEFAULT
```

SET QUEUE/ENTRY Chapter 12

SET QUEUE/ENTRY modifies by entry number some attributes of print or batch jobs once they are in a queue. However, you can specify only one attribute at a time. See SET QUEUE/JOB to modify by job name.

Format

```
SET QUEUE/ENTRY:n[/qualifier]
```

Command Qualifiers

```
/AFTER:(dd-mmm-yy hh:mm)
/COPIES:n
/[NO]DELETE
/FILE_POSITION:n
/FORMS:n
/JOBCOUNT:n
/LENGTH:n
/[NO]LOWERCASE
/PAGE_COUNT:n
/PRIORITY:n                  !n is 1-150 nonprivileged; through
                             !250 privileged; default n=50
/RELEASE                     !Same as RELEASE/QUEUE
/[NO]RESTART
/[NO]UPPERCASE
```

SET QUEUE/JOB Chapter 12

SET QUEUE/JOB modifies by job name some attributes of print or batch jobs once they are in a queue. However, you can specify only one attribute at a time. See SET QUEUE/ENTRY to modify by entry number.

Format

```
SET QUEUE/JOB[/qualifier] queuename [[uic]]jobname
```

Command Qualifiers

```
/AFTER:(ddd-mmm-yy hh:mm)
/COPIES:n
/[NO]DELETE
/FILE_POSITION:n
/FORMS:n
/JOBCOUNT:n
/LENGTH:n
/[NO]LOWERCASE
```

```
/PAGE_COUNT:n
/PRIORITY:n                      !n is 1-150 nonprivileged; through
                                 !250 privileged; default n=50
/RELEASE                         !Same as RELEASE/QUEUE
/[NO]RESTART
/[NO]UPPERCASE
```

SET SYSTEM *Micro/RSX System Manager's Guide*

SET SYSTEM establishes certain characteristics of the system. You must always specify a qualifier. Privileged.

Format

```
SET SYSTEM/qualifier
```

Command Qualifiers

```
/[NO]CRASH_DEVICE[:ddn:]         !See Advanced Programmer's Kit.
/DIRECTORY:[directory]           !Sets directory where system tasks
                                 !are kept; does not create a
                                 !directory.
/EXTENSION_LIMIT:n               !Sets maximum size a task can
                                 !extend itself with the Extend Task
                                 !directive.
/[NO]LOGINS
/NETWORK_UIC
/PACKETS:n                       !n = 0 through 15.
/POOL:top:max:total              !Increases size of pool; see main
                                 !text.
/POOL/LIMITS:arg
          HIGH=n
          LOW=n
          MINIMUM_SIZE:n
          TASK_PRIORITY:n        !See main text.
```

SET TERMINAL Chapter 11

SET TERMINAL sets various attributes of your terminal. Privileged users can set attributes for any terminal.

Format

```
SET TERMINAL[:ttnn:]/qualifier[s]
```

Command Qualifiers

Group 1: Common Use

/[NO]BROADCAST	
/CLI:cliname	
/[NO]CONTROL=C	
/DCL	
/[NO]HOLD_SCREEN	!Not for VT100s
/INQUIRE	!Automatically sets proper terminal !characteristics
/[NO]LOWERCASE	!/NOLOWER same as /UPPER and is !default.
/[NO]PRIVILEGED	!Privileged
/SPEED:(transmit,receive)	!Remember to set hardware !after this command.
/[NO]UPPERCASE	!/NOUPPERCASE same as /LOWERCASE
/WIDTH:n	

Group 2: Terminal Setup

/[NO]ADVANCED_VIDEO	
/[NO]ANSI_CRT	
/[NO]AUTOBAUD	
/ASR33	
/ASR35	
/[NO]BLOCK_MODE	
/CRFILL:n	!n can be 0–7.
/[NO]DEC_CRT	
/DTC01	
/[NO]EDIT_MODE	
/[NO]FORM_FEED	
/[NO]HARDCOPY	
/[NO]HOSTSYNC	
/KSR33	
/KSR35	
/LA12	
/LA30P	
/LA30S	
/LA34	
/LA36	
/LA38	
/LA50	
/LA100	
/LA120	

```
/LA180S
/LA210
/LFFILL
/LN03
/LQP02
/LQP03
/PAGE_LENGTH:n
/PRINTER_PORT
/PRO_SERIES
/[NO]REGIS
/[NO]SCOPE
/[NO]SOFT_CHARACTERS
/[NO]TAB
/[NO]TRANSLATION_ROUTINE[:arg]
                                      n          !ACD number
                                      logical    !Logical name
                                                 !for ACD number
/[NO]TTSYNC
/VT05
/VT50
/VT52
/VT55
/VT61
/VT100
/VT101
/VT102
/VT105
/VT125
/VT131
/VT132
/VT200_SERIES
/WIDTH:n
```

Group 3: Task Setup

```
/[NO]ECHO
/[NO]EIGHT_BIT
/[NO]ESCAPE
/[NO]FULL_DUPLEX
/[NO]INTERACTIVE
/[NO]LOCAL
```

```
/[NO]PARITY[:type]          !ODD is default.
          ODD
          EVEN
/[NO]PASSALL
/[NOPASTHRU
/[NO]REMOTE
/[NO]SERIAL                 !/SERIAL is default.
/[NO]SLAVE
/[NO]TYPEAHEAD[:n]          !n can be 0–255.
/[NO]WRAP
```

SET UIC   Chapter 12

SET UIC changes the User Identification Code of privileged users. If your terminal's default is /NONAMED_DIRECTORY, then this command also changes the directory location of a privileged user.

Format

```
SET UIC  [uic]
```

SHOW ACCOUNTING   *Micro/RSX Advanced Programmer's Kit*

SHOW ACCOUNTING displays current information on your terminal session, if you are nonprivileged. Privileged users can display information about any terminal session.

Format

```
SHOW ACCOUNTING/qualifier
```

Command Qualifiers

```
/INFORMATION
/TRANSACTIONS[:infile] outfile
```

SHOW ASSIGNMENTS Chapter 13

SHOW ASSIGNMENTS displays at your terminal your local and login logical name assignments. Privileged users can display assignments for other terminals as well as all assignments in the operating system.

SHOW LOGICAL displays the same information as SHOW ASSIGNMENTS.

Format

```
SHOW ASSIGNMENTS[/qualifier[s]]
```

Command Qualifiers

```
/ALL
/GLOBAL                  !Privileged
/GROUP[:g]               !UIC group number
/LOCAL                   !Default
/LOGIN                   !Same as /LOCAL
/SYSTEM                  !Synonym for /GLOBAL; privileged
/TERMINAL:ttnn:          !Privileged
```

SHOW CACHE *RSX-11M/M-PLUS System Management Guide*

SHOW CACHE displays data caching information.

Format

```
SHOW CACHE [ddnn:][/qualifier]
```

Command Qualifiers

```
/RATE:n                  !n is number of seconds
```

SHOW CLOCK_QUEUE Chapter 15

SHOW CLOCK_QUEUE displays information about tasks currently in the clock queue. This information consists of the task names, the next time each task is to run, and each task's reschedule interval, if any.

Format

```
SHOW CLOCK_QUEUE
```

SHOW COMMON Chapter 15

SHOW COMMON displays the names of resident commons installed in the system, their PCB addresses, the number of attached tasks, and the status of the common. If you do not specify a common name, all commons are displayed.

Format

SHOW COMMON[:name][/qualifier]

Command Qualifier

/TASK	!Displays tasks attached to a !named common.

SHOW [DAY]TIME *Micro/RSX System Manager's Guide*

SHOW [DAY]TIME displays the system time and date setting.

Format

SHOW [DAY]TIME

SHOW DEFAULT Chapter 12

SHOW DEFAULT displays the current default device and directory for your terminal. This command also displays whether your terminal's default is /NAMED_DIRECTORY or /NONAMED_DIRECTORY, and your User Identification Code.

Format

SHOW DEFAULT

SHOW DEVICES Chapter 13

SHOW DEVICES displays information about the devices included in the system. This command displays 2-character mnemonic names only.

Format

```
SHOW DEVICES[/qualifier]
```

Command Qualifiers

```
/[NO]CACHE
/dd[nn:]
/[NO]PUBLIC
/[NO]SYSTEM                   !Synonym for /PUBLIC
/WIDTH:ddnn:
```

SHOW ERROR_LOG *Micro/RSX System Manager's Guide*

SHOW ERROR_LOG provides a brief display of error-logging information on the devices specified. If you do not specify any devices, the qualifier provides information on all devices in the system. The default is to display this report on your terminal.

Format

```
SHOW ERROR_LOG[/qualifier[s]] devlist
```

Command Qualifiers

```
/CURRENT
/HISTORY
/OUTPUT[:filespec]
/RECENT
```

SHOW HOST Chapter 2

SHOW HOST displays the name of the processor to which your terminal currently is connected. It also shows you the name and version number of the operating system running on the processor.

Format

```
SHOW HOST
```

SHOW LIBRARY   Chapter 15

SHOW LIBRARY displays the current Micro/RSX library directory. This is the directory where the nonprivileged system utilities are kept.

Format

```
SHOW LIBRARY[/qualifier]
```

Command Qualifier

```
/DIRECTORY                  !Non-operational, for
                            !for VMS compatibility
```

SHOW LOGICALS   *Micro/RSX Guide to Advanced Programming*

SHOW LOGICALS displays at your terminal your local and login logical name assignments. Privileged users can display assignments for other terminals as well as all assignments in the operating system.

SHOW ASSIGNMENTS displays the same information as SHOW LOGICAL.

Format

```
SHOW LOGICAL[/qualifier[s]]
```

Command Qualifiers

```
/ALL
/GLOBAL                     !Privileged
/GROUP[:g]                  !UIC group number
/LOCAL                      !Default
/LOGIN                      !Same as /LOCAL
/SYSTEM                     !Synonym for /GLOBAL; privileged
/TERMINAL:ttnn:             !Privileged
```

SHOW PARTITIONS   Chapter 15

SHOW PARTITIONS displays address and content information about the partitions in the system. You can display information about all partitions or about a single partition.

Format

```
SHOW PARTITIONS[:name]
```

SHOW PROCESSOR Chapter 12

SHOW PROCESSOR displays information about the processors, batch processors, printers, card readers, and other devices under control of the Queue Manager.

Format

```
SHOW  processortype processorname
```

Processor Types

CARD_READER	!INPUT is synonym.
DEVICE	!All nonbatch output processors; !synonym for PRINTER.
INPUT	!CARD_READER is synonym.
PRINTER	!All nonbatch output processors; !synonym for DEVICE.
PROCESSOR	

SHOW PROTECTION Chapter 12

SHOW PROTECTION displays your personal default file protection code. Your default file protection can be established in two ways: by issuing the SET PROTECTION/DEFAULT command and by using the Account File Maintenance Utility (ACNT) to enter a protection code for your account.

Format

```
SHOW PROTECTION
```

SHOW QUEUE Chapter 12

SHOW QUEUE displays information about batch and print jobs in queues.

Format

```
SHOW QUEUE[/qualifier] [queuename]
```

Command Qualifiers

/ALL	!All entries in all queues. !Default.
/BATCH	!All entries in all batch queues.
/BRIEF	
/DEVICE	!All nonbatch queues; synonym for !/PRINTER.

/ENTRY:n	
/FILES	!Lists files in each job; not as !long as /FULL.
/FORMS:n	
/FULL	
/NAME:jobname	!Lists only jobs with that name; !may be more than one.
/OWNER_UIC:[[uic]]	!Lists only jobs from that UIC; !default UIC is login UIC.
/PRINT	!All nonbatch queues; synonym for !/DEVICE.

SHOW SYSTEM *Micro/RSX System Manager's Guide*

SHOW SYSTEM displays information about the current system.

Format

```
SHOW SYSTEM[/qualifier]
```

Command Qualifiers

/CLI	!CLIs on current system
/CRASH_DEVICE	
/DIRECTORY	!Default; displays current system !directory
/EXTENSION_LIMIT	!Task extension limit
/NETWORK_UIC	!Directory that DECnet-tasks !are located
/PACKETS	!Maximum I/O packets and number !currently available
/POOL	!Displays pool statistics
/POOL/LIMITS	!Dispalys pool limits
/SECONDARY_POOL	!Shows secondary pool

SHOW TASKS   Chapter 15

SHOW TASKS displays information about active or installed tasks.

Format

SHOW TASKS[:taskname]/qualifier[s]

Command Qualifiers

/ACTIVE[:ttnn:]	
/DEVICE:ddnn:	!Show tasks installed from named !device
/INSTALLED	
/LOGICAL_UNITS	!Static LUNs for installed task !qualifier
/BRIEF	
/FULL	
/ALL	

SHOW TERMINAL   Chapter 11

SHOW TERMINAL displays information about your terminal and other terminals on your system. If you do not specify a qulifier, SHOW TERMINAL displays all attributes for your terminal (TI:).

Format

SHOW TERMINAL[:ttnn:][/qualifier]

Command Qualifiers

/[NO]ADVANCED_VIDEO
/[NO]ANSI_CRT
/[NO]AUTOBAUD
/[NO]ASR33
/[NO]ASR35
/[NO]BLOCK_MODE
/[NO]BROADCAST
/CLI:cliname
/[NO]CONTROL:C
/[NO]CRFILL
/DCL
/[NO]DEC_CRT
/[NO]DTC01
/[NO]ECHO
/[NO]EDIT_MODE

```
/[NO]EIGHT_BIT
/[NO]ESCAPE
/[NO]FORM_FEED
/[NO]FULL_DUPLEX
/[NO]HARDCOPY
/[NO]HOLD_SCREEN
/[NO]HOSTSYNC
/HT                           !DECnet host terminal
/[NO]INTERACTIVE
/[NO]KSR33
/[NO]KSR35
/[NO]LA12
/[NO]LA24
/[NO]LA30P
/[NO]LA30S
/[NO]LA34
/[NO]LA36
/[NO]LA38
/[NO]LA50
/[NO]LA100
/[NO]LA120
/[NO]LA180S
/[NO]LA210
/[NO]LFFILL
/[NO]LN03
/[NO]LOCAL
/LOGGED_ON
/[NO]LQP02
/[NO]LQP03
/[NO]LOWERCASE                !NOLOWER same as /UPPER
                              !and is default.
/MODEL
/PAGE_LENGTH
/[NO]PARITY
/[NO]PASSALL
/[NO]PASTHRU
/[NO]PRIVILEGE
/PRINTER_PORT
/PRO_SERIES
/[NO]REGIS
/[NO]REMOTE
/RT                           !DECnet host terminal
```

```
/[NO]SCOPE
/[NO]SERIAL
/[NO]SLAVE
/[NO]SOFT_CHARACTERS
/SPEED
/[NO]TAB
/TI:                    !All real terminals
/TT
/[NO]TTSYNC
/[NO]TYPE_AHEAD
/[NO]UPPERCASE          !/NO UPPERCASE same as
                        !/LOWERCASE.
/VT:                    !Virtual terminal
/[NO]VT05
/[NO]VT50
/[NO]VT52
/[NO]VT55
/[NO]VT61
/[NO]VT100
/[NO]VT101
/[NO]VT102
/[NO]VT105
/[NO]VT125
/[NO]VT131
/[NO]VT132
/[NO]VT200_SERIES
/WIDTH
/[NO]WRAP
```

SHOW UIC   Chapter 12

SHOW UIC displays your User Identification Code (UIC). Your UIC is unique and identifies you to the operating system. Your UIC also determines whether you are a privileged or nonprivileged user.

Format

```
SHOW UIC
```

SHOW USERS *Introduction to Micro/RSX*

SHOW USERS displays all currently logged-in terminals, including DECnet host terminals and virtual terminals, with the default directory and login UIC for each.

Format

```
SHOW USERS
```

START Chapter 15

START resumes execution of a task stopped by a STOP$S directive. The task name defaults to TTnn:

Format

```
START[/qualifier] [taskname]
```

Command Qualifier

```
/TERMINAL:ttnn:              !Privileged
```

START/ERROR_LOG *Micro/RSX System Manager's Guide*

START/ERROR_LOG begins logging on all error-logging devices in the system. This qualifier uses LB:[1,6]LOG.ERR as the default Error Log file and LB:[1,6]BACKUP.ERR as the default backup file. It starts error limiting with default limits of five hard errors and eight soft errors.

Format

```
START/ERROR_LOG[/qualifier[s]] filespec
```

Command Qualifiers

```
/INCLUDE[:(arg[s])]          !Selects what kind of errors
                             !will be logged
          ALL
          CONTROL
          ERRORS
          MEMORY
          PERIPHERAL
          PROCESSOR
          SYSTEM_INFORMATION
/NEW_VERSION
/[NO]LIMITING
/UPDATE
/ZERO
```

START/processortype *Micro/RSX System Manager's Guide*

This form of the START command starts a processor, batch processor, output processor, or card-reader processor. Privileged.

Format

```
START/processortype processorname[/qualifier[s]]
```

Processortypes

```
APPLICATIONS_PROCESSOR
BATCH_PROCESSOR
CARD_READER                    !INPUT is synonym.
DEVICE                         !PRINTER is synonym.
INPUT                          !CARD_READER is synonym.
PRINTER                        !DEVICE is synonym.
PROCESSOR
```

Command Qualifiers

```
/FORMS:n                       !Overrides value set on
                               !initialization.
/CONTINUE                      !Default
/RESTART
/NEXT
/TOP_OF_FILE
/BACKSPACE:n
/FORWARDSPACE:n
/PAGE:n
/ALIGN
```

START/QUEUE *Micro/RSX System Manager's Guide*

START/QUEUE starts a queue. Privileged.

Format

```
START/QUEUE   queuename
```

START/QUEUE/MANAGER *Micro/RSX System Manager's Guide*

START/QUEUE/MANAGER starts the Queue Manager. Privileged.

Format

```
START/QUEUE/MANAGER
```

START/UNBLOCK Chapter 15

START/UNBLOCK continues the execution of a task blocked by the STOP /BLOCK command. Nonprivileged users can unblock any task running from their own terminal. Privileged users can unblock any task. See main text.

Format

```
START/UNBLOCK[/qualifier] [taskname]
```

Command Qualifier

```
/TERMINAL:ttnn:          !Privileged
```

STOP/ABORT Chapter 12

STOP/ABORT stops the current job on a line printer immediately. Privileged users can stop any job. Nonprivileged users can stop their own jobs.

Format

```
STOP/ABORT   printer[:]
```

STOP/BLOCK Chapter 15

STOP/BLOCK blocks an installed running task. The task no longer executes or competes for memory. Nonprivileged users can block tasks running from their own terminals. Privileged users can block any task. See main text.

Format

```
STOP/BLOCK[/qualifier] [taskname]
```

Command Qualifier

```
/TERMINAL:ttnn:          !Privileged
```

STOP/ERROR_LOG *Micro/RSX System Manager's Guide*

STOP/ERROR_LOG causes logging to stop on all devices and stops error limiting as well.

Format

```
STOP/ERROR_LOG
```

STOP/processortype *Micro/RSX System Manager's Guide*

This form of the STOP command stops a processor, batch processor, card-reader processor, printer, or other output processor. Privileged.

Format

```
STOP/processortype processorname[/qualifier[s]]
```

Processortypes

APPLICATIONS_PROCESSOR	
BATCH_PROCESSOR	
CARD_READER	!INPUT is synonym.
DEVICE	!PRINTER is synonym.
INPUT	!CARD_READER is synonym.
PRINTER	!DEVICE is synonym.
PROCESSOR	

Command Qualifiers

/ABORT	!See main text.
/FILE_END	
/JOB_END	
/PAUSE	

STOP/QUEUE *Micro/RSX System Manager's Guide*

STOP/QUEUE stops queues. Privileged.

Format

```
STOP/QUEUE   queuename
```

STOP/QUEUE/MANAGER

STOP/QUEUE/MANAGER stops the Queue Manager. The Queue Manager stops after the current job is processed (unless you specify the /ABORT qualifier). Privileged.

Format

```
STOP/QUEUE/MANAGER[/qualifier]
```

Command Qualifier

```
/ABORT                         !Stops QMG immediately.
```

SUBMIT Chapter 8

SUBMIT queues QMG batch jobs consisting of one or more user batch jobs for processing by a batch processor.

Format

```
SUBMIT[/qualifier[s]] filespec[s]
```

Command Qualifiers

```
/AFTER:TOMORROW
/AFTER:(dd-mmm-yy hh:mm)
/[NO]DELETE                    !Deletes batch file after run;
                               !command or filespec qualifier.
/[NO]HOLD                      !Default is /NOHOLD; /HOLD has
                               !same effect as HOLD command.
/[NO]LOG_FILE
/NAME:jobname                  !1–9 characters; default is first
                               !filename.
/[NO]PRINTER[:queuename]       !Optionally name queue for log
                               !print job.
/PRIORITY:n                    !n is 1–150 nonprivileged; through
                               !250 privileged; default n=50.
/QUEUE:queuename
/[NO]RESTART
/[NO]TRANSFER
```

TYPE Chapter 12

TYPE prints selected files on your terminal.

Format

```
TYPE  [/qualifier[s]]filespec[s]
```

Command Qualifiers

```
/EXCLUDE:filespec            !Filespec can include wildcards.
/NOWARNINGS                  !Suppresses error messages
/SHARED                      !Allows other user to access file
                             !while you type it
/DATE:dd-mmm-yy              !Given day only
/SINCE:dd-mmm-yy             !From given day through current day
/THROUGH:dd-mmm-yy           !From beginning through given day
/SINCE:dd-mmm-yy             !From given day through given day
/TODAY                       !Today only
```

UNFIX Chapter 15

UNFIX frees a fixed task or region from memory. The taskname parameter can be the name of either a task or a region. Privileged.

Format

```
UNFIX[/qualifier] taskname
```

Command Qualifiers

```
/READONLY_SEGMENT            !RO segment of multiuser task.
/REGION
```

UNLOCK Chapter 12

UNLOCK unlocks locked files. Locked files are files that have been improperly closed. They are identified by an "L" in the directory listing.

Format

```
UNLOCK  [/qualifier[s]] filespec[s]
```

Command Qualifiers

```
/EXCLUDE:filespec            !Filespec can include wildcards.
/NOWARNINGS                  !Suppresses error messages
/DATE:dd-mmm-yy              !Given day only
/SINCE:dd-mmm-yy             !From given day through current day
```

```
/THROUGH:dd-mmm-yy          !From beginning through given day
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
                            !From given day through given day
/TODAY                      !Today only
```

Index

Micro/RSX User's Guide
Volume 1
AA-Y539B-TC

USER'S COMMENTS

Your comments and suggestions are welcome and will help us in our continuous effort to improve the quality and usefulness of our documentation and software.

Remember, the system includes information that you read on your terminal: help files, error messages, prompts, and so on. Please let us know if you have comments about this information, too.

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.

What kind of user are you? ____ Programmer ____ Nonprogrammer

What do you use the system for? ____________________

Years of experience as a computer programmer/user: ____________

Name ____________________ Date ____________

Organization ____________________

Street ____________________

City ____________________ State ____________ Zip Code or Country ________

— — — Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

— — — Do Not Tear - Fold Here —

digital

AA-Y539B-TC