RSTS/E
DCL User's Guide

Order No. AA–L426B–TC

March 1983

This manual describes the use of DCL (the DIGITAL Command Language) on RSTS/E. It includes general information about RSTS/E, provides rules for using DCL, and describes DCL commands in file, system, and programming operations.

OPERATING SYSTEM AND VERSION: RSTS/E V8.0

digital equipment corporation, maynard, massachusetts

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

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

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

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

The following are trademarks of Digital Equipment Corporation:

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

Commercial Engineering Publications typeset this manual using DIGITAL's TMS–11 Text Management System.

Contents

7 Program Development

Figures

Tables

Preface

Document Objectives

This manual explains the use of DCL (the DIGITAL Command Language) on RSTS/E and then describes each DCL command. A glossary of DCL and RSTS/E terminology is provided at the end.

You can use the commands in this manual for such operations as working with files, getting information about the system and its devices, running batch jobs, and developing programs.

Intended Audience

Although some familiarity with computers is helpful, you do not need to be an experienced computer user or programmer to use this manual.

Related Documents

If you need information on fundamental RSTS/E concepts, refer to the *RSTS/E Primer*. To get started with BASIC–PLUS–2 programming, read the *Introduction to BASIC*. For more information about the use of RSTS/E, consult the *RSTS/E System User's Guide*. Other manuals in the RSTS/E documentation set are described in the *RSTS/E Documentation Directory*.

Using This Manual

The manual is divided into seven chapters and three appendixes:

1. Using RSTS/E

   This chapter describes the basics of RSTS/E and how to use it. It includes descriptions of the DCL environment, running a program, and using DECnet/E. (DECnet/E allows you to communicate with other computers.)

2. Using DCL Commands

   This chapter includes rules that apply to the commands in this manual and the use of these commands.

3. Working With Files

   This chapter describes the use of DCL commands to create, display, edit, and compare files.

4. System Operations

   This chapter describes how to use DCL in day-to-day RSTS/E operations, such as displaying system status and setting characteristics for your terminal.

5. Working With Devices

   This chapter describes the devices available for your use on a RSTS/E system.

6. Batch Processing

   This chapter describes how to create a batch control file and run a batch job.

7. Program Development

   This chapter describes the commands available for you to develop and run programs. DCL gives you access to the BASIC, COBOL, DIBOL, FORTRAN, and MACRO programming languages.

Appendixes

A – Contains an alphabetical table of DCL commands.

B – Lists error messages that RSTS/E displays when you enter a command that RSTS/E cannot execute.

C – Lists the differences between DCL on RSTS/E and VMS, DIGITAL's operating system for VAX computers.

The manual also contains a glossary of RSTS/E and DCL terminology and an index.

The commands in this manual are grouped by their functions. For example, the section in Chapter 3 on "Printing Files" describes four commands available for requesting hard–copy listings of files. Each command is discussed briefly, and then its format is shown with any available qualifiers.

Conventions

The following symbols and conventions are used in this manual:

[ ] Square brackets show the optional parts of a command in format statements. For example:

```
DIRECTORY [file-spec[,...]]
```

The optional parts of the command are that you can include a file specification ([file–spec]), or more than one ([,...]), if you choose.

Square brackets also indicate the choice you have in using a command. For example:

```
/[NO]DELETE
```

This means you can type either /DELETE or /NODELETE, depending on the form of the qualifier you select.

Square brackets in command formats are not to be confused with the square brackets in PPNs (Project–Programmer numbers, as in [52,20]), which are described in Chapter 1.

< > Angle brackets surrounding text indicate a place holder for what the system inserts when an error message occurs.

Color Red print shows what you type in examples.

CTRL/x The control key, which you use in combination with another key. For example, you enter CTRL/U by holding down the CTRL key and pressing the keyboard key labeled "U". RSTS/E "echoes," or displays CTRL/U at your terminal as ^U.

RET The key labeled RETURN on your terminal. You press the RETURN key to complete lines and commands that the system will process.

(N) The network symbol, used in command descriptions to show which qualifiers you can use over the network.

Summary of Technical Changes

For V8.0, the overall organization of this manual is the same as the previous edition. However, the manual is revised to describe:

- Changes to the LINK command for cluster libraries and the /DMS qualifier (Chapter 7).
- New commands for spooling (Chapters 3 and 6).
- New qualifiers for the COBOL command (Chapter 7).
- Changes to the BASIC command resulting from BASIC–PLUS–2 V2.0 (Chapter 7).
- /BLOCK_SIZE and /CLUSTER_SIZE qualifiers for the COPY command (Chapter 3).
- Changes to the MOUNT command (Chapter 5).
- New error messages (Appendix B).
- Restrictions for file specifications in DCL — previously in the RSTS/E Maintenance Notebook (Chapter 1).
- New appendix on RSTS/E and VMS DCL differences (Appendix C).

1 Using RSTS/E

This chapter contains information to help you become familiar with using RSTS/E.

Becoming a RSTS/E User

In order for you to use the computer, your system manager must assign you an account number and password. The account number identifies you to the system. The password is a precaution to keep uninvited users from accessing the system.

When you log in, the system displays a message to confirm that you are logged into your account. The system may also display informational messages at your terminal.

After you are logged in, you have access to all the RSTS/E facilities that your account allows. There are two kinds of accounts: privileged and nonprivileged.

A privileged account is primarily for system management functions. For example, a privileged user can:

- Create and delete accounts
- Change passwords
- Add commands to the system
- Control the number of users on the system
- Save user files in case of accidental damage
- Shut down the system

By contrast, nonprivileged users work with files in their own accounts and can access other people's files only with their permission. (A file is a unit that contains text, programs, or data of any kind.)

Your PPN

Account numbers on RSTS/E are called PPNs, or project-programmer numbers. Your PPN corresponds to your account and identifies you as a system user.

A PPN consists of two numbers enclosed in square brackets and separated by a comma, as in [52,20]. The numbers are:

1. A project number (the number 52 in [52,20]), which corresponds to a group of users. For example, several people who work together and who use a RSTS/E system may be assigned the same project number.
2. A programmer number (the number 20 in [52,20]), which is unique for each user in a group.

The brackets around the PPN show that the two numbers are used together. You do not type the brackets when you log in, although you may need the brackets in operations such as file transfers (explained later). Do not confuse brackets around a PPN with brackets used to indicate optional parts of a command (see "Conventions" in the Preface).

If you have more than one account on the system, then you use different PPNs and passwords to log into your different accounts.

Your Password

Your password protects your account from unauthorized use. Passwords on RSTS/E systems can consist of one to six alphanumeric characters.

Your system manager may assign you a password or let you request one. (The method of assigning a password depends on the policy at your site.) Your system manager can also change your password at your request.

If you request a password, it is a good idea (for security reasons) to select one that is not obvious to others. This is especially important for privileged accounts.

Some suggestions for selecting a unique password are:

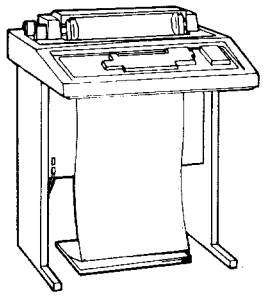
- Request a password that is at least five characters long.
- Do not use a word or name as a password. (Selecting your own name or that of a friend might be too easy for unauthorized users to guess.)
- Choose a password that contains letters and digits.

Having the system manager change your password periodically is also a good idea.

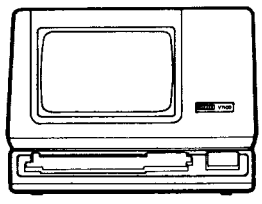
Using the Terminal (CTRL, DELETE, RETURN Keys)

You use a terminal to communicate with the system.

Terminals can be grouped into two main categories: hard-copy and video (see Figure 1–1). A hard-copy terminal, such as an LA36, prints the characters you type and output from the system onto lineprinter paper. A video terminal, such as a VT52 or VT100, displays characters on a CRT (Cathode Ray Tube) screen.



LA36: A Hard-Copy Terminal



VT100: A Video Terminal MK-01111-00

Figure 1–1: LA36 and VT100 Terminals

Most of the keys on a terminal match keys on a typewriter keyboard. However, several keys on your terminal's keyboard have special functions. Three of the most commonly used keys are labeled CTRL, DELETE (also known as RUBOUT), and RETURN.

The key labeled CTRL is called the *control* key. You always use the control key in combination with another key. For example, you can use CTRL/U to delete all the characters on a command line if you have not yet pressed the RETURN key. Use CTRL/U by holding down the control key and then pressing the "U" key. RSTS/E displays this at your terminal as ^U:

```
$ EDIT/EDT RACER.PDQ(CTRL/U)
```

On your terminal, this looks like:

```
$ EDIT/EDT RACER.PDQ ^U
```

The line is now deleted. You can press the RETURN key for the DCL prompt:

```
(RET)
$
```

For a complete description of each control function available on RSTS/E, refer to the *RSTS/E System User's Guide.*

The DELETE key erases the last character you typed. Although you can delete more than one character on a line, DELETE does not erase characters on the preceding line.

The DELETE key has the same function on a hard-copy terminal as on a video terminal, but the effects look different. (On some hard-copy terminals, the key labeled RUBOUT performs the delete function.) For example, suppose you misspell a DCL command and use the DELETE key twice:

```
$ COPIE(DEL) (DEL)
```

On a video terminal, the resulting line reads:

```
$ COP
```

By contrast, deleted characters on a hard-copy terminal are displayed between backslashes (\), because a hard-copy terminal is not designed to "erase" characters:

```
$ COPIE\EI\(RET)
```

The RETURN key allows you to:

- End commands — When you enter the name of a command, pressing the RETURN key tells the system to execute the command. For example:

```
$ DIRECTORY(RET)
```

- End lines — When you enter text in a file, the RETURN key ends each line and begins a new one. For example:

```
Each year about 500(RET)
runners enter Littleton's(RET)
annual road race.(RET)
```

- Respond to a prompt — Sometimes pressing the RETURN key causes the system or a program to assume a response. (These assumptions are known as "defaults.") For example, the following prompt allows you to enter a line width, but lets you know that, if you press RETURN without entering a number, a width of 50 characters per line is assumed:

```
Width of line <50> ?(RET)
```

- Verify that the system is in operation — When the system is *down,* or not operational, it does not respond when you press RETURN. Otherwise, the system displays another DCL prompt:

```
$ (RET)
$
```

A common mistake is to press the NO SCROLL key (on VT100s) or CTRL/S to stop output to the terminal, and then forget to press NO SCROLL again or CTRL/Q to resume output. (Nothing is printed at the terminal when you press NO SCROLL or CTRL/S.) In either case, the system does not display another DCL prompt when you press RETURN. Press NO SCROLL again or CTRL/Q to resume output.

This section discussed three of the most commonly used keys: CTRL, DELETE, and RETURN. Other keys and special characters are listed in Table 1–1.

Table 1–1: Special Characters on RSTS/E

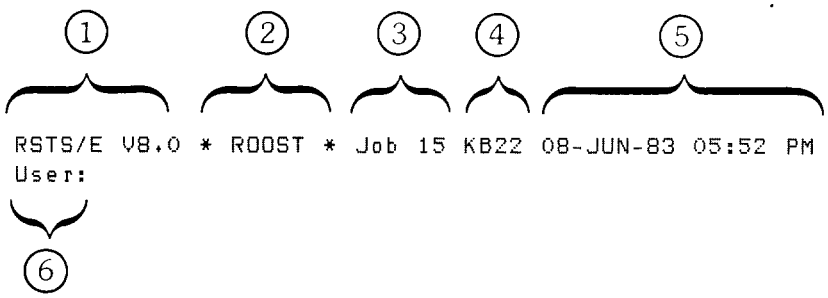
Key	Function
CTRL/C	Halts execution of the current command or program and returns control to the job keyboard monitor. Echoes on the terminal as "^C".
CTRL/O	Stops and restarts terminal output while a program is running. Echoes on the terminal as "^O".
CTRL/Q	Resumes terminal output suspended by CTRL/S while a program is running. You can use CTRL/Q only if the TTSYNC characteristic is set for the terminal.
CTRL/R	Redisplays the current terminal line.
CTRL/S	Suspends terminal output while a program is running. You can use CTRL/S only if the TTSYNC characteristic is set for the terminal.
CTRL/T	Displays a one-line status report for your job. The report includes your job number, keyboard number, job state, and other information (CTRL/T is an optional feature, so it may not be available on your system.)
CTRL/U	Deletes the current terminal line. CTRL/U does not erase characters from the screen. Instead, it echoes "^U" and moves the cursor to the next line.
CTRL/Z	Is an end-of-file marker.
DELETE or RUBOUT	Erases the last character typed. On hard-copy terminals, erased characters are displayed between backslashes.
ESCAPE or ALTMODE	Sends a typed line to the system for processing. Echoes on your terminal as a "$".
FORM FEED or CTRL/L	Sends a typed line to the system for processing. Performs four line feed operations at a video terminal. Advances paper to the top of the next page on hard copy terminals.
LINE FEED	Ends the current line. (Same as RETURN.)
NO SCROLL	Performs the same function as CTRL/S and CTRL/Q on a VT100, VT101, VT125, and other VT100–type terminals. The TTSYNC characteristic must be set for the terminal.
RETURN	Sends a typed line to the system for processing. Performs a carriage return/line feed operation at the terminal.
TAB or CTRL/I	Moves the cursor to the next tab stop on the terminal line. By default, tab stops are eight spaces apart.

Beginning a Session at Your Terminal

You can log in to your account on the system as soon as you have a PPN and password. To get started, first type HELLO and press RETURN:

```
HELLO(RET)
```

The system signals that it is in operation by displaying a message similar to:



This message gives you the following information:

1 The name and version number of the computer's software.

2 The system name (ROOST, in this example). The system manager assigns a name to the computer at system installation.

3 Your job number. RSTS/E assigns a unique job number (as in Job 15) at the beginning of the login procedure to identify each user. The job number remains in effect until you log out. (You can be assigned a different job number each time you log in.)

4 Terminal number. RSTS/E recognizes your terminal by an identification number, such as KB22 for "keyboard 22." The identification number for each terminal remains the same, except for dial-up lines.

5 The date and time when you log in. RSTS/E keeps track of the length of time you use the system and its resources, starting with the date and time of login.

6 The prompt for your PPN. RSTS/E displays the User: prompt.

Enter your PPN and password after the prompts as shown. Notice that the system does not display your password as you type it, to prevent anyone who may be looking on from discovering what your password is.

```
User: 52,20RET
Password: RET
```

You will generally be able to log in this way with no problems. However, it may help you to know the following two constraints.

First, if you wait too long before answering the "User:" and "Password:" prompts (the limit is 30 seconds) or if your response to either prompt is incorrect, the system prints "Invalid entry – try again", and reprints the User: prompt. If you type in the wrong PPN, the system doesn't tell you until after you have typed your password.

Second, after five invalid login attempts, the system prints "Access denied". If you are trying to use the system over a dial-up line when this happens, the system disconnects the phone line. Verify your PPN and password before trying again.

When you log in successfully, you may see an optional message from your system manager, confirming that you now have access to RSTS/E. For example:

```
Welcome to RSTS/E!
```

Any other messages of general interest may then be displayed:

```
08-JUN-83    The system will be down Saturday, 11-JUN-83,
             from 8:00 AM to 1:00 PM for maintenance.

                                        REB

$
```

RSTS/E is now at *command level,* which means that it is ready to accept the commands you enter. The dollar sign prompt ($) indicates DCL.

DCL is a *keyboard monitor,* which means that it furnishes a command environment by accepting commands from the keyboard. You can type DCL commands after the prompt.

It is possible that your RSTS/E system displays a prompt other than the dollar sign when you log in, such as Ready for the BASIC–PLUS environment. However, you can type the commands in this manual from either DCL or from other keyboard monitors on RSTS/E. The format for using DCL commands from other keyboard monitors is to type DCL, a space, and the DCL command. For example:

```
DCL PRINT(RET)
```

You can also use this format when you are in the DCL keyboard monitor.

The next section explains how to switch to DCL.

The DCL Keyboard Monitor

The default keyboard monitor on your system is the command environment that is present when you log in. If DCL is the default keyboard monitor, the dollar prompt ($) is displayed. The dollar prompt means that RSTS/E is ready for you to enter DCL commands.

If DCL is not the default keyboard monitor, you see a prompt other than the dollar sign after you log in. If you are using a keyboard monitor other than DCL, you can enter DCL by typing:

```
DCL(RET)
```

When you succeed in switching to DCL, RSTS/E displays the dollar prompt.

Getting Information About Commands: HELP

You can use the help facility to learn about the commands that are available on RSTS/E. Each time you use HELP, RSTS/E displays the information you request and then displays the DCL prompt.

To use HELP, type:

```
$ HELP(RET)
```

The system displays the list of topics on which you can get information, similar to Figure 1–2:

```
RSTS/E DCL HELP FILE

The RSTS/E DCL User's Guide contains a complete description of all
nonprivileged DCL commands supported on RSTS/E.

For instructions on using this HELP facility, type HELP HELP.


Additional help is available on:

ALLOCATE          APPEND            ASSIGN            BASIC             CCL
COBOL             COPY              CREATE            DEALLOCATE        DEASSIGN
DELETE/ENTRY      DELETE/JOB        DELETE            DIBOL             DIFFERENCES
DIRECTORY         DISMOUNT          EDIT              FORTRAN           HELP
INITIALIZE        LINK              LOGOUT            MACRO             MOUNT
PRINT             RENAME            REQUEST           RUN               SET HOST
SET PROTECTION    SET QUEUE/ENTRY   SET QUEUE/JOB     SET TERMINAL      SHOW DEVICE
SHOW QUEUE        SHOW NETWORK      SHOW SYSTEM       SHOW TERMINAL     SHOW USERS
SUBMIT            TYPE              RSTS

$
```

Figure 1–2: Display of Help Text

The help display on your system may be different from what is shown here because the system manager can change the help information.

The display lists topics on which there are help messages available. For example, assume you typed HELP for the list of topics shown in Figure 1–2. Now you want to read about the LOGOUT command, so you type:

```
$ HELP LOGOUT(RET)
```

RSTS/E displays information about the LOGOUT command and lists the qualifiers /BRIEF and /FULL. The display on your terminal reads:

```
LOGOUT

   The LOGOUT command ends your session at the terminal.

Format

   LO[GOUT]

Command Qualifiers

    /BRIEF
    /FULL (default)

If you include the /BRIEF qualifier after the LOGOUT command, RSTS/E
ends your session at the terminal without displaying a message. If you
include the /FULL, or simply type LOGOUT, RSTS/E displays information
about the status of your account.

$
```

The RSTS topic provides you with the RSTS/E system help facility, similar to the display in Figure 1–3:

```
$ HELP RSTS(RET)

RSTS

Help can be obtained on a particular topic by typing:
        HELP topic subtopic subsubtopic ...
A topic can have the following format:
    1) an alphanumeric string (e.g., a command name, option, etc.)
    2) same preceded by a "/" (= interpreted as a switch)
    3) the match-all symbol "*"

Examples:
    HELP DIRECTORY /S
    HELP SET STALL
Abbreviations result in all matches being displayed.

Additional help is available on:

/OUTPUT          /PROMPT
ADVANCED         ASSIGN          DCL             ATTACH
BASIC            BYE             EXIT            DEASSIGN
DISMOUNT         DIRECTORY       HELLO           FILENAMES
FIT              HELP            MOUNT           KEYBOARD
LOGIN            MAIL            REASSIGN        PIP
PLEASE           QUE             SET             RT11
RSX              RUN             TYPE            SWITCH
SYSTAT           TECO                            UTILTY
VTEDIT

$
```

Figure 1–3: RSTS/E Help Text

RSTS/E help messages are available in a "hierarchy." This means that the help you request is available in levels of increasing detail. You can type HELP RSTS and a topic, such as RSX, for information about the topic:

```
$ HELP RSTS RSX(RET)
```

Each topic may have a subtopic, which you can include in the command line:

```
$ HELP RSTS RSX DISMOUNT(RET)
```

Ending a Session at the Terminal: LOGOUT

Use the LOGOUT command to end your session at the terminal. You can include one of two qualifiers when using the LOGOUT command: /BRIEF or /FULL. A *qualifier* modifies the effects of the command.

If you simply type LOGOUT, the /FULL qualifier is assumed:

```
$ LOGOUT(RET)
```

RSTS/E displays information about the status of your account at the time of logout:

```
1  Saved all disk files; 40 blocks in use, 960 free
2  Job 16 User 52,20 logged off KB22 at 08-Jan-83 06:13 PM
3  System RSTS V8.0 *ROOST*
4  Run time was 3 minutes, 2.4 seconds
5  Elapsed time was 21 minutes
6  Good afternoon
```

The logout information tells you:

(1) The status of your permanent files (Saved all disk files). RSTS/E also shows the use of your file storage space (40 blocks in use, 960 free).

(2) Your job number (16), PPN (52,20), terminal number (KB22), the date (08–Jan–83) and time of logout (06:13 PM).

(3) The name and version of the computer's software (RSTS/E V8.0) and the system name (ROOST).

(4) The time you spent at the terminal ("elapsed" and "run" times).

(5) Any logout messages (Good afternoon).

Notice the difference between run time (3 minutes and 24 seconds) and elapsed time (15 minutes). *Run time* is the amount of time you used the system. (The commands you type require the use of the system's CPU, or central processing unit.) *Elapsed time* shows how long you were logged into the system. Although you were logged in for 15 minutes, you used the computer's CPU for only a fraction of that time.

If you log in and then are called away before you can enter commands, you increase the amount of elapsed time but not of run time. You can be logged in for two hours and not use the system resources at all.

If you include the /BRIEF qualifier after the LOGOUT command, RSTS/E ends your session at the terminal without displaying a message. The command line for using /BRIEF is:

```
$ LOGOUT/BRIEF(RET)
```

Exceeding Your Disk Quota

Sometime in your use of RSTS/E you may create more files than your disk quota allows. A *disk quota* is the amount of file storage space allocated to your account. You cannot log out if you have exceeded your quota; you must first delete one or more files and then try again.

In general, it is a good idea to delete any files you do not need before attempting to log out. This practice saves space on the system. For example, if you created a file named BIRDS.TXT to practice editing text, but you no longer need the file, you can delete it by typing:

```
$ DELETE BIRDS.TXT(RET)
```

If you try to log out and find you have exceeded your *disk quota*, the system displays a message similar to:

```
$ LOGOUT(RET)
Disk quota of 1000 exceeded by 128 blocks
Some file(s) must be deleted before logging out
Type '?' for help
Confirm: ?(RET)
Options for 'Confirm' are:
 ?       This help message
 Y       Log me out
 N       Don't log me out
 I       Individual file deletion
          K to delete
          <CR> to save
 F       Fast logout
Confirm: N(RET)
$
```

In the preceding example, RSTS/E will not let you log out until you free at least 128 blocks of storage space. The Size column in the DIRECTORY command (described in Chapter 3) includes the blocks of storage space each file requires:

```
$ DIRECTORY(RET)

 Name .Type   Size     Prot      Name.Typ      Size      Prot    SY:[52,20]
VANISH.COB      89    < 60>    PATTYC.RNO       130     < 60>
STRIDR.BAS     120    < 60>    ROAD  .MAP       280     < 60>
FLAGS .DAT     159    < 60>    TEST  .MAC       350     < 60>

Total of 1128 blocks in 6 files in SY:[52,20]
```

If you need all your files, you can copy some of them onto magnetic tapes to store them before deleting them from your directory. (See the "Devices" section in Chapter 5 for information about putting files on magnetic tapes.)

Using Commands

RSTS/E allows you to type both DCL (DIGITAL Command Language) and CCL (Concise Command Language) commands. Whereas CCL commands have been on RSTS/E for many years, DCL is more recent. DCL is also available on other DIGITAL systems, but CCL is specific to RSTS/E.

The CCL section in this manual discusses the relationship between DCL and CCL. You can use CCL commands from the DCL environment on RSTS/E.

This manual groups DCL commands into five types of functions:

1. *Working with files* (Chapter 3) — displaying, copying, and printing them. Commands for file operations let you create and edit files, copy or rename them, and perform other related functions.
2. *Using system operations* (Chapter 4) — performing day-to-day system functions. System commands allow you to begin and end a session at the terminal and identify the status of the system.
3. *Working with devices* (Chapter 5) — using the devices on the system. For example, commands for using devices allow you to transfer a file from a disk to a magnetic tape.
4. *Batch processing* (Chapter 6) — using the batch processor to do work without your having to be there. In batch operations you create a batch control file and enter commands to run a batch job.
5. *Developing programs* (Chapter 7) — using BASIC, COBOL, DIBOL, FORTRAN, or MACRO programming languages.

Chapter 2 provides guidelines for using DCL on your system.

Using Privileged DCL Commands

The DCL commands that are useful only to privileged users are not described in this manual. See the *RSTS/E System Manager's Guide* for information on these commands.

Using DCL Commands from Non–DCL Environments

RSTS/E has three keyboard monitors in addition to DCL that allow you to use DCL commands: BASIC–PLUS, RSX, and RT11. If you are in one of the three keyboard monitors, type:

```
DCL [DCL command name]
```

To enter the DCL keyboard monitor, type:

```
DCL<RET>
```

Using DCL Qualifiers

In the DCL environment, you use commands by typing a command name and pressing the RETURN key. You can also use *qualifiers*, which modify the effects of a command, by typing a slash (/) followed by a word the computer recognizes. For example, the LOGOUT command explained earlier has two possible qualifiers: /BRIEF and /FULL.

You type a qualifier after the command name but before pressing the RETURN key. If you do not use qualifiers, DCL makes assumptions known as *defaults.*

For example, when you used the LOGOUT command to end the first terminal session, the /FULL qualifier was the default. You could have typed the /BRIEF qualifier instead, which modifies the effect of LOGOUT by suppressing the display of logout information.

Chapter 2 describes the use of qualifiers in more detail. In addition, the command descriptions in this manual list the qualifiers and defaults available for each command.

Using CCL Commands from the DCL Environment

The Concise Command Language (CCL) allows you to enter a command name rather than type RUN and a program name. You might think of CCL as a shorthand way to run programs. For example, it is faster to type the CCL command SWITCH (abbreviated to SW) than RUN $SWITCH. (Furthermore, with RUN $SWITCH, you have to wait for a prompt.)

Although there may not be any CCL commands on your system, most RSTS/E systems provide them. Certain CCL commands are installed on almost all RSTS/E systems. However, CCLs are still defined at different RSTS/E installations and are not part of DCL. Be aware that some CCL commands you have used may not be valid on all RSTS/E systems.

CCLs are created by the system manager, who assigns a command name to a program. For example, if someone at your site writes a program to balance checkbooks, the system manager can define a CCL command "BANKER" for you to run the checkbook program. Therefore, your system manager can create new commands for special purposes and make these commands available for your use. In addition, your system manager can install individual DCL commands as CCL commands.

You can type CCL commands directly after DCL's dollar prompt ($). The format of the CCL command is defined by your system manager. For details about the use of a CCL command, refer to the documentation written for your site.

When you are using the DCL keyboard monitor, DCL commands take precedence over CCL commands. If your system manager gives a CCL command the same name as a DCL command, you must type the prefix "CCL", a space, and the CCL command itself.

You can use the CCL prefix with any CCL command. This is advisable in batch control files, so that they will continue to function if new DCL commands are added in future versions of RSTS/E. (Batch is described in Chapter 6.) You can also use the CCL prefix when you are in other keyboard monitors.

For example, a CCL command named DIRECTORY and the DCL command DIRECTORY may produce different results, depending on how the CCL command works at your site. To use the CCL version, type:

```
$ CCL DIRECTORY(RET)
```

Maintaining Files

Many of the DCL commands in this manual affect files. It is helpful to understand the relationship between accounts and directories when you work with files.

This section discusses accounts and directories, as well as storing and accessing files.

Accounts, Directories, and Files

An account is what you log in to when you begin a RSTS/E session. Your PPN is an account number, which identifies you to the system.

The system uses your account to keep track of the system resources you use, such as the amount of time you access the computer's memory or are logged in, and the amount of storage space your files require. If you have more than one account on the system, RSTS/E sees each account as belonging to a separate user, who is identified by a PPN.

A directory is a list of the files in your account that are kept on a specific location on a disk. Each directory stores such information as the name and size of each file.

You identify a file by specifying its location and its name. A file's location consists of:

- The node on the network. A "node" is a computer system with DECnet/E, which connects two or more systems together. Nodes and the DECnet/E network are described later in this chapter.
- The device. Files are usually kept either on disks or magnetic tapes. Your files are located on a set of disks called the *public structure*, unless you specify otherwise.
- The directory. A device can be divided up into directories; each directory belongs to a different account.

A file's name, chosen by the person who created the file, consists of:

- The file name (one to six alphanumeric characters)
- The file type (zero to three alphanumeric characters)

The following sections discuss files and directories in more detail.

Putting Files into Your Directory

You can put files into your directory either by moving them from another directory or by creating them. The following example creates a file with the CREATE command to illustrate the placement of a file in a directory. (The CREATE command is explained in Chapter 3.)

Use the CREATE command by typing:

```
$ CREATE(RET)
```

The command prompts you for a file specification:

```
File:
```

You can assign the file almost any name you choose. For this example, name the file ROAD.MAP:

```
File: ROAD.MAP(RET)
```

Enter the text as shown, and press CTRL/Z when you are finished. CTRL/Z is displayed on the terminal as ^Z, but the characters ^Z do not become part of the file. If you make a mistake on a line before you press the RETURN key, you can press the DELETE key to correct it.

```
Although I have been a jogger for(RET)
more than eight years, I did not(RET)
enter any road races until 1978.(RET)
(CTRL/Z)
$
```

The newly created file named ROAD.MAP is now in your directory.

Displaying Your Files

If you are new to a RSTS/E system, you may or may not have files in your directory. Files exist in your directory only if someone put them there; it is possible that the system manager or a privileged user gave you some files. If you created ROAD.MAP in the preceding section, then information about the file appears in a directory listing.

You can check for files in your directory with the command:

```
$ DIRECTORY(RET)
```

If you have no files in your directory, RSTS/E displays a message similar to:

```
%No files matching [52,20]??????.??? - continuing
```

The PPN ([52,20]) identifies your directory, and the question marks (??????.???) show that no files are found.

If there are files in your directory, the system displays information about each file. For example:

```
 Name .Type    Size     Prot      Name.Typ      Size     Prot     SY:[52,20]
FINISH.COB      89    < 60>    TENK   .RNO      130    < 60>
START .BAS      12    < 60>    ROAD   .MAP        3    < 60>
TEMP16.TMP     258    < 60>

Total of 492 blocks in 5 files in SY:[52,20]
```

Your directory provides the following information about each file in your account:

Name.Type	The names and types of your files.
Size	The size of your file in numbers of blocks. (A "block" is 512 characters.)
Prot	The protection code assigned to your file. This determines whether or not other users can examine or modify the file.

SY:[52,20] identifies your directory as being on the system disk (SY:) in your account ([52,20]).

Certain operations (such as compiling and linking programs) cause RSTS/E to create a temporary file with a .TMP file type. Temporary files are scratch files that programs use for work space. The file named TEMP16.TMP in the preceding listing is a temporary file.

Do not display or work with temporary files, because they often contain codes that can make your terminal work abnormally. Temporary files are mentioned only because they use disk space and are displayed in the directory listing; the system deletes them when you log out.

The TYPE command displays the contents of a file. The following example shows the contents of the file ROAD.MAP:

```
$ TYPE ROAD.MAP(RET)
Although I have been a jogger for
more than eight years, I did not
enter any road races until 1978.

$
```

File Specifications

A file specification is the full name and location of a file. In its complete form, the file specification includes:

- A node name on the DECnet/E network, if applicable
- The device name or logical name (see Chapter 5 for descriptions)
- The directory
- The file name
- The file type

The format of a complete file specification is:

node::device:[directory]filename.typ

The delimiters in a file specification are brackets, commas, and colons. Brackets ([]) surround a PPN. A comma (,) separates the two numbers in a PPN. Double colons (::) follow node names, while single colons (:) follow the device name.

You can include a dollar sign ($) in place of a PPN to indicate the directory [1,2], which is the system library. (The *system library* stores most of the system files and programs.) For example, the following command displays a system file named NOTICE.TXT:

```
$ TYPE $NOTICE.TXT
```

A file specification can be as simple as typing:

ROAD.MAP

In this case, RSTS/E assumes your own directory, the host node, and the system disk by default. (The term "host node" is explained in the section named "DECnet/E and You" at the end of this chapter. The command descriptions explain whether or not you can use the command over DECnet/E.)

An example of a file specification that does not assume any defaults is:

ROOST::DR0:[52,20]ROAD.MAP

The parts of the file specification are:

ROOST::	–	The node (system) on the network
DR0:	–	The device at node ROOST:: on which the file is located
[52,20]	–	The directory on the disk at node ROOST::
ROAD.MAP	–	The file name and type

DCL File Specification Restrictions

The rules about file specifications in DCL are slightly more restrictive than elsewhere in RSTS/E.

- The special PPN character "!" is not available in DCL. Instead, the exclamation point is used as the comment delimiter.
- The special PPN characters "@", "%", "&", and "#" are reserved in DCL and may have different meanings in future RSTS/E releases.
- The special PPN character "$" is valid in DCL as elsewhere and designates the system library [1,2].
- Embedded spaces and tabs are not allowed in the file specification in DCL.

(continued on next page)

- The RSTS/E file specification switches /MODE, /SIZE, /POSITION, /CLUSTERSIZE, and /RONLY cannot be used in DCL. Many of the same features are available in DCL, but you must use DCL notation.
- Protection codes in angle brackets ("<nn> ") cannot be used in DCL. The DCL syntax for protection codes is "/PROTECTION:nn".
- Parenthesis ("( )") are reserved characters in DCL and cannot be used to delimit the project-programmer number of a file. Use brackets ("[ ]") instead.
- The single character wildcard "?" is currently available in DCL, but may have a different meaning in a future RSTS/E release.

File Names and Types

When you create a file, you can assign it any name of up to six alphanumeric characters. You can also change the name of a file with the RENAME command.

A file type consists of one to three alphanumeric characters that follow a file name. There is a period between the file name and file type. File types provide you with a shorthand to identify a file's contents. For example, the file STATUS.DOC has a file name of STATUS and a file type of .DOC.

You can assign any alphanumeric file name and type in a file specification, although it is a good idea to use a "mnemonic" name. This means that you should assign the file a name that reminds you of its contents. For example, TEST.DAT could be a data file (.DAT) that you create to test a program (TEST). If you have a file containing a BASIC–PLUS program to balance your checkbook, you might name it CHECK.BAS. The file type .BAS indicates a BASIC–PLUS source file. (A "source" file contains the program you typed in.)

An advantage of using a mnemonic file type is that RSTS/E recognizes several file types. If you assign a file one of these types, then you do not always have to type it explicitly; many commands assume it by default.

For example, if you are using the COBOL command to compile a program, you can type:

```
$ COBOL CLIFF<RET>
```

The source file which results is assumed to be CLIFF.CBL, because the file type .CBL is the default for a COBOL source file.

Table 1–2 lists some common file types on RSTS/E.

Table 1–2: RSTS/E File Types

File Type	Meaning
.B2S	BASIC–PLUS–2 source file
.BAC	BASIC–PLUS compiled file
.BAK	Backup file created by several types of utility programs
.BAS	BASIC–PLUS source file
.CBL	COBOL source file
.CMD	Command file
.CRF	Cross reference listing file
.CTL	Batch control file
.DAT	Data file
.DBL	DIBOL source file
.DIF	DIFFERENCES output file
.DIR	File produced by the DIRECTORY command
.DOC	RUNOFF output file
.EDT	EDT initialization file
.FLB	FMS–11 form library file
.FOR	FORTRAN–IV source file
.FTN	FORTRAN–77 source file
.HLP	System program help text file
.LIB	Resident library file
.LNK	Input file used by the DCL LINK command
.LOG	Batch output log file
.LST	Listing file
.MAC	MACRO source file
.MAI	DECmail/RSTS mailbox file
.MAP	Map file created by the LINK command or Task Builder
.MLB	MACRO library file used by the RSX–based MACRO assembler
.OBJ	Object file, which is a compiled BASIC–PLUS–2, COBOL, DIBOL, FORTRAN, or MACRO program
.ODL	Overlay Description Language input file used by the Task Builder
.OLB	Object module library file used by the Task Builder
.PMD	Post-Mortem Dump file
.RNO	RUNOFF input file
.RTS	Run-time system

(continued on next page)

Table 1–2: RSTS/E File Types (Cont.)

File Type	Meaning
.SAV	Executable FORTRAN–IV or MACRO program produced by the LINK command or the RT11–based LINK program
.SIL	Save Image Library (RSTS/E monitor or system program)
.SKL	Skeleton file produced by the COBOL–81 compiler
.SRT	SORT–11 file
.STB	Symbol table file produced by the Task Builder
.SYS	System file, usually for internal use
.TMP	Temporary file created by a system program (deleted when you log out or if the disk is rebuilt)
.TSK	Executable BASIC–PLUS–2, COBOL, DIBOL, FORTRAN–77, or MACRO program produced by the LINK command or the Task Builder
.TXT	ASCII text file

Protection Codes

Protection is a system feature that lets you determine who, if anyone, can have access to the file you are creating. RSTS/E displays the protection codes of files between angle brackets (< >) in your directory listing.

For example, 60 is the default protection code that RSTS/E assigns. (On some systems, the system manager may have chosen a different default protection code.) If one of your files has a protection code of 60, only you or a privileged user can read, edit, or delete it. A protection code remains fixed until you change it.

The section "Allowing Access to Your Files" in Chapter 3 lists the available protection codes and describes how to assign them.

Wildcards

Wildcards let you refer to a group of files all at once by specifying a pattern of identification. You can use wildcards to print all files that have the same name. The DCL wildcard symbol is an asterisk (*). For example, if you have the files TIMING.COB, TIMING.BAS, and TIMING.DAT in your directory, you can display information about them by typing:

```
$ DIRECTORY TIMING.*RET
```

Running a Program

You run most programs by typing RUN and entering the name of the program. A program that can be run is called an "executable" program. (Chapter 7 includes the DCL commands for running, compiling, and linking programs on RSTS/E. For more information about each programming language, refer to the appropriate language manual and user's guide.)

Some programs are referred to as *user programs*, because they are created by a system user. For example, if you create a BASIC–PLUS program named BANKER that balances a checkbook, you can run the user program BANKER.BAC by typing:

```
$ RUN BANKER.BAC(RET)
```

Other programs are referred to as *system programs*, because they are installed as part of the system software. You can run system programs with one of the following:

1. A DCL command. For example, you run the system's editor program (EDT) by typing EDIT and pressing the RETURN key. EDIT is the DCL command that runs the EDT program.
2. A CCL command (if it is available on your system). If your system manager installs a CCL command, such as QUEUE to print files, you can run the QUEUE program from the DCL keyboard monitor by typing:

   ```
   $ QUEUE(RET)
   ```

3. The RUN command. For example, you can switch to DCL from another keyboard monitor by typing:

   ```
   RUN $SWITCH(RET)
   ```

DECnet/E and You

The DECnet facility links two or more DIGITAL computer systems as a network. DECnet refers to both the hardware (machinery) and software (programs) involved in network operations. The DECnet software available on RSTS/E is called DECnet/E.

This section discusses what DECnet/E is and what it means to you as a DCL user. For a complete description of the capabilities that DECnet/E has to offer, refer to the *Introduction to DECnet* and to the manuals in the DECnet/E documentation set.

If your system has DECnet/E, then your system is part of a network that allows you to perform file, system, and programming operations on another system. To find out if your system has DECnet/E, type the following:

```
$ SHOW NETWORK(RET)
```

You should get one of the following responses:

1. If the command displays network information, as described in the next section, you can use DECnet/E from your system.
2. If the network is on your system but is not currently in operation, the following is displayed:

```
$ SHOW NETWORK(RET)

                              Active
Node            State         Links          Delay          Circuit

No information

$
```

3. If your system does not have DECnet/E, you get the message:

```
?Command not available
```

Note

If your system does not have DECnet/E, then you can skip the rest of this chapter and proceed to Chapter 2.

Some of the commands in this manual can be used over the network. The commands and their qualifiers that can apply to network operations are labeled with the symbol (N) in the command descriptions.

DECnet/E also allows you to log in to another computer system from your account. You can then use that system as though your terminal were connected to it directly. You log in to another node (system) on the network with the SET HOST command, which is described later in this chapter.

To log in to another system on the network, you need to have an account and password for that system. In addition, the network between your system and the other system must be working. (You use the SHOW NETWORK command, described in the next section, to check for available nodes on the network.)

You should know the following terms to become familiar with DECnet/E:

Network — The family of software modules, files, hardware components, and facilities that ties different DIGITAL systems together.

Node — A computer system that is connected to other computers by a network. Node names are assigned by the system manager. The following are different types of nodes:

- Host — The node you access with the SET HOST command.
- Local — The node you logged in to before using the network; your original system.
- Remote — Any node in the network other than the local node.

To summarize, the *node* of the *network* you first log in to is *local*, and the system you reach by the network is *remote*. You connect to another system by specifying the *host* and then logging in. If you want to communicate with a system on the network from your local node, you can use DCL commands over DECnet/E without your having to log in to the other system.

Displaying Network Status: SHOW NETWORK

The SHOW NETWORK command displays the systems you can connect to on the network. You use the command by typing:

```
$ SHOW NETWORK(RET)
```

If the network is in operation, RSTS/E displays the names of different nodes your system can access.

For a complete description of each element you may find in the SHOW NETWORK display, refer to the *DECnet/E System Manager's Guide*, under the SHOW ACTIVE NODES command of NCP (Network Control Program).

```
$ SHOW NETWORK(RET)

                          Active
Node          State       Links          Delay          Circuit

37 (FRODO)    Reachable                                 DMC-5
```

The display tells you that node 37 (the node named FRODO) can be reached on the network.

If you cannot use the network, no nodes are displayed, and the following message is printed:

```
No information
```

Using the Network: SET HOST

The SET HOST command lets you log in to another computer from the system you are currently logged in to. You use the command by typing:

```
$ SET HOST(RET)
```

RSTS/E prompts you for the node name with:

```
Node:
```

Enter the name of a node that is available on the network. For example, you connect to a node named SPEEDY by responding:

```
Node: SPEEDY(RET)
```

The node SPEEDY is remote because you are connecting to it over the network.

You can instead type SET HOST followed directly by a node name and press RETURN. For example:

```
SET HOST = SPEEDY::
```

If you include the node name in the same command string with SET HOST, you must include either an equal sign or a space between the command SET HOST and the remote node you specify.

After the connection is made, the local node displays a message stating that the connection has been established. You must have an account on the remote node to log in. When the connection is established, you use the normal log-in dialogue for that system. When you successfully log in, you can use the remote node essentially as if your terminal were connected to it directly.

When you are finished, use the LOGOUT command to log out from the remote node. After approximately five seconds, control returns to the local node, with a message similar to:

```
Link to node SPEEDY disconnected
```

There are two ways you can end your operations on the remote node and return directly to the local node. The first way, which is recommended for normal use, is to follow the host system's normal log-out procedure. This deletes any temporary files you may be using.

The second way is to type CTRL/P, which prompts you for a network command:

```
(CTRL/P)
SPEEDY::NET> EXIT(RET)
```

The possible responses to the prompt are described in the *DECnet/E Guide to User Utilities*. The response shown above, EXIT, returns you to the local node.

Network File Specifications

Remote file specifications may follow different rules from local file specifications. These rules are mostly determined by the remote node, which may or may not be running RSTS/E.

The host DCL does not allow:

1. Exclamation points, parentheses, commas (except in angle brackets or square brackets), plus signs, hyphens, double colons, question marks, slashes, apostrophes, or double quotes.
2. Unbalanced or nested pairs of angle brackets or square brackets.
3. Embedded spaces.
4. Equal signs.

If you need to embed any of this punctuation into the file specification, you can quote part or all of it beyond the double colon. You quote it with the double quotation mark (") character, not the apostrophe ('). To embed a quote character in a quoted string, use a pair of quote characters.

You can include accounting information in the remote file specification in the format:

node"user password account"::file-spec

A single space is required between the user, password, and account. "Account" is not used by RSTS/E; you can omit it in such cases. If you do not include an accounting string, the system prompts you with:

```
User:
Password:
Account:
```

If you specify several input files in one command, all of the files must be on the same node. In addition, you must specify the node name for each input file specification you include, otherwise, you will receive the message, "?Files cannot be on different nodes." For example, if you want to delete two files on node APS, type:

```
DELETE APS::ARMAN.DOS, APS::JOKE.TXT
```

Using DCL Commands 2

This chapter provides rules that apply to DCL commands. You may want to read this chapter now to understand what kinds of rules apply to DCL, or you may choose to refer to this chapter as needed when you use individual commands. For example, several commands allow you to include a date or time in the command line, but you need to know how to specify dates and times. However, not all commands require you to know these rules.

This chapter explains how to enter:

- Commands
- File specifications
- Qualifiers
- Character string data
- Numeric values
- Date and time values

Understanding Command Formats

New users often feel threatened when they first encounter command formats. However, interpreting the command formats is easy when you learn the terminology and understand the syntax elements. This section describes command formats and how to use them.

The commands in this manual are grouped by related functions. (For example, commands that involve files are described in Chapter 3.) Each command description begins with a brief description of the command's function. Next is syntax information, which is enclosed in a rectangle. An example of the simplest form of the command follows. After this is information about each part of the syntax and examples. (You may want to flip through the commands in Chapters 3 to 7 to see how this works before you continue reading.)

The command format descriptions in this manual use the terms in Table 2–1.

Table 2–1: Terms Used in Command Formats

Term	Meaning
Format	The use of the components in a command string.
Qualifiers	The components that modify the effect of the command string in some way.
Command Qualifiers	Qualifiers that modify the command itself.
File Qualifiers	Qualifiers that modify the treatment of a file included in the command string.
Defaults	The assumptions DCL makes when you enter the command string and do not use all the available parts of the format.
Prompts	The text that DCL displays to request your input.
Remote	The parts of the command string that are allowed over the network.
Command Parameters	The parts of a command string that specify what the command is operating on. (For example, what file is affected by the command.)
file-spec	A file specification.
input file-spec	The specification of a file to be used as input in the command string.
output file-spec	The specification of a file to be used as output in the command string.

For example, the command format in Figure 2–1 lists the syntax elements for using the COPY command. (The complete command description is in Chapter 3.)

Duplicating, Renaming, and Appending Files

The commands described in this section let you duplicate, rename, and append the contents of files.

COPY

The COPY command duplicates one or more existing files, or concatenates two or more files.

Format

COPY input-file-spec[,...] output-file-spec

Qualifiers	Defaults
/ALLOCATION=n	
/BLOCK_SIZE=n	/BLOCK_SIZE=512
/CLUSTER_SIZE=n	
(N) /[NO]CONTIGUOUS	
(N) /[NO]LOG	/LOG
/[NO]OVERLAY	/NOOVERLAY
/PROTECTION=n	
(N) /[NO]QUERY	/NOQUERY
(N) /[NO]REPLACE	

Prompts

From: input-file-spec[,...]
To: output-file-spec
OK to replace existing file file-spec?

You can use COPY to:

- Copy one file to another file
- Merge (*concatenate*) more than one file into a single file
- Copy a group of files to another group of files

For example, if you have a file named SMITH.MAR and you want to make a copy of it as file JONES.JOE, you type:

```
$ COPY SMITH.MAR JONES.JOE(RET)
```

Figure 2–1: The COPY Command Format

Now, suppose you want to use COPY to duplicate all the files named JOKES and GAMES in your account to a file named PARTY.TYM on disk DB2: on the system named SPEEDY. (SPEEDY is a possible node name on a network of computer systems. See the section named "DECnet/E and You" at the end of Chapter 1 for information about nodes on DECnet/E.)

In the following example, you specify that the files should not supersede existing files, and that you want to see a message when the files are successfully copied over. Therefore, you include the following information in the command line:

- A command qualifier (/LOG)
- A file qualifier (/NOREPLACE)
- Two input file-specs (JOKES.* and GAMES.*)
- One output file-spec (SPEEDY::DB2:PARTY.TYM)

The dialogue at your terminal looks like:

```
$ COPY/NOREPLACE/LOG↵
From:   JOKES.*,GAMES.*↵
To:     SPEEDY::DB2:PARTY.TYM↵
File JOKES.1 copied to SPEEDY::DB2:[2,212]PARTY.TYM
File JOKES.2 copied to SPEEDY::DB2:[2,212]PARTY.TYM
File GAMES.TXT copied to SPEEDY::DB2:[2,212]PARTY.TYM

$
```

Note that the example takes advantage of system prompting, and that DCL displays its prompt when the COPY operation is complete.

Entering Commands

The complete specification of a command, including the command name, command qualifiers, parameters, and file qualifiers (if any), is called a *command string*. Because you can continue a command on more than one line, the term command string is used to define the entire command that is passed to the system. By contrast, the term *command line* describes the part of a command string that is typed on one physical line.

The general format of a command string is:

$ command-name[/qualifiers...] parameter[/qualifiers...][...]

You can precede any command string with a dollar sign character ($). In interactive mode the command interpreter ignores the dollar sign. (*Interactive mode* is when the system prompts for your input.) However, the dollar sign is required in batch mode. (*Batch mode* is when you have the system process a series of commands for you. Batch processing does not require your input.)

Each item in a command must be properly delimited, as follows:

- At least one blank or tab character must separate the command name from the first parameter, and at least one blank must separate each additional parameter from the previous parameter. You can use multiple blanks and tabs in all cases where a single blank is required.
- Each qualifier must be preceded by a slash (/). You can enter any number of blanks or tabs before or after the slash.

The qualifiers you use cannot conflict. For example, you cannot specify both /FULL and /BRIEF with the DIRECTORY command. (/FULL requests a complete listing; /BRIEF requests a listing with only file names and types.) If you try, you get an error message, and the command is not executed:

```
$ DIRECTORY/FULL/BRIEF<RET>

%Conflicting qualifiers.
$
```

Continuing Commands on More than One Line

The maximum number of characters you can enter on one line is 132. However, you can enter a command string on more than one line by using the continuation character, a hyphen (–), as the last element on a command line.

Command line continuation is especially useful when you enter a command and want to specify many qualifiers, or when you place a command in a batch control file and want to make the procedure more readable. (You create a *batch control file* to have the system process a series of commands for you.) For example:

```
$ PRINT MYFILE -<RET>
/AFTER=17:00 -<RET>
/COPIES=20 -<RET>
/NAME=GUIDO<RET>
```

The technique used in the command file allows you to break any keyword, qualifier, or parameter over multiple lines. Note that when you continue a command, you must still provide the required space before each parameter.

There is no restriction on the number of continued lines you can use to enter a command. However, the total number of characters in a command string must not exceed 255.

Many DCL commands run utility programs. These commands are translated to form command strings for the utilities. The maximum length of these lines, after they have been translated, is 80 or 127 characters, depending on the utility. The message "%Command too long" appears if you exceed this limit.

Entering Comments

Comments are used to explain or document commands or files. You indicate a comment by preceding it with an exclamation mark (!). For example, the following line in a command file uses a comment to explain the use of the DIRECTORY command in a control file:

```
$ DIRECTORY W:[*,*] !W: is a logical name for disk DB2:
```

Everything from the exclamation point to the end of the line is a comment, which does not affect the use of the DIRECTORY command.

Comments are valid in the following positions:

- As the first item on a command line; in this case, the entire line is considered a comment and is not processed.
- Following the last character in a command string, or after a hyphen that signals continuation in a command line.

Some examples of valid placement of comments follow:

```
$ !THIS ENTIRE LINE IS A COMMENT(RET)
$ PRINT MYFILE - ! PRINT COMMAND COMMENT(RET)
Continue: /COPIES=3 ! 3 COPIES, PLEASE(RET)
```

The word "Continue:" in the last line is printed by the system to show that what follows is a continuation line.

When you continue a command on more than one line, the command interpreter uses the Continue: prompt to indicate that it is still accepting the command. Note that the hyphen to continue the command must precede the comment. Hyphens after the exclamation point (!) are ignored. This means that comments cannot be continued, although the next line could consist of:

```
$ - ! Comment included,(RET)
```

Abbreviating Keywords

Many keywords that you enter as command input can be abbreviated by truncating characters on the right. You can abbreviate:

- Command names
- Command keyword parameters
- Qualifiers
- Qualifier keyword values

Abbreviating Command Names

You can abbreviate all command names to the first four letters. You can abbreviate command names to two characters as long as the abbreviation is unique. For example, the PRINT command is the only command that begins with the characters "PR". Therefore, the PRINT command can be abbreviated to two characters. The DEALLOCATE and DEASSIGN commands, however, have the same first three characters, so these commands cannot be truncated to fewer than four characters.

You can protect yourself from possible future release changes by keeping to a four character minimum. (This is mostly relevant to batch control files, rather than interactive use.)

Abbreviating Parameters, Qualifiers, and Qualifier Arguments

You can abbreviate parameters, qualifiers, and qualifier arguments to two characters, unless they conflict with other elements. For example, /SIZ is the minimum abbreviation of /SIZE; fewer than three characters conflicts with /SINCE. /SIZ is the minimum abbreviation wherever it is used with a DCL command.

Some qualifiers permit a negative form. For example, /NOJOURNAL is the negative form of the /JOURNAL qualifier. In applying the minimum four-character truncation rule, do not count the NO prefix as the first two of the four characters. In this case, the minimum truncation that guarantees uniqueness is /NOJOUR. The slash character (/) is also disregarded in counting characters.

The underscore (_), as in the /[NO]D_LINES qualifier, is considered optional syntax. This means that the minimum abbreviation for guaranteed uniqueness is /D_LIN and /NOD_LIN.

Abbreviations in Batch Control Files

Special considerations apply when you type commands in batch control files. It is recommended that you use the full names of all the above components to ensure readability. If you abbreviate any of these items, you should not abbreviate to fewer than four characters. If you do, your batch control file may not be compatible with future releases of RSTS/E.

Entering File Specification Lists

An input file parameter for many commands has the format:

file-spec[,...]

This format indicates that you can enter more than one file specification. You can separate the file specifications with commas (,) or plus signs (+).

Any number of blanks or tab characters can precede or follow the commas or plus signs. The list of file specifications is treated as a single parameter.

Entering Job Specification Lists

An input job specification parameter, generally used with the spooling commands, has the format:

queue-name:[proj,prog]job-name

This format indicates the following:

1. The queue in which the job is to be placed.
2. The project-programmer number assigned to the job. If this field is not specified, your PPN is assumed. Only privileged users may specify a PPN different from their own.
3. The name of the job. Job names need not be unique. You can submit several jobs with the same name and project-programmer number.

Entering Qualifiers

Commands can take one or both of the following types of qualifier:

- Command qualifiers
- File qualifiers

Command qualifiers have the same meaning regardless of whether they appear after the command name or after a command parameter. For example:

```
$ PRINT/QUEUE=LP1: SPRING.SUM,FALL.SUM<RET>
$ PRINT SPRING.SUM,FALL.SUM/QUEUE=LP1:<RET>
```

The /QUEUE qualifier is a command qualifier; therefore, the two PRINT commands shown above are equivalent.

Unlike command qualifiers, file qualifiers can have different meanings depending on where you place them in the command line. If specified immediately after a file specification parameter, they affect only the file they follow. If specified after the command name, they affect all files specified as parameters.

For example:

```
$ PRINT/COPIES=2  SPRING.SUM,FALL.SUM<RET>
$ PRINT  SPRING.SUM/COPIES=2,FALL.SUM<RET>
```

The first PRINT command requests two copies of each of the files SPRING.SUM and FALL.SUM. The second PRINT command requests two copies of the file SPRING.SUM, but only one copy of FALL.SUM.

Some file qualifiers must be specified after a parameter and cannot be specified following the command name. (An example is the /LIBRARY qualifier used with the MACRO command.) When this is the case, the qualifier description indicates that the qualifier is associated with a file parameter. In all other cases, if you specify a file qualifier following the command name, the qualifier is applied to all files specified.

Determining Qualifier Defaults

Most of the command formats in this manual show defaults. This means that when you do not explicitly include a qualifier in a command line, a qualifier is assumed. For example, when you copy a file, DCL normally displays a message confirming that the COPY operation was successful. If you type /NOLOG in the command line, you suppress the message. Otherwise, DCL assumes the /LOG qualifier and displays the message.

In some cases, a default is not applicable, so the term "N/A" is included in the default column in command descriptions.

Entering Qualifier Arguments

Many qualifiers can be followed by a keyword, file specification, character string, or numeric value. For keywords, follow the rules for truncating keywords; for other types of values, follow the rules for entering file specifications, character data, or numeric values.

You must separate a qualifier and value with either an equal sign (=) or colon (:). For example, the following specifications are equivalent:

```
/OUTPUT=DB1:NEW.DAT          /OUTPUT:DB1:NEW.DAT
```

Many qualifiers accept one keyword or variable value. The syntax is represented in the manual as:

```
/qualifier=value
```

For example:

```
/COPIES=3
/DATE=MODIFIED
```

Entering Output File Qualifiers

Some qualifiers request output from a command and optionally accept a file specification value. For example, the /LIST and /OBJECT qualifiers for the compilers, as well as the /EXECUTABLE and /MAP qualifiers for the LINK command, are output file qualifiers that fit into this category.

1. If the qualifier is present by default, the output file specification defaults to your directory on the public disk structure and the name of the first input file. The qualifier provides a default file type. Some examples are:

Command	Output File
LINK A	A.TSK
LINK A,B	A.TSK
LINK [52,20]A,[52,20]B	A.TSK
LINK A.OBJ	A.TSK

2. If the qualifier does not specify an output file specification, the output file specification defaults to your account on the public disk structure and the name of the first input file. The qualifier provides a default file type. Some examples are:

Command	Output File
LINK/EXECUTABLE A	A.TSK
LINK/EXECUTABLE A,B	A.TSK
LINK/EXECUTABLE A.OBJ	A.TSK

3. If you specify /LIST without a file specification, the listing goes where the object file goes, but it has an .LST file type. If you do not specify an object file, then the listing has the same name as the first input file, an .LST file type, and is in your directory on SY:.

4. If you specify /MAP without a file specification, the map goes where the executable file goes, but it has a .MAP file type. If you do not specify an executable file, then the map has the same name as the first input file, a .MAP file type, and is in your directory on SY:.

5. If the qualifier indicates a file specification for the output file, then any parts entered in the file specification are used to name the output file, and no default file name is supplied. Some examples are:

Command	Output Files
LINK A,B/EXECUTABLE=C	C.TSK
FORTRAN/LIST=A B+C	A.LST, B.FOR
LINK/EXE=[52,20]A	[52,20]A.TSK

Entering Uppercase, Lowercase, and Nonalphanumeric Characters

When you enter commands, you can use any combination of uppercase and lowercase letters; both are treated the same.

DCL treats a string of blanks and tabs the same as a single blank. Blanks and tabs are ignored at the beginning or end of the command string, around commas and plus signs, around equal signs, and around colons that delimit qualifier values. They are not permitted in file specifications, dates and times, keywords, or numbers.

Quoted strings are allowed in the REQUEST command and in network file specifications. A *quoted string* consists of characters enclosed in quotation marks, to be interpreted literally.

Table 2–2 lists nonalphanumeric characters that have special meanings in DCL.

Table 2–2: Nonalphanumeric Characters

Symbol	Name	Meaning
:	Colon	1. Device name delimiter in a file specification. 2. Node name delimiter; used as a double colon (::). 3. Qualifier value delimiter; separates a qualifier name from its value.
/	Slash	Qualifier delimiter.
+	Plus sign	List element separator for parameters.
,	Comma	List element separator for parameters.
–	Hyphen	Continuation character. As the last nonblank character in a command string, indicates that a continuation line follows. If the statement contains a comment, the hyphen must be the last nonblank character before the exclamation point.
[ ]	Square brackets	Directory delimiters in a file specification.
?	Question mark	Reserved special character.
\	Back slash	Reserved special character.
=	Equal sign	Qualifier value delimiter; separates a qualifier name from its value.
^	Circumflex	Reserved special character.
#	Number sign	Reserved special character.
*	Asterisk	Wild card character in a file specification.
.	Period	File type delimiter in file specifications.
!	Exclamation point	Comment delimiter.
"	Quotation mark	Literal string delimiter.
$	Dollar sign	Batch control file character. Used as the first character in first position of a control statement; causes recognition by DCL of control statement.
		In a file specification, denotes the directory [1,2], which is the system library.
	Space	Separates fields in a command string. Otherwise ignored unless embedded in a string delimited by quotation marks.
	Tab	Separates fields in a command string. (Equivalent to one space (blank) character; otherwise ignored.)

Entering Numeric Values

The command interpreter treats all numbers as decimal integers.

Entering Dates and Times

Some DCL commands allow you to specify a date or time when the command will take effect. For example, you can print ten copies of a file printed after 5:00 PM on a weekend, when fewer people will need the line printer, by typing a line such as:

```
$ PRINT NEWPRT.MAR/COPIES=10/AFTER=1-NOV-82:5:00PM↵
```

The following sections explain how to specify dates and times.

The /BEFORE, /SINCE, and /AFTER Qualifiers

You can give a date with any command that allows the /SINCE or /BEFORE qualifier. (Generally these commands are used for working with files.) /SINCE=date affects files that are created or modified since the specified date, depending on whether or not you also use the /CREATED or /MODIFIED qualifier. Files created on the specified date are affected.

/BEFORE=date affects files that are created or modified before the specified date, depending on whether or not you also use the /CREATED or /MODIFIED qualifier.

You can use the /SINCE and /BEFORE qualifiers together to provide a range of dates for file operations.

The /AFTER qualifier allows you to specify a date, a time, or both. /AFTER is used with commands that affect jobs in a batch or print queue. You use /AFTER to indicate that the system should not begin executing the batch job, or print the file, until after a specified time.

Date Formats

You can specify any date from 1–JAN–70 to 31–DEC–99, inclusive. If you specify a date outside that range, you get the message:

```
%Invalid date
```

There are three formats for dates:

1. The alphanumeric format is dd-mmm-yy, as in 23–MAR–83, where dd is the date of the month; mmm is the first three letters of the month; yy is the last two digits of the year, or the entire year (as in 1983). The yy is optional; you can just type dd-mm. If you omit the year, the current year is assumed.
2. The numeric format is yy.mm.dd, as in 83.3.23 (for March 23, 1983). yy is the last two digits of the year, or the entire year (as in 1983). mm is the month, and dd is the date of the month.
3. The keywords you can use to specify a date are TODAY, TOMORROW, or YESTERDAY.

Time Formats

There are two formats for times: 24–hour time and AM/PM time.

1. A 24–hour time has the form hh:mm, where hh is the hour, in the range 0 to 23, and mm is the minute, in the range 0 to 59. The minute is optional. The default minute is 00. For example, 11 is equivalent to 11:00. The limits of time formats are as follows:

   00:00 is midnight, and is considered the beginning of the specified date.

   12:00 is noon.

   24:00 is midnight, and is considered the end of the specified date.

2. An AM/PM time can be in the format hh:mmAM, hh:mmPM, or 12:00M. The hour, hh, is in the range 1 to 12. The minute, mm, is in the range 0 to 59. M (for meridian) stands for noon. You can use either 00:00 or 24:00 to specify midnight. (The notations 12:00AM and 12:00PM are not recommended.)

   Do not include a space between the minute and AM, PM, or M.

Combinations of Dates and Times

The order for specifying both a date and time is date:time. (Use the formats described earlier for each.) For example:

23–MAR–83:11:00AM

TOMORROW:1:00AM

If you specify just a time, TODAY is assumed. If you specify just a date when using the /AFTER qualifier, then 11:59PM is assumed. (In other words, the job is processed after the end of the specified day.)

Syntax

The punctuation marks in a time specification indicate the time value you enter. If you specify both the date (dd-mmm-yyyy) and the time (hh:mm), you must type the colon between the date and the time.

A line ending with a hyphen (-) is continued, rather than recognized as a part of the date. For example:

```
$ PRINT/AFTER=15-
  CONTINUE:RET
```

The PRINT command specifies 3:00 PM today, according to 24–hour time.

Examples

Some examples of specifying absolute times are:

Time Specification	Result
8–JUN–1982:12	2:00 noon on June 8, 1982
15	3:00 p.m. today (or 1500 hours)
18:30	6:30 p.m. today

Working with Files 3

This chapter describes DCL commands that are used in file operations. The commands are presented in the order they appear in Table 3–1.

Table 3–1: Commands for File Operations

Creating and Modifying Text Files	
CREATE	Allows you to enter text that you can save as a file. (See page 3–3.)
EDIT	Starts the EDT text editor. (See page 3–6.)
Displaying File Names and Files	
DIRECTORY	Displays information about files in a directory. (See page 3–11.)
TYPE	Displays the contents of individual text files at the terminal. (See page 3–19.)
Deleting Files	
DELETE	Deletes files from a directory. (See page 3–22.)
Copying, Renaming, and Appending Files	
COPY	Duplicates a file or concatenates files. (See page 3–26.)
RENAME	Assigns a new name to a file. (See page 3–35.)
APPEND	Copies a file to the end of another file. (See page 3–38.)
Printing Files	
PRINT	Produces a listing, or "hard copy," of a file. (See page 3–41.)
SHOW QUEUE	Displays the names of files to be printed or files awaiting batch processing, as well as files that are being processed. (See page 3–48.)
SET QUEUE/JOB	Uses the name of a job to modify the status of a file that is queued for a printer or batch queue. For example, a SET QUEUE/JOB qualifier can specify the time at which a file is printed. (See page 3–52.)
SET QUEUE/ENTRY	Same as SET QUEUE/JOB, except that you use the sequence-number of the job instead of the job-name. (See page 3–55.)
DELETE/JOB	Uses the name of a job to cancel a request to the line printer or batch queue. (See page 3–57.)
DELETE/ENTRY	Same as DELETE/JOB, except that you use the sequence-number of the job instead of the job-name. (See page 3–59.)

(continued on next page)

Table 3–1: Commands for File Operations (Cont.)

Comparing Files	
DIFFERENCES	Creates a new file that lists differences in the text of two other files. DIFFERENCES is often used to compare earlier versions of a file with its later versions. (See page 3–61.)
Allowing Access to Your Files	
SET PROTECTION	Specifies the protection code of a file to determine who, if anyone, can read, modify, or delete a file. The /DEFAULT qualifier assigns the same protection code to all the files you create in a session at the terminal. (See page 3–68.)

Creating and Modifying Text Files

The CREATE and EDIT commands let you create files in DCL.

CREATE

The CREATE command allows you to enter text and save it as a file.

Format

CREATE file-spec

Qualifiers	**Defaults**
/ALLOCATION=n	/ALLOCATION=0
/CLUSTER_SIZE=n	
/[NO]CONTIGUOUS	/NOCONTIGUOUS
/PROTECTION=n	
/[NO]REPLACE	

Prompts

File: file-spec
OK to replace existing file filespec?

You can use the DELETE and CTRL/U keys to delete text on the current line. To save the text as a file, you press CTRL/Z . For example:

```
$ CREATE PACE.DAT<RET>
I was exhausted after my<RET>
first speedwork session.<RET>
Nine-minute miles seemed<RET>
awfully fast.<RET>
<CTRL/Z>

$
```

The CREATE command is especially useful when you want to preallocate the size of a file or create a file from a batch job. (See Chapter 6, "Batch Processing," for information about batch.) There is less overhead when you *preallocate* the size of a file, because the system assigns enough space for the file only once, rather than having to assign more space several times as the file grows larger.

Because preallocation causes less overhead, access to the file is faster. Allocating the size of a file is not necessary, but it is occasionally useful if a program will be using the file. Preallocation avoids possible overhead caused when the file size is extended by the program's use.

You also preallocate the size of a file if you need a contiguous file of a specific size. You must specify /ALLOCATION with the /CONTIGUOUS qualifier.

It is not necessary to create or preallocate the size of a file before using it as an output file for some other operation.

Command Parameters

file-spec
Specifies the name of a file to be created. No wildcard characters are allowed in the file specification.

Qualifiers

/ALLOCATION=n
Creates space on the disk by forcing the initial allocation of the file to a number of 512–character blocks, which you specify with n. The following example provides three blocks of space to create a file named BOSTON.MAR:

```
$ CREATE BOSTON.MAR/ALLOCATION=3
```

/CLUSTER_SIZE=n
Establishes the cluster size for a disk file. A cluster is a number of contiguous blocks taken together as a unit. The cluster size is the minimum unit of allocation for the file.

This qualifier is especially useful for large files because specifying a large cluster size speeds up random access to the data.

You can also avoid filling up your directory by specifying a large cluster size. (In your directory, the system maintains an internal list of the clusters in each file. If your directory fills up, you will receive the message, "?No room for user on device" even though the disk is not full.)

RSTS/E allows cluster sizes of 1, 2, 4, 8, 16, 32, 64, 128, and 256 blocks. With the SHOW DEVICES command, you can display the minimum cluster size of each disk. If you specify a cluster size below the minimum, the system will use the disk's minimum and will not display an error message. Note that the minimum set for each disk might vary.

/CONTIGUOUS
/NOCONTIGUOUS
Indicates whether or not the file is to be contiguous. (*Contiguous* means that the file occupies consecutive physical disk blocks.)

For example, the following command line creates the file SMOKER.YUK as a contiguous file:

```
$ CREATE SMOKER.YUK/CONTIGUOUS
```

If you do not specify an allocation, or if the allocation is too small, you get the following error message when the system tries to increase the size of the contiguous output file:

```
?Protection violation
```

/PROTECTION=n

Specifies the protection code of the file.

RSTS/E normally assigns files a protection code of 60, unless your system manager has changed the default. Furthermore, if you use the SET PROTECTION/DEFAULT command before creating the file, the protection code becomes the value you specified as the default. (The SET PROTECTION command is described at the end of this chapter.)

/REPLACE
/NOREPLACE

/REPLACE requests that if a file already exists with the same file specification as the one you are creating, the existing file is to be deleted.

For example, if there is a file named BLISTR.DAT in your directory, and you want the file you are creating to replace the existing file, type:

```
$ CREATE BLISTR.DAT/REPLACE(RET)
```

If the CREATE command would replace an existing file, but you do not specify /REPLACE or /NOREPLACE, DCL displays the prompt:

```
OK to replace existing file?
```

You can reply with one of the following:

Y – Yes, replace the file
N – No, do not replace the file
(RET) – No, do not replace the file

A yes response allows you to create the file and replace the existing file. Otherwise, you are returned to command level.

For example, assume you inadvertently enter the name of an existing file that you do not want replaced. The display at the terminal reads:

```
$ CREATE CALLUS.BAK(RET)
OK to replace existing file DR1:[52,20]CALLUS.BAK? N(RET)
```

/NOREPLACE forces the CREATE command to fail if the file being created already exists. In addition, it suppresses the "OK to replace existing file" prompt.

If you use the CREATE command in a batch control file, you should explicitly specify either /REPLACE or /NOREPLACE. This prevents DCL from issuing the "OK to replace existing file" prompt.

EDIT

The EDIT command starts the EDT editor program, which lets you create and edit text files.

This section provides basic command and syntax information for the EDIT command. To learn about editing text with the EDT editor, refer to the *Introduction to the EDT Editor*, which is a tutorial manual that you can use at your terminal. For details about each command and feature of EDT, refer to the *EDT Editor Manual*.

You can also get information about EDT while using it. EDT's help facility provides a description about each of its editing commands. After you use the EDIT command to start EDT, type HELP after the asterisk prompt for help text. (In keypad editing, you press the HELP key for help text. Refer to the *EDT Editor Manual* for details.)

Format

EDIT file-spec

Command Qualifiers	**Defaults**
/COMMAND = file-spec	/COMMAND = EDTINI.EDT
/NOCOMMAND	
/EDT	/EDT
/FORMAT = STREAM	/FORMAT = STREAM
/FORMAT = VARIABLE	
/JOURNAL = [file-spec]	/JOURNAL
/NOJOURNAL	
/OUTPUT = file-spec	/OUTPUT
/NOOUTPUT	
/[NO]READ_ONLY	/NOREAD_ONLY
/[NO]RECOVER	/NORECOVER

Prompts

File: file-spec

Command Parameters

file-spec

Specifies the file to be created or edited using the EDT editor. If the file does not exist, it is created.

The only part of the file specification that you must supply is the file name. EDT does not provide a default file type when creating files; if you do not include a file type, no file type is assigned.

You cannot use wildcard characters in the file specification.

Command Qualifiers

/COMMAND = file-spec
/NOCOMMAND

Controls whether or not EDT reads a command file before prompting at the terminal. If you specify a command file, EDT executes all commands in the file before beginning the editing session. For example, if you have a file named CHANGE.INI containing line editing commands, you use it when editing file RUN.DAT by typing:

```
$ EDIT RUN.DAT/COMMAND=CHANGE.INI
```

By default, EDT looks for a command file named EDTINI.EDT when you begin an editing session. If the file EDTINI.EDT is in your directory on the public structure, EDT uses the commands it contains when the session begins. If you specify the /NOCOMMAND qualifier, EDT does not read a command file before beginning the editing session. To suppress the effects of the command file EDTINI.EDT when you start an editing session with file RUN.DAT, type:

```
$ EDIT RUN.DAT/NOCOMMAND
```

Otherwise, the commands in the EDTINI.EDT file automatically affect your editing session.

You cannot use wildcard characters in the command file specification.

/EDT

Ensures that the EDIT command starts EDT, if your system manager has changed the default on your system from EDT to another editor. Otherwise, you do not need to use this qualifier.

For example:

```
$ EDIT/EDT TEARS.JOY
```

The EDT editor is invoked, so you can produce a file named TEARS.JOY.

/FORMAT = STREAM
/FORMAT = VARIABLE

Determines the format of the output file.

/FORMAT = STREAM creates the file in stream ASCII format. The default is /FORMAT = STREAM. /FORMAT = VARIABLE creates the file in variable-length format. (Refer to the *RMS–11 User's Guide* for information about stream and variable file formats.)

/JOURNAL[=file-spec]
/NOJOURNAL

Controls whether or not a journal file is created for the editing session. A *journal file* contains a copy of the edits you make to another file, from the beginning to the end of the editing session. If your editing session ends abnormally (as in a system failure or "crash"), you can restart EDT (using the /RECOVER qualifier) and reinstate all commands from the aborted session.

When you begin an editing session, EDT automatically creates a journal file with the same name as the input file and a .JOU file type. If you are editing a file named RUN.DAT, for example, the journal file is named RUN.JOU.

The command line to state explicitly that you want a journal file when you edit RUN.DAT, and to name the journal file RUNDAT.V01 is:

```
$ EDIT RUN.DAT/JOURNAL=RUNDAT.V01RET
```

If you specify the /NOJOURNAL qualifier, no journal file is created.

If you omit the /JOURNAL qualifier or if you specify the qualifier without a file specification, the editor creates a journal file with the same file name as your input file and a default file type of .JOU. EDT also assumes a default file type of .JOU if you specify a file name but not a file type when you use the /JOURNAL qualifier.

The journal file is listed in your directory when the system resumes operation after a system crash. By default, EDT places the journal file in the same directory and device as the output file, if you specified /OUTPUT when creating the file. Otherwise, EDT uses the same directory and device as the input file.

For example, suppose you start a session to edit the existing file NANCY.DAT, but you want the finished output file to become file CLIFF.DAT:

```
$ EDIT NANCY.DAT/OUTPUT=CLIFF.DATRET
```

If the system crashes, the journal file is named CLIFF.JOU. You can then recover from a system crash by typing:

```
$ EDIT NANCY.DAT/OUTPUT=CLIFF.DAT/RECOVERRET
```

The /OUTPUT qualifier is required in file recovery if you used it to invoke EDT, because the journal file does not "remember" the command line you typed to begin the editing session.

You cannot use wildcard characters in the journal file specification.

You cannot use /NOJOURNAL if you specify /NOREAD_ONLY.

/OUTPUT = file-spec
/NOOUTPUT

Defines the file specification of the file created during the editing session. If you do not specify the /OUTPUT qualifier, the output file replaces the input file, and the previous version of the input file is saved as filename.BAK. Any previous file named filename.BAK is deleted.

For example, suppose you type:

```
$ EDIT SWEAT.SOX(RET)
```

If you continue the editing session and then exit from EDT by saving your edits, EDT automatically creates a new output file named SWEAT.SOX. The file you began the session with is renamed to SWEAT.BAK.

If you want the output file to be named something other than the original file name, you can specify the output file name explicitly. For example, the following command line tells EDT to put the contents of the file you are about to update into a file named SUPPLY.DAT

```
$ EDIT SWEAT.SOX/OUTPUT=SUPPLY.DAT(RET)
```

You cannot use wildcard characters in the file specification.

You can suppress the creation of the output file with the /NOOUTPUT qualifier:

```
$ EDIT SWEAT.SOX/NOOUTPUT(RET)
```

Use of the /NOOUTPUT qualifier does not suppress creation of the journal file.

You cannot use /OUTPUT if you specify /NOREAD_ONLY.

/READ_ONLY
/NOREAD_ONLY

Controls whether journaling and the creation of an output file are disabled when you view a file. The /READ_ONLY qualifier is equivalent to specifying /NOOUTPUT and /NOJOURNAL. You might use the /READ_ONLY qualifier to examine files without modifying them.

The default is /NOREAD_ONLY, which allows journaling and output.

/RECOVER
/NORECOVER

Determines whether or not EDT reads commands from a journal file before starting the editing session. The /RECOVER qualifier is useful if you are editing a file and the system ceases operation abruptly, as in the case of a power failure. The /RECOVER qualifier allows you to retrieve your work on the file. The default is /NORECOVER, which tells EDT not to recover the edits you made to the file.

For example, if you are editing file SWEAT.SOX when the system ceases operations, or "crashes," you can use the /RECOVER qualifier to restore your work when the system is back in operation:

```
$ EDIT SWEAT.SOX/RECOVER↵
```

If you named the journal file something other than SWEAT.JOU (for example, to SNEAK.ERS) when you began the previous editing session, the command line to use when restoring your work is:

```
$ EDIT SWEAT.SOX/RECOVER/JOURNAL=SNEAK.ERS↵
```

The /RECOVER qualifier reads EDT commands from the file specified by filename.JOU (which was created by EDT's /JOURNAL feature). This restores all commands that were lost in a previously aborted editing session.

The /RECOVER qualifier does not accept a file specification. Therefore, if the recovery file has a name different from filename.JOU, the journal file, you must specify both the /JOURNAL qualifier and the /RECOVER qualifier.

Displaying File Names and Files

The DIRECTORY command displays information about files. Use the TYPE command to display the contents of individual files.

DIRECTORY

The DIRECTORY command displays information about files.

Format

DIRECTORY [file-spec[,...]]

Command Qualifiers	Defaults
/BEFORE = date	
(N) /BRIEF	
/CREATED	
/CREATED	
/DATE[= CREATED]	/NODATE
[= MODIFIED]	
[= ALL]	
/NODATE	
(N) /FULL	
/MODIFIED	/CREATED
(N) /OUTPUT = file-spec	
/[NO]PROTECTION	/PROTECTION
/SINCE = date	
/SIZE[= ALLOCATION]	/SIZE = USED
[= USED]	
/NOSIZE	/SIZE
(N) /TOTAL	

If you type the DIRECTORY command and press the RETURN key, DCL lists the files in your directory on the public structure.

```
$ DIRECTORY(RET)

 Name .Typ       Size     Prot        Name .Typ      Size      Prot    SY:[52,20]
FINISH.COB        130    < 60>      TEN    .K         4     < 60>
START .BAS         89    < 60>      ROAD   .MAP       3     < 60>
TEMP16.TMP        258    < 60>

Total of 484 blocks in 5 files in SY:[52,20]
```

Nonprivileged users normally can get directory listings only of files to which they have access (read, write, or execute). If you specify a file or directory which is not yours, and to which you do not have access, one of the following messages is displayed:

```
?File does not exist
?Can't find file or account
```

However, the system manager can allow you to get directory listings of any file.

Command Parameter

file-spec[,...]

Specifies one or more files to be listed. If your installation has DECnet/E, you can also specify the node on which the files are located. The default is the host node.

The rules for using the file specification are:

- If you do not enter a file specification, the DIRECTORY command lists all the files in your directory on the system disk as shown in the previous example.
- If you specify only a device name, the DIRECTORY command lists the files in your directory on that device. For example, if you have files on a disk named DB1:, you can type:

  ```
  $ DIR DB1:RET
  ```

  To learn the names of the devices on your system, use the SHOW DEVICES command, which is described in Chapter 5.
- If you use wildcards for a file name or type, all files with that name or type are listed. For example, if your account contains files named RACE.1, RACE.2, and RACE.3, you can type:

  ```
  $ DIRECTORY RACE.*RET
  ```

  To get information about all the files in the directories you have access to, type:

  ```
  $ DIRECTORY [*,*]*.*RET
  ```

  This lists all files in all directories on the public disk structure. The first set of asterisks (in [*,*]) corresponds to every account number (as in [52,20]). The second set of asterisks (*.*) represents every file name and type.
- If a file specification contains a file name and a wildcard for the file type, the DIRECTORY command assumes all file types. Similarly, you can specify a file type and a wildcard for the file name. Therefore, both of the following are valid command lines:

  ```
  $ DIRECTORY RACE.*RET

  $ DIRECTORY *.1RET
  ```

If you provide more than one file specification, separate the file specifications with commas (,) or plus signs (+). You can use wildcard characters in the directory specification, file name, or file type of a file specification to list all files that satisfy the components you specify. For example:

```
$ DIRECTORY RACE.1,*.DAT,DB1:*.*RET
```

Command Qualifiers

/BEFORE=date

Specifies that only those files created or modified earlier than a given date be listed. The valid dates you can use are described in Chapter 2, "Entering Dates and Times."

You can use the /BEFORE qualifier in conjunction with either the /CREATED or /MODIFIED qualifier. For example:

```
$ DIRECTORY/CREATED/BEFORE=28-MAY-83RET

$ DIRECTORY/MODIFIED/BEFORE=TODAYRET
```

If you omit the /BEFORE qualifier, the files are displayed without regard to their dates.

You can use /BEFORE only when listing files on the host (your own) node.

/BRIEF

Lists only the file name and type of each file. If you specify /BRIEF, you do not see the number of blocks for the files or their protection codes. For example:

```
$ DIRECTORY/BRIEFRET

     Name .Typ   Name .Typ    Name .Typ     Name .Typ  Name .Typ  Name .Typ
     SY:[52,20]
     ROAD  .MAP TEMP25.TMP   STRIDE.TXT    PACE  .DAT

4 files 25 blocks
```

If you do not specify /BRIEF, the sizes and protection codes are also included in the display.

/CREATED

Selects the files according to their dates of creation. Use the /CREATED qualifier only with the /BEFORE or /SINCE qualifiers. For example:

```
$ DIRECTORY/BEFORE=10-JUL-83/CREATEDRET

  Name .Typ        Size     Prot     Name .Typ   Size    Prot     SY:[52,20]
  ROAD  .MAP          4    <  60>    STRIDE.TXT   21    <  40>
  PACE  .DAT          8    <  60>

Total of 33 blocks in 3 files in SY:[52,20]
```

Do not use /CREATED with the /MODIFIED qualifier. (You can select files either according to their date of creation or date of modification, but not both.) After a file is created, any subsequent updates cause the file to be modified.

By default, the selection of files according to a date uses the creation date.

You can use /CREATED only when listing files on the host (your own) node.

/DATE=CREATED
/DATE=MODIFIED
/DATE[=ALL]

The /DATE qualifier causes the DIRECTORY listing to include the date when a file was created or last accessed, or both. If you specify /DATE without an argument, /DATE=ALL is assumed.

/DATE=CREATED requests information about the file's creation date and time:

```
$ DIRECTORY/DATE=CREATED

  Name .Typ  Size    Prot     Date        Time          SY:[52,20]
  ROAD  .MAP  4     < 60>   14-MAY-83   10:13 AM

Total of 4 blocks in 1 file in SY:[52,20]
```

/DATE=MODIFIED displays the date when the file was last updated or accessed:

```
$ DIRECTORY/DATE=MODIFIED

Name .Typ   Size   Prot     Access     Name .Typ   Size    Prot     Access   SY:[52,50]
ROAD  .MAP   4    < 60>   08-JUN-83  STRIDE.TXT    21    < 40>   14-MAY-83

Total of 25 blocks in 2 files in SY:[52,20]
```

/DATE=ALL provides date and time information about the creation and access of a file:

```
$ DIRECTORY/DATE=ALL

  Name .Typ   Size     Prot        Access       Date         Time
  SY:[52,20]
  ROAD  .MAP    4     < 60>      08-JUN-83   08-JUN-83    04:25 PM

Total of 4 blocks in 1 file in SY:[52,20]
```

/NODATE

Lists information about files but does not include dates. Unless you specify /DATE, the /NODATE qualifier is assumed. For example:

```
$ DIRECTORY/NODATE

  Name .Typ   Size  Prot   Name .Typ   Size     Prot    SY:[52,20]
  ROAD  .MAP    4  < 60>  STRIDE.TXT    21     < 40>

Total of 25 blocks in 2 files in SY:[52,20]
```

You can use /DATE and /NODATE only when listing files on the host (your own) node.

/FULL

Lists the following items for each file:

- Name
- Type
- Number of blocks used
- Protection
- Date last accessed or modified
- Creation date and time
- Cluster size
- Run-time system name
- Position of first block in file
- File attributes, if any

A file's run-time system name pertains to executable files, because it is the run-time system that a program runs under. For information about any of the above items, refer to the *RSTS/E System User's Guide.*

The /FULL qualifier implies the qualifiers /DATE=ALL and /PROTECTION as defaults. /FULL also includes information about the file's cluster size (in the Clu column) and position on the disk (in the Pos column). Often the /FULL qualifier is used with a file specification:

```
$ DIRECTORY/FULL RACE.*RET

  Name.Typ      Size      Prot          Access      Date       Time     Clu    RTS  Pos
  SY:[52,20]
  RACE  .1        4      < 60>       12-SEP-83  12-SEP-83  01:02 PM    4 ...RSX 22699
  RACE  .2        3      < 60>       25-SEP-83  25-SEP-83  09:55 AM    4 ...RSX 23284

Total of 7 blocks in 2 files on SY:[52,20]
```

/MODIFIED

Selects files according to the last date the file was modified. Use this qualifier only with the /BEFORE or /SINCE qualifiers. For example:

```
$ DIRECTORY/MODIFIED/BEFORE=12-JUN-83RET

  Name .Typ  Size    Prot      Name .Typ Size    Prot    SY:[52,20]
  ROAD  .MAP   4    < 60>     STRIDE.TXT   21   < 40>

Total of 25 blocks in 2 files in SY:[52,20]
```

```
$ DIRECTORY/MODIFIED/SINCE=18-MAY-83

  Name .Typ  Size    Prot      Name .Typ Size    Prot    SY:[52,20]
  ROAD  .MAP   4    <  60>    STRIDE.TXT   21   <  40>
  RACE  .1     3    <  60>

Total of 28 blocks in 3 files in SY:[52,20]
```

Note that only the file name and type, file size, and protection code are displayed. By contrast, /DATE = MODIFIED displays the dates of file access.

Do not use /MODIFIED with the /CREATED qualifier. (The date you specify can apply only to the creation or modified date; hence you cannot specify both /CREATED and /MODIFIED.) After a file is created, any subsequent updates cause the file to be modified.

You can use /MODIFIED only when listing files on the host (your own) node.

/OUTPUT = file-spec

Creates a file containing the output of a DIRECTORY listing, instead of displaying the output at your terminal. You must include a file name with the /OUTPUT qualifier. If you omit the file type in the file specification, the default type is DIR.

For example, the following command line puts the directory listing into a file named RACES.DIR:

```
$ DIRECTORY/OUTPUT=RACES[RET]

$ TYPE RACES.DIR[RET]
  Name .Typ  Size    Prot      Name .Typ Size    Prot    SY:[52,20]
  ROAD  .MAP   4    <  60>    STRIDE.TXT   21   <  40>
  RACE  .1     3    <  60>

Total of 28 blocks in 3 files in SY:[52,20]

$ DIRECTORY RACES[RET]

  Name .Typ  Size    Prot      Name .Typ Size    Prot    SY:[52,20]
  RACES.DIR    2    <  60>

Total of 2 blocks in 1 file in SY:[52,20]
```

Note that the output file has the type .DIR.

No wildcard characters are allowed in the output file specification.

/PROTECTION
/NOPROTECTION

Specifies the protection code of files to be in the directory listing.

You can use /[NO]PROTECTION only when listing files on the host (your own) node.

/SINCE=date

Specifies that only those files created or modified after a specified date be displayed.

You normally use the /SINCE qualifier with either the /CREATED or /MODIFIED qualifier. If you omit the /SINCE qualifier, you obtain all the files, regardless of creation date. The date can be either TODAY or YESTERDAY, or a date in the format dd-mmm-yy (as in 12–MAY–83 or 83.5.12). For example:

```
$ DIRECTORY/CREATED/SINCE=06-APR-83RET

  Name .Typ  Size    Prot      Name .Typ Size    Prot    SY:[52,20]
  ROAD .MAP   4    < 60>     STRIDE.TXT   21   < 40>
  RACE .1     3    < 60>

Total of 28 blocks in 3 files in SY:[52,20]
```

You can use /SINCE only when listing files on the host (your own) node.

/SIZE=ALLOCATION

/SIZE=USED

The /SIZE qualifier lists the number of blocks a file either uses or is allocated. Compare the following two examples:

```
$ DIR *.LOG/SIZE=USEDRET

  Name .Typ  Size    Prot      Name .Typ  Size    Prot    SY:[52,20]
  BAR  .LOG   3    < 60>     FOO  .LOG   6    < 40>
  LINK .LOG  12    < 60>     HELP .LOG   6    < 60>

Total of 27 blocks in 4 files in SY:[52,20]

$ DIR *.LOG/SIZE=ALLOCATIONRET

  Name .Typ  Size    Prot      Name .Typ  Size    Prot    SY:[52,20]
  BAR  .LOG   4    < 60>     FOO  .LOG   8    < 40>
  LINK .LOG  12    < 60>     HELP .LOG   8    < 60>

Total of 32 blocks allocated in 4 files in SY:[52,20]

$
```

The first example shows /SIZE=USED, which is the default. The Size column shows the number of blocks used by each file. By contrast, the Size column in the second example shows the number of blocks allocated to store each file. This may be larger because disk space is allocated in multiples of the file's cluster size.

You can use /SIZE only when listing files on the host (your own) node.

/NOSIZE

The /NOSIZE qualifier suppresses information on file size. /SIZE is the default.

/TOTAL

The /TOTAL qualifier displays the total amount of space your files require, without listing information about each file:

```
$ DIRECTORY/TOTAL[RET]

SY:[52,20]
Total of 647 blocks in 24 files in SY:[52,20]
```

You can use /TOTAL only when listing files on the host (your own) node.

TYPE

The TYPE command displays the contents of a text file (as opposed to a binary or temporary file) at the terminal.

Format

TYPE file-spec[,...]

Command Qualifiers	Defaults
(N) /OUTPUT = file-spec	/OUTPUT = KB:
(N) /[NO]QUERY	/NOQUERY

Prompts

File: file-spec[,...]

For example, you can display the contents of the file TRAIN.TXT:

```
$ TYPE TRAIN.TXT(RET)

Even though it was difficult to
undertake a new running regime,
I was amazed at how soon I was
able to increase my daily mileage.

$
```

You can control the display of a file in any of the following ways:

- To temporarily halt the output, use CTRL/S. To resume output where it was interrupted, use CTRL/Q. This works only if your terminal has the TTSync characteristic set. (See the section in Chapter 4 named "Setting Your Terminal's characteristics.") On VT100 terminals you can also press the NO SCROLL key to stop and restart output.
- To suppress the display but continue command processing, use CTRL/O. If you press CTRL/O again before processing is completed, output resumes at the current point in command processing.
- To stop command execution entirely, press CTRL/C. The use of CTRL/C returns you to DCL command level.

Command Parameters

file-spec[,...]

Specifies one or more files to be displayed. If your installation has DECnet/E, you can specify the node where the files are located. You can also use wildcards.

If you specify a file name and do not specify a file type, the TYPE command defaults to the null file type.

If you want to display more than one file, separate the file specifications with commas (,) or plus signs (+). In either case, the files are displayed in the order you specify. The TYPE command allows wildcards in place of the directory, file name, and file type fields. The following command line contains wildcards and commas:

```
$ TYPE *.TXT,ROAD.MAP,RACE.*RET
```

The contents of all files you specify with the type .TXT, the file ROAD.MAP, and all files named RACE are displayed.

Command Qualifiers

/OUTPUT = file-spec

Creates a file from the output of the TYPE command, rather than displaying the output at the terminal. For example, you might use /OUTPUT to merge several files into one:

```
$ TYPE *.TXT+ROAD.MAP,RACE.*/OUTPUT=MERGER.TXTRET
```

The contents of all files with the file type .TXT, the file ROAD.MAP, and all files named RACE are copied in the order specified into one file named MERGER.TXT. Unlike the COPY command, the /OUTPUT qualifier converts data to stream ASCII, and lets you combine text files that have attributes. (Refer to the *BASIC–PLUS Language Manual* for more information about stream ASCII data and the *RSTS/E System User's Guide* for more information about file attributes.)

/QUERY
/NOQUERY

When you use the /QUERY qualifier, the system displays the name of each file you request and asks whether or not you want to see the file. (The /QUERY qualifier is useful only when you use wildcard file specifications in the TYPE command line.)

For example, if you have eight files in your directory with the file type .TXT, you can enter:

```
$ TYPE *.TXT/QUERYRET
```

DCL prints a file specification followed by a question mark (?) prompt. You can reply by typing either:

Y	Yes, display the file.
N	No, do not display the file.
CTRL/Z	Quit; return to command level.
RET	No, do not display the file.

```
[52,20]TRAIN .TXT? N(RET)
[52,20]STRIDE.TXT? Y(RET)

Hill training taught me that
short, quick steps are good for
running uphill. Longer strides
are best for downhill inclines
and straightaways.

[52,20]PACE  .TXT?(CTRL/Z)

$
```

Deleting Files: DELETE

The DELETE command permanently removes a file from a directory.

Format

DELETE file-spec[,...]

Command Qualifiers	Defaults
/BEFORE = date	
/CREATED	/CREATED
/ERASE	
(N) /[NO]LOG	/LOG
/MODIFIED	
(N) /[NO]QUERY	/NOQUERY
/SINCE = date	

Prompts

Files: file-spec[,...]

For example, if you want to delete the file CLIFF.DAT, type:

```
$ DELETE CLIFF.DAT(RET)
```

Command Parameter

file-spec[,...]

Specifies the name of one or more files to be deleted. If your installation has DECnet/E, you can specify the node where the files are located. See Chapter 1, Network File Specifications, for more information.

Each file specification must contain a file name and a file type. You can specify wildcard characters for the directory, file name, or file type:

```
$ DELETE DB2:RACE.*(RET)
```

If you do not include a file type with DELETE, the default is a null file type.

To delete more than one file, separate the file specifications with commas (,) or plus signs (+). If you omit the account specification or device name, your directory and the system disk are used:

```
$ DELETE ROAD.MAP,*.TXT+STRIDE.*(RET)
```

If you give a node name, files on that node are deleted. For example:

```
$ DELETE(RET)
Files: SPEEDY"GONZO MK1"::HIWAY.RUN(RET)
```

Command Qualifiers

/BEFORE = date

Specifies that only the files dated earlier than a particular date be deleted. Use the syntax rules for date values. The /BEFORE qualifier is useful only when you include wildcard characters in the file specification.

Use the /CREATED or /MODIFIED qualifiers in conjunction with /BEFORE to request that only files created or modified before the specified date be deleted. If neither of these qualifiers is specified, /CREATED is assumed.

For example, suppose you have files named RESULT.A, RESULT.B, and RESULT.C in your directory, and you created file RESULT.C today. To delete the files created before today, type:

```
$ DELETE/BEFORE=TODAY RESULT.*RET
RESULT.A deleted
RESULT.B deleted
```

You can use /BEFORE only when deleting files on the host (your own) node.

/CREATED

Specifies that only files created within a defined time period be deleted. Use the /CREATED qualifier with either the /BEFORE or /SINCE qualifiers.

For example, if you do not want to save any of the files you created today, type:

```
$ DELETE/CREATED/SINCE=TODAYRET
```

If you do not specify /MODIFY or /CREATED, the default is /CREATED.

You can use /CREATED only when deleting files on the host (your own) node.

/ERASE

Specifies that you want to zero the file prior to deleting it. This qualifier is useful if you want to erase a file for security purposes.

For example, to erase the file JIM.LOG, type:

```
$ DELETE/ERASE JIM.LOGRET
JIM.LOG erased and deleted
```

/LOG
/NOLOG

Controls whether the DELETE command displays the file specification of each file after its deletion.

The default is /LOG; normally the DELETE command displays the names of files after it deletes them. For example:

```
$ DELETE RISTUS.ONE[RET]
RISTUS.ONE deleted

$
```

You can request the deletion message explicitly with /LOG:

```
$ DELETE/LOG PETER.WRK[RET]
PETER.WRK deleted

$
```

Note that the /NOLOG qualifier suppresses the usual message:

```
$ DELETE/NOLOG ROAD.MAP[RET]

$
```

/MODIFIED

Specifies that only files modified within a defined time period be deleted. Use the /MODIFIED qualifier with either the /BEFORE or /SINCE qualifiers. If you do not specify /MODIFIED, the default is /CREATED.

For example, if you want to delete all files that have not been modified since June 11, type:

```
$ DELETE/MODIFIED/BEFORE=11-JUN[RET]
```

A file's revision date is updated whenever the file is modified. (Your system manager can change the default, so that the file is updated whenever the file is read or modified.)

You can use /MODIFIED only when deleting files on the host (your own) node.

/QUERY

Specifies that you want to select files to be deleted. The /QUERY qualifier is recommended when you use a wildcard in the command line, so you can avoid deleting files unintentionally. /NOQUERY is the default.

For example, suppose you want to delete some of your files with a .DAT file type. The command line is:

```
$ DELETE *.DAT/QUERY[RET]
```

DCL prints a file specification followed by a question mark (?) prompt. You can reply by typing either:

Y	Yes, delete the file.
N	No, do not delete the file.
CTRL/Z	Quit; return to command level.
RET	No, do not delete the file.

The display at your terminal looks similar to:

```
[52,20]WRITE .DAT? N
[52,20]MANUAL.DAT? Y
MANUAL.DAT deleted
[52,20]PACE  .TXT? CTRL/Z

$
```

/SINCE = date

Specifies that only the files dated on or after a particular date be deleted. Use the rules for date values described in Chapter 2, "Specifying Dates and Times."

A file is implicitly modified on the date it was created. Therefore, a file's date of modification never exceeds its date of creation.

Use the /CREATED or /MODIFIED qualifier to request that only files created or modified on or after the specified date be deleted. If you do not specify /CREATED or /MODIFIED, the DELETE command deletes all files created since the specified date.

For example, the following command line deletes all files created on or after June 11:

```
$ DELETE/SINCE=11-JUN *.*RET
```

To delete all files named RACE that were modified since April 6, type:

```
$ DELETE/MODIFIED/SINCE=06-APR RACE.*RET
```

You can use /SINCE only when deleting files on the host (your own) node.

Copying, Renaming, and Appending Files

The commands described in this section let you copy, rename, and append the contents of files.

COPY

The COPY command duplicates one or more existing files, or concatenates two or more files. You can use this command for local and network file specifications. See Chapter 1 (Network File Specifications) for more information.

Format

COPY input-file-spec[,...] output-file-spec

Qualifiers	Defaults
/ALLOCATION=n	
/BLOCK_SIZE=n	/BLOCK_SIZE=512
/CLUSTER_SIZE=n	
(N) /[NO]CONTIGUOUS	
(N) /[NO]LOG	/LOG
/[NO]OVERLAY	/NOOVERLAY
/PROTECTION=n	
(N) /[NO]QUERY	/NOQUERY
(N) /[NO]REPLACE	

Prompts

From: input-file-spec[,...]
To: output-file-spec
OK to replace existing file file-spec?

You can use COPY to:

- Copy one file to another file
- Merge (*concatenate*) more than one file into a single file
- Copy a group of files to another group of files

For example, if you have a file named SMITH.MAR and you want to make a copy of it as file JONES.JOE, you type:

```
$ COPY SMITH.MAR JONES.JOE↵
```

The following command copies all of your files on the public structure (SY:) to a tape (MT0:). (If the tape is in DOS format, the files are copied to your PPN on the tape.)

```
$ COPY SY:*.* MT0:RET
```

To copy every file on a tape (MT0:) into your directory on the public structure (SY:), type:

```
$ COPY MT0:[*,*]*.* *.*RET
```

If you copy files from ANSI tape to disk, you do not need to include the [*,*] directory specification. (ANSI format tapes do not have directories.)

When you specify more than one input file and a single output file, the COPY command merges all the input files into the output file:

```
$ COPY RABBIT.1,RABBIT.2,RABBIT.3 BUNNY.ALLRET
```

The example merges a copy of files RABBIT.1, RABBIT.2, and RABBIT.3 into one file named BUNNY.ALL.

If you specify more than one input file and give an output file specification that contains a wildcard character, the COPY command creates one output file for each input file:

```
$ COPY HARE.1,HARE.2,HARE.3 BIGFUT.*RET
```

The example copies files HARE.1, HARE.2, and HARE.3 into files named BIGFUT.1, BIGFUT.2, and BIGFUT.3, respectively.

Similarly, the following example copies all files named ONE into files named TWO, with the same file types:

```
$ COPY ONE.* TWO.*RET
```

You can specify multiple input files in any of the following ways:

- Separate input file specifications with commas (,) or plus signs (+). For example:

  ```
  $ COPY ROAD.LST,RACE.DAT,ENTRY.LST RESULT.MRGRET
  ```

  The input files ROAD.LST, RACE.DAT, and ENTRY.LST are copied in that order into an output file named RESULT.MRG.

- Specify wildcard characters in place of the directory specification, file name, or file type of an input file specification. The following example copies all COBOL source files on D:[160.*] to the public structure:

```
$ COPY D:[160,*]*.CBL SY:*.*(RET)
```

The expression D:[160,*] means all directories on disk D: with project number 160. (For example, directories D:[160,25], D:[160,5], and D:[160,39]). All COBOL source files (*.CBL) on D:[160,*] are copied to the public structure (SY:) into files of the same name and type. You must have access to each of these files to use this command string.

The COPY command creates multiple output files when you specify multiple input files and one of the following:

- Asterisk wildcard characters (*) in the output directory specification, file name, or file type fields:

```
$ COPY RACE1.*,ROAD.MAP DB1:*.*(RET)
```

- You omit the output file name, or you omit the file type and the period preceding it (either of which is equivalent to specifying a wildcard).

When you specify a wildcard for any part of the output file specification, the COPY command uses the corresponding part of the input file specification. When you omit the output file name, or when you omit the output file type and the preceding period, either of these is equivalent to specifying a wildcard.

Command Parameters

input-file-spec[,...]

Specifies one or more input files to be copied. If you specify more than one input file, separate them with either commas (,) or plus signs (+). You can use wildcard characters for the directory, file name, or file type.

The following example uses wildcards and commas:

```
$ COPY *.TXT,RACE.1,RACE.2 FINISH.TXT(RET)
```

This command line copies all files with the type .TXT, and files RACE.1 and RACE.2 into a file named FINISH.TXT.

output-file-spec

Specifies the name of the file into which the input files are to be copied.

You must specify at least one part of the output file specification.

If you do not specify a device or directory, the COPY command uses the public structure and your own directory. For other parts that you do not specify, the COPY command uses the corresponding field of the input file specification.

If you specify an asterisk wildcard character (*) in place of the file name or file type of the output file specification, the COPY command creates one or more output files, based on the input file specification. It uses the corresponding part of the input file specification to name the output file.

For example, assume you have files NAME.1, NAME.2, and NAME.3 in your directory. To copy these files into duplicates named NEW.1, NEW.2, and NEW.3, you type:

```
$ COPY NAME.* NEW.*
```

You can use wildcard characters in the directory specification of an output file. For example, the following command copies all files in all directories on the public structure onto DB1:, into the corresponding directory, file names, and file types:

```
$ COPY [*,*]*.* DB1:[*,*]
```

The preceding command line copies all files in all directories onto a disk named DB1:. The file names remain the same because you do not specify an output file name.

The ability to copy files depends on the file protection codes and whether or not you have write access to the directory to which you are copying them.

Qualifiers

/ALLOCATION=n

Creates space on the disk by forcing the initial allocation of the output file to a number of 512–byte blocks, which you specify with n. The following example provides three blocks of space to copy the file SNEAK.ERS into SHOE.STR:

```
$ COPY SNEAK.ERS SHOE.STR/ALLOCATION=3
```

If you do not specify the initial allocation of the output file, then RSTS/E determines it by the size of the input file being copied.

You can use /ALLOCATION only if the input and output files are on the host (your own) node.

/BLOCK_SIZE=n

Allows you to specify a block size for output to magnetic tape.

For example:

```
$ COPY MYFILE.TXT MMO:MYFILE.TXT/BLOCK_SIZE=2048
```

The argument of the /BLOCK_SIZE=n qualifier must be an even integer in the range of 18 to 4096 bytes. The default is 512 bytes.

If you are copying a large file, a larger block size can be useful because it reduces the number of interblock gaps that are needed, and can therefore allow you to fit more data onto a tape. On the other hand, a small block size is often preferable for a small file, since it reduces the amount of space wasted at the end of the file.

If you specify an invalid value for n, the COPY command prints the "?Bad block size" error message.

The /BLOCK_SIZE=n qualifier is subject to the following restrictions:

1. The qualifier can be used only for magnetic tape. Its use on disk is ignored.
2. The qualifier applies only to output files. The COPY command automatically handles input magnetic tape files with block sizes other than 512 bytes.
3. If the output tape is in ANSI format and is intended for interchange with another operating system, the block size must be an even integer between 18 and 2048 bytes. Tapes having block sizes greater than 2048 bytes cannot be read by other operating systems.
4. If the output tape is intended for use on the RT–11 operating system, the block size must be 512 bytes.
5. If both the input and output files are on magnetic tape, and both have large block sizes, the COPY command may fail because of insufficient buffer space. If this happens, transfer the file to disk, and then use the COPY command to transfer the file from disk to tape.

/CLUSTER_SIZE=n

Establishes the cluster size for a disk file. A cluster size is a number of contiguous blocks taken together as a unit. The cluster size is the minimum unit of allocation for the file.

This qualifier is especially useful for large files because specifying a large cluster size speeds random access to the data.

In addition, you can avoid filling up your directory by specifying a large cluster size. (In your directory, the system maintains an internal list of the clusters in each file. If your directory fills up, you will receive the message "?No room for user on device" even though the disk is not full.)

RSTS/E allows cluster sizes of 1, 2, 4, 8, 16, 32, 64, 128, and 256 blocks. With the SHOW DEVICES command, you can display the minimum cluster size of each disk. If you specify a cluster size below the minimum, the system will use the disk's minimum and will not display an error message. Note that the minimum set for each disk might vary.

/CONTIGUOUS
/NOCONTIGUOUS

Indicates whether the output file is to be contiguous. (A *contiguous* file occupies consecutive physical disk blocks).

For example, the following command line causes the file MANUAL.RUN to be stored on consecutive blocks on disk DB2:.

```
$ COPY MANUAL.RUN DB2:*.*/CONTIGUOUS(RET)
```

By default, the COPY command creates an output file in the same format as the corresponding input file. If an input file is contiguous, the COPY command attempts to create a contiguous output file. If there is not enough contiguous disk space, the system copies the file into available empty disk blocks and displays the message:

```
%File file-spec created noncontiguous
```

The /CONTIGUOUS qualifier is ignored if the output file is not on disk.

If you are concatenating several input files, or if the single input file is on a device other than disk, and you are creating a contiguous output file, then you must use the /ALLOCATION qualifier to preallocate the size of the output file. This is necessary because the system cannot precompute the size of the output file in these cases.

If you do not specify an allocation, or if the allocation is too small, the following error message is displayed when the system tries to increase the size of the contiguous output file:

```
?Protection violation
```

If you copy a file from a tape and want the file to be contiguous, you can use two COPY commands: one to copy the file from the tape, and another to create a contiguous file. For example, assume you want to copy file FRIDAY.WOW from tape MT0: to a contiguous file in your account ([52,20]) on the public structure. You rename the file to WORK.END temporarily, because you cannot copy file FRIDAY.WOW to itself when making it contiguous:

```
$ COPY MT0:FRIDAY.WOW SY:[52,20]WORK.END/LOG(RET)
[File FRIDAY.WOW copied to SY:[52,20]WORK.END]

$ COPY WORK.END FRIDAY.WOW/CONTIGUOUS/LOG(RET)
[File SY:[52,20]WORK.END copied to SY:[52,20]FRIDAY.WOW]

$ DELETE WORK.END(RET)
```

The first command copies file FRIDAY.WOW from tape MT0: into a file named WORK.END in your account ([52,20] on the public structure.) The second command copies WORK.END to a contiguous file named FRIDAY.WOW. The third command deletes the file WORK.END, because you used it only to move file FRIDAY.WOW into your account as a contiguous file. (Note that the /LOG qualifier displays each completed file operation.)

If you specify /CONTIGUOUS and there is not enough contiguous disk space, the system makes the output file noncontiguous and prints the warning message:

```
%File file-spec created noncontiguous
```

If you specified a network node name on either the input or output file, the COPY command fails, and the following message is displayed:

```
?Device full: Can't create or extend file
```

/LOG
/NOLOG

Controls whether the COPY command displays the file specifications of each file copied. /LOG is the default.

Unless you specify /NOLOG, the COPY command displays the file specifications of the input and output files. For example:

```
$ COPY DR3:[52,20]CREATE.BAS [1,248]*.*(RET)
[File DR3:[52,20]CREATE.BAS copied to [1,248]CREATE.BAS]
```

/OVERLAY
/NOOVERLAY

Requests that data in the input file be copied into an existing output file, replacing the output file's existing data. The physical location of the file on disk does not change.

For example, assume you rewrote the first chapter (CHAP1.TXT) of a manual (MANUAL.RUN). To overlay the existing chapter with the new first chapter, type:

```
$ COPY CHAP1.TXT MANUAL.RUN/OVERLAY(RET)
```

If the input file is smaller than the output file, the excess data in the output file is not removed. In such a case, the COPY command prints a message similar to the following (unless you use the /NOLOG qualifier):

```
[File Y.CMD copied into prefix of [1,248] T.CMD]
```

The default is /NOOVERLAY. In addition, RSTS/E ignores the /OVERLAY qualifier if the output file is written to any device other than a disk.

You can use /[NO]OVERLAY only when the input and output files are on the host (your own) node.

/PROTECTION=n

Specifies the protection code of the output file.

/PROTECTION cannot be used with /OVERLAY, and cannot be used in network operations. (The protection code 128 is ignored unless you are a privileged user. Refer to Table 3–3 for more information.)

By default, RSTS/E assigns files a protection code of 60, unless your system manager changes the default. Furthermore, if you use the SET PROTECTION /DEFAULT command before the COPY operation, the protection code becomes the value you specified as the default. (The SET PROTECTION command is described at the end of this chapter.)

You can use /PROTECTION only if the input and output files are on the host (your own) node.

/QUERY
/NOQUERY

Lets you select files to be copied when you specify more than one file. The /QUERY qualifier is useful only if you use wildcards. /NOQUERY is the default.

When you use /QUERY, DCL displays each input file name and a question mark prompt. You can enter any one of the following after each prompt:

Y	Yes, copy the file.
N	No, do not copy the file.
CTRL/Z	Skip remaining files, return to command level.
RET	No, do not copy the file.

For example, if you have files STRIDE.TXT, RACE.TXT, and FINISH.TXT in your account, you can use the /QUERY qualifier as follows:

```
$ COPY *.TXT MANUAL.RUN/NOOVERLAY/QUERY
STRIDE.TXT? Y
File STRIDE.TXT copied to MANUAL.RUN
RACE  .TXT? N
FINISH.TXT? CTRL/Z
```

/REPLACE
/NOREPLACE

/REPLACE requests that if a file already exists with the same file specification as the one you are creating, the existing file is to be deleted.

For example, if there is a file named ROAD.MAP in your default directory, and you want the file you are copying to replace the existing file, type:

```
$ COPY ROAD.MAP/REPLACE(RET)
```

If the COPY command would replace an existing file, but you do not specify /REPLACE or /NOREPLACE, DCL displays the prompt:

```
OK to replace existing file-spec?
```

You can reply with one of the following:

Y	– Yes, replace the file
N	– No, do not replace the file
(RET)	– No, do not replace the file

A yes response allows you to copy the file and replace the existing file. Otherwise, you are returned to command level.

For example, assume you inadvertently enter the name of an existing file that you do not want replaced. The display at the terminal reads:

```
$ COPY SORE.FUT(RET)
OK to replace existing file DR1:[52,20]SORE.FUT? N(RET)
```

/NOREPLACE forces the COPY command to fail if the file being created already exists. In addition, it suppresses the "OK to replace existing file" prompt.

If you use the COPY command in a batch control file, you should explicitly specify either /REPLACE or /NOREPLACE. This prevents DCL from issuing the "OK to replace existing file" prompt.

RENAME

The RENAME command changes the file name or type of an existing file.

Format

RENAME old-file-spec[,...] new-file-spec

Qualifiers	**Defaults**
/[NO]LOG	/LOG
/PROTECTION=n	
/[NO]QUERY	/NOQUERY
/[NO]REPLACE	/NOREPLACE

Prompts

From: old-file-spec[,...]
To: new-file-spec

For example, you can rename the file CLIFF.DAT to KAHN.DAT with the command line:

```
$ RENAME CLIFF.DAT KAHN.DAT
```

Command Parameters

old-file-spec[,...]

Specifies one or more files whose specifications you want to change. (The files must be stored on a disk.)

You can use wildcard characters in the directory, file name, or file type of the file specification. If you use wildcards, all the files you specify are renamed (assuming that they are your files or that you are a privileged user).

new-file-spec

Provides the new file name or type to be applied to the input file. The RENAME command uses the file name and type of the input file specification to provide defaults for fields you do not specify in the output file. (You cannot change the device or directory of the files. Attempts to do so will be ignored.)

You can specify an asterisk (*) in place of the file name or type of the output file specification, or you can omit either one; the RENAME command uses the corresponding field in the input file specification to name the output file. The /QUERY qualifier is useful when you use RENAME with wildcards.

Qualifiers

/LOG
/NOLOG

When you rename a file, DCL automatically displays a message to confirm that the file was renamed. You can also request this message explicitly with the /LOG qualifier, although /LOG is the default. /NOLOG supresses the message. For example:

```
$ RENAME RACE1.TXT FUN1.RUN(RET)
RACE1.TXT renamed to FUN1.RUN

$ RENAME RACE2.TXT FUN2.RUN/LOG(RET)
RACE2.TXT renamed to FUN2.RUN

$ RENAME RACE3.TXT FUN3.RUN/NOLOG(RET)

$
```

/PROTECTION=n

Specifies the protection code to assign to the file to be renamed. By default, the protection codes of the files are not changed.

/QUERY
/NOQUERY

Allows you to select which files you want to rename. The /QUERY qualifier is only useful when you use wildcards in the old file specification; /NOQUERY is the default.

When you use /QUERY, DCL displays individual file names and a question mark prompt. You can reply with:

Y	Yes, rename the file.
N	No, do not rename the file.
CTRL/Z	Skip remaining files, return to command level.
RET	No, do not rename the file.

If the file exists and you do not specify /REPLACE, DCL displays the message:

```
?Name or account now exists - file file-spec - continuing
```

For example, suppose you want to rename some of your files with a .TXT (text) file type to files with a .DAT (data) file type. You can enter the following command line and then answer the prompts as shown:

```
$ RENAME *.TXT *.DAT/QUERY(RET)
[52,20]RACE1.TXT? Y(RET)
RACE1.TXT renamed to RACE1.DAT
[52,20]STRIDE.TXT? Y(RET)
?Name or account now exists - file STRIDE.DAT - continuing
[52,20]ATLAS.TXT? N(RET)
[52,20]TEXT.TXT? CTRL/Z

$
```

/REPLACE
/NOREPLACE

Replaces any existing files with a file to which you assign the same name. /NOREPLACE is the default. If you do not specify /REPLACE or /NOREPLACE, the COPY command queries you before replacing a file. The /NOREPLACE qualifier (whether by default or by inclusion) is useful when you know that you need all your files and do not want to risk replacing one.

For example, assume you want to rename the file PAVED.DAT to TRACKS.DAT, and that you no longer need the existing file named TRACKS.DAT. The first of the following command lines assumes the /NOREPLACE qualifier, the second explicitly includes /NOREPLACE, and the third says to replace the file:

```
$ RENAME PAVED.DAT TRACKS.DAT
?Name or account now exists - file TRACKS.DAT - continuing

$ RENAME PAVED.DAT TRACKS.DAT/NOREPLACE
?Name or account now exists - file TRACKS.DAT - continuing

$ RENAME PAVED.DAT TRACKS.DAT/REPLACE
File PAVED.DAT renamed to TRACKS.DAT

$
```

APPEND

The APPEND command adds the contents of one or more files to the end of the file you specify. APPEND is similar in syntax and function to the COPY command.

Format

APPEND input-file-spec[,...] output-file-spec

Command Qualifiers	Defaults
(N) /[NO]LOG	/LOG
(N) /[NO]QUERY	/NOQUERY

Prompts

From: input-file-spec[,...]
To: output-file-spec

For example, assume you create two files and then decide to append the first file to the second one:

```
$ CREATE FILE.A(RET)
AAAAAAAAAAAAAAAAAA(RET)
(CTRL/Z)

$ CREATE FILE.B(RET)
BBBBBBBBBBBBBBBBBB(RET)
(CTRL/Z)

$
```

The APPEND command puts the contents of FILE.A at the end of FILE.B:

```
$ APPEND FILE.A FILE.B(RET)
[File FILE  .A appended to [52,20]FILE.B  ]

$ TYPE FILE.B(RET)
BBBBBBBBBBBBBBBBBB
AAAAAAAAAAAAAAAAAA

$
```

FILE.B now includes a copy of FILE.A; the original FILE.A remains unchanged.

Note that you cannot append a file to itself. If you try, an error message is displayed:

```
$ APPEND MOXIE.DAT MOXIE.DAT(RET)
?Protection violation - file MOXIE .DAT

$
```

Command Parameters

input-file-spec[,...]

Specifies one or more input files to be appended to the output file. If your installation has DECnet/E, you can specify the node where the input files are located.

If you specify more than one input file, separate the specifications with commas (,) or plus signs (+). All files are appended, in the order specified, to the end of the output file.

You can use wildcards in any part of the file specification. For example, to combine files STRIDE.REP, STRIDE.COP, and STRIDE.TXT into an existing file named EVENT.DOC, you type:

```
$ APPEND STRIDE.*(RET)
To:      EVENT.DOC(RET)
[File STRIDE.REP appended to [52,20]EVENT.DOC]
[File STRIDE.COP appended to [52,20]EVENT.DOC]
[File STRIDE.TXT appended to [52,20]EVENT.DOC]

$
```

output-file-spec

Specifies the file to which the input files are to be appended. (The output file must be on a disk.) If your installation has DECnet/E, you can specify the node where the output file is located.

If the output file does not exist, the following warning messages are displayed:

```
?Can't find file or account - file output-file-spec - continuing
[File input-file-spec copied to output-file-spec]
```

A new output file is then created.

You must specify at least one part of the output file specification. If you do not specify a device and/or directory, the APPEND command uses your directory on the public disk structure. For other fields that you do not specify, the APPEND command uses the corresponding field of the input file specification.

If you specify the asterisk wildcard character (*) in any part of the output file specification, the APPEND command uses the corresponding field of the related input file specification. For example:

```
$ APPEND ONE.*(RET)
To: TWO.*(RET)
[File ONE.TXT appended to [52,20]TWO.TXT]
[File ONE.DAT appended to [52,20]TWO.DAT]

$
```

Command Qualifiers

/LOG
/NOLOG

Controls whether the APPEND command displays the file specifications of each file appended. Unless you specify /NOLOG, the APPEND command displays the file specifications of the input and output files after each append operation.

For example:

```
$ APPEND ATHENS.TXT OHIO.TXT/LOG
[File ATHENS.TXT appended to [52,20]OHIO.TXT]

$ APPEND ATHENS.TXT MASS.TXT
[File ATHENS.TXT appended to [52,20]MASS.TXT]

$ APPEND ATHENS.TXT MAINE.TXT/NOLOG

$
```

/QUERY
/NOQUERY

Determines whether to append individual files to other files. The /QUERY qualifier is only useful when you use wildcards in the input file specification; /NOQUERY is the default.

When you use the /QUERY qualifier, DCL displays a file name followed by a question mark prompt. You can reply with either:

Y	Yes, append the file.
N	No, do not append the file.
CTRL/Z	Skip remaining files, return to command level.
RET	No, do not append the file.

For example, suppose you want to append some of your files with the type .DAT to a file named RACES.TXT:

```
$ APPEND *.DAT TO RACES.TXT/QUERY
RACE1.DAT? Y
[RACE1.DAT appended to [52,20]RACES.TXT]
RACE2.DAT? N
RACE3.DAT? CTRL/Z

$
```

Printing Files

The PRINT, SHOW QUEUE, SET QUEUE/ENTRY, DELETE/ENTRY, DELETE /JOB, and SET QUEUE/JOB commands affect the way your files are printed on hard-copy devices.

The PRINT command queues a file for printing, either on a default system printer or on a device you specify. A *queue* is the list of files to be printed on a specific device.

RSTS/E supports two spoolers for control of queue processing. The Standard RSTS spooling package is larger (and has more features) than the micro–RSTS spooling package. This chapter describes the DCL commands for both large and small spooling packages. The *RSTS/E System Manager's Guide* describes operator commands. Throughout the documentation, the Standard RSTS spooling package is called the "large Spooler", and the micro–RSTS spooling package is called the "small Spooler." The syntax you use depends on which spooling package is available on your system. See your system manager to find out which package is available.

PRINT

Format

PRINT file-spec[,...] [job-spec]

Command Qualifiers	Defaults
/AFTER=date-time	
/[NO]CONVERT	/NOCONVERT
/[NO]FEED	/FEED
/FORMS=form-name	/FORMS=NORMAL
/JOB_COUNT=n	/JOB_COUNT=1
/NAME=job-name	
/OWNER=PPN	
/PRIORITY=n	/PRIORITY=128
/QUEUE=queue-name[:]	
/[NO]TRUNCATE	/NOTRUNCATE

File Qualifiers	Defaults
/COPIES=n	/COPIES=1
/[NO]DELETE	/NODELETE
/[NO]FLAG_PAGES	/FLAG_PAGES

Prompts

File: file-spec[,...]

Note

In the small Spooler, /[NO]CONVERT, /[NO]FEED, and /[NO]TRUNCATE can also be file qualifiers. These qualifiers may be placed after a file to indicate that only the file you specify is affected.

When you use the PRINT command, your request is placed in a queue with other files to be printed. You can determine the order in which files are printed by:

1. The priority of the request. A higher priority job is always printed before a lower priority job. (Normally the system assigns a priority of 128.)
2. The order in which requests are submitted. Print requests are assigned sequence numbers in ascending (1, 2, 3,...) order. Priorities being equal, the lower the sequence number, the sooner a job is printed. For example, a job with a sequence number of 968 is printed before a job numbered 1238.

To request a hard copy of a file such as FINE.TED, type:

```
$ PRINT FINE.TED
```

The file or files you specify with a PRINT command are considered one print job. (A *print job* has no relationship to the job number RSTS/E assigns to each user at the start of a session at the terminal.)

The system assigns a unique sequence-number to each queued job in the system. For example, three consecutive jobs in the queue could have sequence-numbers 4266, 4267, and 4268. These numbers show the order in which the jobs entered the queue.

Printer queues are identified by a queue-name (which is generally, but not always, a device name). If you do not specify a queue-name with the /QUEUE qualifier, the system queues the job to the default print queue.

Command Parameters

file-spec[,...]
Specifies one or more files to be printed. If you specify more than one file, separate the file specifications with commas (,) or plus signs (+). In either case, the PRINT command prints all the files as a single print job.

In the large Spooler, you can use wildcard characters for the file name, and file type, but not for the directory. You can use wildcard characters for the file name, file type, and directory in the small Spooler. For both spoolers, if you do not specify a file type, the PRINT command uses the default file type of .LST.

You can specify a node name as part of the file specification with the large Spooler, but not with the small Spooler. For both spoolers, if you specify a file on a remote node, the file is temporarily copied into your account, printed, and then deleted.

job-spec
queue-name:[proj,prog]job-name

Indicates the job specification for a file to be printed. This parameter is optional. See Chapter 2 (Entering Job Specification Lists) for a description of the parameters.

There are some differences between the two spoolers:

Large Spooler

The default for queue-name is LP0:. The default for job-name is the first file name, regardless of wildcards. The job-name can contain one to six characters.

Small Spooler

The default for queue-name is PRINT:. The default for job-name is PRINTJOB if the first file name contains wildcards. The job-name can contain one to nine characters.

The default for the project-programmer number is your own PPN for both spoolers.

You cannot specify /NAME, /QUEUE, or /OWNER in conjunction with the job-spec parameter. You will receive an error message if you specify any combination of these qualifiers with the job-spec parameter.

On systems with DECnet/E and the large Spooling package, you can print a file on another system by including a node name in the file specification. When you do specify a file on another system, the system copies the file into your account, prints it, and then deletes it.

Command Qualifiers

/AFTER=date-time
Requests that the file not be printed until a specific time of day or date. Use the standard syntax rules for specifying date and time values.

For example, to have a file named RICHMO.BRG start printing at 5:30 PM on September 11 (after normal working hours when there are fewer requests for listings), specify the date and time as:

```
$ PRINT RICHMO.BRG/AFTER=11-SEP:05:30PM
```

If the specified date or time has already passed, the file is queued for printing immediately.

/CONVERT
/NOCONVERT
Specifies whether all zeros should be converted to the letter O before printing. The default is /NOCONVERT.

/FEED
/NOFEED
Specifies whether the printer should leave six blank lines at the end of each page. The default is /FEED.

/FORMS=form-name
Specifies the form-name required for the specified file(s).

You determine the form-name with an alphanumeric name, which is defined at your installation. (Check with your system manager to learn these names.) The default is NORMAL.

/JOB_COUNT=n
Requests that the entire job be printed n times, where n is a decimal number from 1–9 in the large Spooler and 1–255 in the small Spooler. By default, the job is printed once.

The following example prints four copies of a file named BLUTO.UGH:

```
$ PRINT BLUTO.UGH/JOB_COUNT=4
```

/JOB_COUNT differs from the /COPIES qualifier in the order in which multiple files are printed. For example, if you request three copies of files FREDA.JOE and SHULTZ.SUE, the /JOB_COUNT qualifier prints one copy of FREDA.JOE, followed by one copy of SHULTZ.SUE, and repeats the grouping until three copies of each file are printed:

```
$PRINT FREDA.JOE,SHULTZ.SUE/JOB_COUNT=3
```

/NAME=job-name

Defines a one- through six-alphanumeric character name to identify the job for the large Spooler; the small Spooler can contain one to nine characters. You can display the job-name with the SHOW QUEUE command. In addition, the job-name is printed on the front page (also known as the *flag* or *banner* page) of the job's hard copy.

You can use the job-name with the SET QUEUE/JOB and DELETE/JOB commands.

If you do not specify /NAME in the large Spooler, the name defaults to the first file name in the job. In the small Spooler, the job-name defaults to PRINTJOB if the first file name contains wildcards.

/OWNER=[proj,prog]

Indicates the owner of the job to be printed. The project-programmer number is displayed on the job header page. If you are nonprivileged, you can specify only jobs with your PPN.

/PRIORITY=n

Specifies the priority of the print job. The priority, n, can be in the range 0 through 255. If you do not specify a priority number, the value 128 is assumed.

A nonprivileged user can specify a priority of 0 to 128. A privileged user can set a higher priority (up to 255), in which case the job will normally be printed first.

For example, in the large Spooler, a nonprivileged user can reset the priority of file STRIDE.TXT in the print queue to 100 by typing:

```
$ PRINT STRIDE.TXT/PRIORITY=100RET
```

However, a nonprivileged user cannot set a priority of more than 128:

```
$ PRINT STRIDE.TXT/PRIORITY=192RET

?Bad switch value - PR
```

The message "?Bad switch value – PR" corresponds to /PRIORITY=192.

/QUEUE=queue-name[:]

Requests that the specified file(s) be queued for printing on a specific device. If you do not specify the /QUEUE qualifier, files are queued by default to LP0: in the large Spooler and PRINT: in the small Spooler. If you specify /QUEUE=LP:, the job is printed on the first available printer.

For example, in the large Spooler, you can specify print queue LP1: and be prompted for a file specification by typing:

```
$ PRINT/QUEUE=LP1:RET
Files: STRIDE.*
```

You can use the queue named LP2: to print out a file named DAVE.DAT by typing:

```
$ PRINT DAVE.DAT/QUEUE=LP2:RET
```

If you queue a job to a queue that does not exist on your system using the large Spooler, no error message is displayed, but your command is ignored. However, you can use the SHOW QUEUE command (described in the next section) to verify a file's position in the queue.

/TRUNCATE
/NOTRUNCATE

Indicates whether lines that exceed the width of the page should be truncated. The PRINT command determines what page width to use according to the form you specify with /FORMS, together with the characteristics of that form as defined by your system manager.

If you specify /NOTRUNCATE, lines that exceed the form width are "wrapped" onto the next line. The default is /NOTRUNCATE.

File Qualifiers

/COPIES=n

Specifies the number of copies to print. By default, the PRINT command prints one copy of a file; you can use /COPIES to request up to 255 copies. For example, the following command line requests three copies of file RIP.TXT:

```
$ PRINT RIP.TXT/COPIES=3RET
```

If you specify /COPIES after the PRINT command name, each file is printed the number of times you specify. For example, to print three copies each of ETFILM.TXT and FAME.MEM, type:

```
$ PRINT/COPIES=3 ETFILM.TXT,FAME.MEMRET
```

If you specify /COPIES after a file specification, only that file is printed the specified number of times. The following command line requests two copies of CLEONE.DAT, one copy of WES.DAT (by default), and five copies of SCAMP.DAT:

```
$ PRINT CLEONE.DAT/COPIES=2,WES.DAT,SCAMP.DAT/COPIES=5RET
```

The /COPIES qualifier differs from /JOB_COUNT in the order in which multiple files are printed. For example, if you request three copies of files WINKLE.NAN and MODNE.LIN, /COPIES prints three copies of WINKLE.NAN, followed by three copies of MODNE.LIN:

```
$ PRINT/COPIES=3 WINKLE.NAN,MODNE.LINRET
```

/DELETE
/NODELETE

Controls whether files are deleted after printing. For example, when a file is the output of a program and can be created again, you can safely delete the file after printing to save disk space.

If you specify /DELETE after the PRINT command name, all files specified are deleted:

```
$ PRINT/DELETE OUTPUT.DAT,RESULT.TXT
```

If you specify /DELETE after a file specification, only that file is deleted after it is printed:

```
$ PRINT ACTON.MAS,BOXBRO.MAS,MAYNRD.MAS/DELETE
```

Only the file MAYNRD.MAS is deleted.

In the following command line, only file END.DAT is deleted after printing:

```
$ PRINT START.DAT,END.DAT/DELETE,MIDDLE.DAT
```

To delete a file after printing, you must have write or delete access to the file. If you do not have sufficient privileges to delete the file, the delete qualifier is ignored. In addition, you cannot specify /DELETE with wildcard characters in the large Spooler. Wildcard characters are allowed in the small Spooler.

The /NODELETE qualifier is the default. Files are not deleted after printing unless you specify /DELETE.

/FLAG_PAGES
/NOFLAG_PAGES

Specifies whether flag pages should be printed ahead of each file listing. If FLAG_PAGES is specified, the number of flag pages printed is determined by your system manager and may vary according to the form you specify with the /FORMS=form-name qualifier.

SHOW QUEUE

The SHOW QUEUE command displays information about jobs in the printer or batch job queue.

Format

SHOW QUEUE [job-spec]

Command Qualifiers

/ALL	
/BATCH	(large Spooler only)
/BRIEF	
/FILES	
/FORMS = form-name	(small Spooler only)
/FULL	

To display the full queue listing of your system's default printer, type:

```
$ SHOW QUEUE/FULL↵

LP0 QUEUE LISTING          14-Dec-82    03:10 PM
UNIT     JOB               S / P FILES
 0       MANGO [52,20]/SE:10738/FO:NORMAL/SKI
                           A /128/AF:14-Dec-82:10:30 PM/TY:EMB
                                     SY:[52,20]MANGO.MAN

$
```

The example shows the status of file MANGO.MAN. Included in the display is the following:

1. Unit number (0) of the queue
2. Job name (MANGO, which is taken from the file name)
3. Account of the user who submitted the job ([52,20])
4. Sequence or entry number (/SE:10738)
5. Forms, if it is an LP queue (/FO:NORMAL)
6. Status (A)
7. Priority (128)
8. Print qualifiers, if specified (/AFTER = 10:30)

The status of a print job can be one of the following:

O Job is in the queue and is waiting to be processed.

S Job has been sent to the spooler.

A Job is waiting for an /AFTER date or time to expire.

H Job is in hold status. (The /RELEASE qualifier of the SET QUEUE command is required to process the job.)

K Job is in kill status. (In other words, your job will stop printing, but it will not be deleted from your directory.)

Some status indicators may appear together. For example, SK means that the job was sent to the spooler and that a DELETE/JOB or DELETE/ENTRY command was later issued for it.

If there are no files in the queue, RSTS/E prints a message similar to:

```
LP0 queue is empty
```

The name of the queue in the example is LP0:.

Command Parameters

job-spec
queue-name:[proj,prog]job-name

Indicates the job specification for a file to be displayed. This parameter is optional. See Chapter 2 (Entering Job Specification Lists) for a description of the parameters.

There are some differences between the two spoolers:

Large Spooler

The default for the queue-name parameter is LP0:. Job-names can contain one to six characters.

Small Spooler

If you do not specify a queue-name, the default is PRINT:. Job-names can contain one to nine characters.

The job-name parameter defaults to an asterisk (*) for both spoolers. Also, the project-programmer parameter defaults to your own PPN for both spoolers.

Command Qualifiers

/ALL
Indicates that wildcard defaults should be used in the job-spec. This is an easier way to list all jobs. If you do not specify /ALL, the job specification defaults to all jobs with your project-programmer number.

/BATCH

Displays entries in the generic batch job queue, BA:. The /BATCH qualifier applies to the large Spooler only. In addition, you cannot specify /BATCH if you include a queue-name in the job specification. A job queued to BA: is processed by the first available copy of the batch program. For example:

```
$ SHOW QUEUE/BATCH/FULL[RET]

BA QUEUE LISTING              21-Feb-83          10:00 AM
UNIT      JOB                 S / P       FILES
 *        ARKOFF[1,9]/SE:8077
                              H /128/AF:15-Jan-83:03:00 AM/TY:EMD
                                          SY :[1,0]ARKOFF.CTL

$
```

You can specify either a specific batch queue name or the /BATCH qualifier, but not both. See Chapter 6, "Batch Processing," for a description of submitting jobs to a batch queue.

/BRIEF

Requests a brief listing of information about jobs in the queue. This is the default for both spoolers. For example:

```
$ SHOW QUEUE/BRIEF[RET]

LP0 SHORT QUEUE LISTING 13-Dec-82          10:13 AM
UNIT      JOB                                 S / P
 0        GRNOLA [2,214]/SE:10777/FO:NORMAL   S /120

$
```

/FILES

Indicates that all file specifications and their qualifiers included in the job should be displayed. However, if you are nonprivileged, only files with your PPN are displayed. With the small Spooler, /FILES is a subset of /FULL. /FULL gives you other information as well. With the large Spooler, /FILES and /FULL display the same information.

/FORMS[=form-name]

Indicates that only jobs requiring a form-name matching the one specified should be displayed. The form-name may include wildcard characters. For example, "/FORMS=NOR???".

If /FORMS is specified without an argument, then only jobs with the default form-name NORMAL will be displayed, making the qualifier equal to /FORMS=NORMAL.

If the /FORMS qualifier is not included, then jobs are listed without regard to the forms they require.

/FORMS applies to the small Spooler only.

/FULL

Indicates that full job status should be displayed. The display includes the same information when you type SHOW QUEUE, except that the names of files to be processed are listed. The small Spooler display also gives you the time the job was submitted and the time it started. For example:

```
$ SHOW QUEUE/FULL<RET>

Job 5   [52,20] NORMAN   Status AFTER  Pri 128  Form NORMAL  Pos 1st
  Submitted 13-Dec-82 03:26 PM, Start after 13-Dec-82 10:00 PM
   _SY:[52,20]BOWER.TXT
```

SET QUEUE/JOB

The SET QUEUE/JOB command uses the name of a job to modify the status of a file that is queued in a printer or batch queue. This command applies to the large Spooler only.

Format

SET QUEUE/JOB [=] job-spec

Command Qualifiers	**Defaults**
/AFTER=date-time	None
/BATCH	
/FORMS=form-name	
/HOLD	
/JOB_COUNT=n	
/PRIORITY=n	
/RELEASE	

Prompts

Job:	job-spec

When you submit a batch job or issue the PRINT command, the job is assigned a name, according to the first input file specification or the name you specify. You can use this name to modify the status of the job in the queue.

For example, suppose you want to print the file MIKEGR.STN on a specific printer, such as LP1:. First, you type:

```
$ PRINT/QUEUE=LP1: MIKEGR.STN<RET>
```

You then decide to lower the job's priority, so you type:

```
$ SET QUEUE/JOB=MIKEGR/PRIORITY=100 LP1:<RET>
```

Command Parameters

job-spec

queue-name: [proj,prog] job-name

Specifies the name of one or more jobs you want to modify in the queue. If you are nonprivileged, you can modify jobs only if they were queued from your account. (Your account is the default project-programmer number.) If you do not specify a queue-name, the system assumes the default name of LP0:. (You cannot assign logical names to queues.)

If you use a wildcard (*) for job-name, this denotes all jobs in the queue that were submitted or printed from your account. Note that if no jobs in the queue have the specified name, the command has no effect, but no error message is displayed. There is no default for the job-spec parameter.

See Chapter 2 (Entering Job Specification Lists) for a description of the parameters.

Command Qualifiers

/AFTER = date-time

Requests that the specified job be held until a specific date and time and then released for printing or processing. For example, you type the following command line to have a job named RONNA printed after August 2 at 1:00 in the afternoon:

```
$ SET QUEUE/JOB=RONNA/AFTER=02-AUG-83-01:00PM(RET)
```

If the specified date or time has already passed, the file is released immediately.

/BATCH

Requests the generic batch queue BA:. You may specify /BATCH in place of the queue-name (in job-spec) to request the generic batch queue.

For example, you type the following to release job VENDOR for processing, which is currently on hold status in the batch queue:

```
$ SET QUEUE/JOB=VENDOR/BATCH/RELEASE(RET)
```

/FORMS = form-name

Modifies the form-name for the specified job. This qualifier overrides the form-name specified or defaulted in the PRINT command. For example:

```
$ SET QUEUE/JOB=RAPID/FORMS=NORMAL(RET)
```

/HOLD

Requests that the specified job(s) be placed in a hold status. Jobs in a hold status are not processed until you release them with the /RELEASE qualifier. The /HOLD qualifier is meaningful only to jobs already placed in a queue. The qualifier might be useful if you need to postpone printing the job until a later time. (For example, if other users have crucial time constraints for getting listings).

Suppose you have a job in the queue named LOIS that you would like to print at a later time, type:

```
$ SET QUEUE_JOB=LOIS/HOLD(RET)
```

/JOB_COUNT=n

Specifies the number of copies of the job to print where n is a decimal number from one to nine. This qualifier overrides the /JOB_COUNT qualifier specified or assumed in the PRINT command. The job count is ignored for batch jobs. The following example prints two copies of a job named TAMPA:

```
$ SET QUEUE/JOB=TAMPA/JOB/COUNT=2RET
```

/PRIORITY=n

Changes the priority of a job relative to other jobs that are currently queued. The priority n must be in the range of 0 through 255, where 0 is the lowest priority and 255 is the highest. By default, the job's priority is not changed.

Nonprivileged users can assign a priority no higher than 128; privileged users can assign a value up to 255. For example, if you want to set the priority of a job named MORGAN to 40, type:

```
$ SET QUEUE/JOB=MORGAN/PRIORITY=40RET
```

/RELEASE

Releases a job that was previously held from processing with the /HOLD qualifier.

For example, suppose you put file SPLITS.DAT on hold in the print queue:

```
$ PRINT SPLITS.DATRET
$ SET QUEUE/JOB=SPLITS/HOLDRET
```

Later, you release the job from its hold status by typing:

```
$ SET QUEUE/JOB=SPLITS/RELEASERET
```

The /RELEASE qualifier causes the system to print file SPLITS.DAT in its turn in the queue.

SET QUEUE/ENTRY

The SET QUEUE/ENTRY command uses the sequence number of a job to modify the status of a file that is queued for a printer or batch queue. This command applies to the large Spooler only.

SET QUEUE/ENTRY has the same function as SET QUEUE/JOB, except that the /ENTRY qualifier uses a sequence number, while the /JOB qualifier uses a job-name.

SET QUEUE/ENTRY is useful when you have submitted or printed two or more jobs by the same name, and you want to change the status of only one of them.

Format

SET QUEUE/ENTRY[=]sequence-number [queue-name[:]]

Command Qualifiers	**Defaults**
/AFTER=date-time	None
/BATCH	
/FORMS=form-name	
/HOLD	
/JOB_COUNT=n	
/PRIORITY=n	
/RELEASE	

Prompts

Entry:	sequence-number

The system assigns a unique entry number, called a sequence-number, to each queued printer or batch job in the system. Use this sequence-number to specify the entry you want to change.

Issue the SHOW QUEUE command to learn the sequence-number. (There may be a delay between the time when you submit the job and the time when it shows up in the SHOW QUEUE listing.)

For example, suppose you want to print the file SMITH.MAR on a specific printer. First, type:

```
$ PRINT SMITH.MAR<RET>
```

Then check the file's position in the queue by entering the SHOW QUEUE command. The sequence-number of each job in the queue is shown after the /SE: qualifier in the display. You need the sequence-number for the SET QUEUE/ENTRY command:

```
$ SHOW QUEUE/ALL/FULL↵

LP0 QUEUE LISTING          12-Mar-83    09:58 AM
UNIT     JOB               S / P      FILES
 0       STRIDE[52,20]/SE:5600/FO:NORMAL
                           A/128/AF:10-Mar-83:11:59 PM/TY:EMB
                                      SY :[2,214]STRIDE.???

 0       BRIAN [2,250]/SE:5601/FO:NORMAL
                           S /128/TY:EMB
                                      DR0:[160,51]KIT.CMD

 0       SMITH [52,30]/SE:5602/FO:NORMAL
                           0 /128/TY:EMB
                                      DR0:[52,30]SMITH.MAR

$
```

The job named SMITH has the sequence number 5602, which is shown by /SE:5602.

You can use the SET QUEUE/ENTRY command to have three copies of file SMITH.MAR printed. Use the sequence number 5602 and the /JOB_COUNT qualifier:

```
$ SET QUEUE/ENTRY=5602/JOB_COUNT=3↵
```

Command Parameters

sequence-number
: Specifies the sequence-number of the job you want to change.

queue-name[:]
: Specifies the name of the queue in which the specified file is entered. (You cannot assign logical names to queues.)

  If you do not specify a queue-name, the system assumes the default name of LP0:.

The command qualifiers are the same as for SET QUEUE/JOB, which is described in the previous section.

DELETE/JOB

The DELETE/JOB command uses the name of a job to cancel a request to the print or batch queue.

Format

DELETE/JOB [=] job-spec

Command Qualifiers

/BATCH (large Spooler only)

Prompts

Job: job-spec

For example, if you decide after you make your print request that you do not want a hard copy of the file after all, you can use the DELETE/JOB command to withdraw your request. (If the file is printed before you enter the DELETE/JOB command, your request is too late. However, if your file is in the middle of printing, the file stops printing.)

By default, files submitted to the queue are given the same job-name as the file name. For example:

```
$ PRINT MAYOR.DEM<RET>
```

The file is assigned the job-name MAYOR in the queue. You delete the job from the queue by typing:

```
$ DELETE/JOB=MAYOR<RET>
```

You can delete multiple jobs at once if they have the same name, or if you specify a wildcard with an asterisk (*). However, you can specify only one job-name at a time:

```
$ DELETE/JOB=JOE,LOUIS<RET>
  DELETE/JOB=JOE,LOUIS<RET>
                      ^
?Invalid character
```

Command Parameters

job-spec

queue-name:[proj,prog]job-name

Indicates the job specification of one or more jobs to be deleted from the queue. Job specifications are not unique identifiers for jobs. For example, more than one

job may exist with the same job specification. Therefore, you should be careful when you delete jobs using the job-spec parameter. See Chapter 2 (Entering Job Specification Lists) for a description of the parameters.

There are some differences between the two spoolers.

Large Spooler

The default queue-name is LP0:. You may specify one to six characters for the job-name. If no jobs in the queue match the job-name you specify, the command has no effect; however, an error message is not displayed.

Small Spooler

The default queue-name is PRINT:. You may specify one to nine characters for the job-name.

You cannot assign logical names to queues for either spooler. In addition, if you do not specify a project-programmer number, the default is your own. If you use a wildcard character (*) for the job-name, you denote all jobs in the queue that were submitted or printed from your account. There is no job-name default for either spooler.

Command Qualifiers

/BATCH

Deletes an entry from the generic batch queue. You can use this qualifier with the large Spooler only.

For example, the following command string deletes a job named MORGAN from the batch queue:

```
$ DELETE/JOB=MORGAN/BATCH(RET)
```

You can specify either the /BATCH qualifier or the name of a specific batch queue, but not both. See Chapter 6, "Batch Processing," for a discussion of batch queues.

DELETE/ENTRY

The DELETE/ENTRY command uses the sequence-number of a job to cancel a request to the print or batch queue.

DELETE/ENTRY has the same function as DELETE/JOB, except that the /ENTRY qualifier uses a sequence-number, while the /JOB qualifier uses a job-spec.

DELETE/ENTRY is useful when you have submitted or printed two jobs by the same name, and you want to cancel only one of them.

Format

DELETE/ENTRY[=]job-number [queue-name[:]]

Command Qualifiers

/BATCH (large Spooler only)

Prompts

Entry: sequence-number

Suppose you make a print request for the wrong file, and then realize you made a mistake. The DELETE/ENTRY command enables you to delete the entry from the print queue by number. For example:

```
$ PRINT SIMMON.RIC(RET)
```

To delete the entry from the queue, verify the sequence-number with SHOW QUEUE. (The sequence-number is indicated by /SE: in the display for the large Spooler or Job nnn for the small Spooler.) Then delete the job by its sequence-number. (This example uses the large Spooler.)

```
$ SHOW QUEUE/FULL(RET)

LPO QUEUE LISTING        07-Feb-83   12:18 PM
UNIT     JOB             S / P FILES
 0       SIMMON[52,20]/SE:10738/FO:NORMAL/SKI
                         A /128/ PM/TY:EMB
                                    SY:[52,20]SIMMON.RIC

$ DELETE/ENTRY=10738(RET)
```

In the large Spooler, you can enter only one number at a time. Otherwise, an error message is displayed:

```
$ DELETE/ENTRY=10765,10774(RET)
  DELETE/ENTRY=10765,10774(RET)
                    ^

?Invalid character
```

If the file is printed before you enter the DELETE/ENTRY command, your request is too late. However, if your file is in the middle of printing, the file stops printing.

Command Parameters

sequence-number
: Specifies the job number to be deleted from the queue. DELETE/ENTRY accepts a list of sequence-numbers in the small Spooler, but will accept only one number in the large Spooler.

queue-name[:]
: Names the queue in which the job(s) exist. LP: is a generic name for all print queues. BA: is a generic name for all batch queues. If you do not specify a queue-name for the large Spooler, LP0: is the default. The default queue-name for the small Spooler is PRINT:.

Command Qualifiers

/BATCH
: Deletes an entry from the generic batch queue, BA:. This qualifier is valid for the large Spooler only. Type SHOW QUEUE/BATCH to determine the sequence-number, which you use in the DELETE/ENTRY/BATCH command line. For example:

```
$ SHOW QUEUE/BATCH<RET>

BA SHORT QUEUE LISTING 21-Jan-83   10:00 PM
UNIT  JOB                            S / P
 *    BANANA[1,9]/SE:8077/FO:NORMAL A/128/AF:21-Jan-83:2:00PM

$
```

The sequence-number of the job named BANANA is 8077. Type the following to delete the job from the queue:

```
$ DELETE/ENTRY=8077/BATCH<RET>
```

You can specify either the /BATCH qualifier or the name of a batch queue (for example, BA:), but not both. See Chapter 6 (Batch Processing) for a discussion of batch queues.

Comparing Files: DIFFERENCES

The DIFFERENCES command compares two text files and lists any sections of text that differ between the two files.

Format

DIFFERENCES input-file-spec compare-file-spec

Command Qualifiers	**Defaults**
/IGNORE=BLANKLINES	
/MATCH=size	/MATCH=3
/MAXIMUM_DIFFERENCES=n	/MAXIMUM_DIFFERENCES=300
/OUTPUT=file-spec	/OUTPUT=KB:

Prompts

File 1: input-file-spec
File 2: compare-file-spec

DIFFERENCES compares the two files by groups of lines and produces an output file if the /OUTPUT qualifier is used. Otherwise, output is sent to the terminal. This output file displayed at your terminal lists the differences, if any.

The qualifiers for the DIFFERENCES command can be categorized according to their functions:

- /IGNORE=BLANKLINES requests DIFFERENCES to ignore blank lines while comparing files.

  By default, DIFFERENCES compares every line in each file.

- The /MATCH and /MAXIMUM_DIFFERENCES qualifiers control the extent of the comparison.

  By default, DIFFERENCES reads every line in the first input file and looks for a matching line in the second input file. Lines are considered matched only if three identical sequential lines are found in each file.

The DIFFERENCES command normally displays its report at your terminal. Use the /OUTPUT qualifier to request DIFFERENCES to write the output to an alternate file or device. If the two files are significantly different, then RSTS/E may print:

```
?Maximum memory exceeded
```

For example, you may want to use DIFFERENCES to compare a file (such as HOME.WRK) with its earlier backup version (HOME.BAK). The command line you enter is:

```
$ DIFFERENCES HOME.WRK HOME.BAK
```

RSTS/E displays a message telling you which files it is comparing:

```
Comparing: 1) SY:[2,214]HOME.WRK to 2) SY:[2,214]HOME.BAK
```

Because you did not include a directory in the command line, the default is your directory.

Next, RSTS/E displays the lines containing text that does not match. The sections in the following example are labeled 1) and 2), corresponding to the first and second file being compared. The sections of text are separated by rows of asterisks (*):

```
******************************
1) SY:[2,214]HOME.WRK
the writings of Milton and
Donne. Fortunately, I find
their work interesting.
**********
2) SY:[2,214]HOME.BAK
the writings of Schultz and
Cronin. Fortunately, I find
their work interesting.

?1 Difference Found

$
```

Notice that the numbers 1) and 2) correspond to HOME.WRK and its backup version, HOME.BAK. DIFFERENCES lists the differences in the two files one group of lines at a time.

Command Parameters

input-file-spec
: Specifies the name of the first input file to be compared.

  You must include a file name. The default file type is null. No wildcard characters are allowed in the file specification.

compare-file-spec
: Specifies the name of the second input file to be compared.

  You must include a file name. The default file type is null. No wildcard characters are allowed in the file specification.

Command Qualifiers

/IGNORE = BLANKLINES

Specifies that blank lines between data lines are to be ignored during the comparison. By default, the DIFFERENCES command compares every line in each file and reports all differences.

For example, if files BURNS.JON and BURNS.JEN have similar contents but have unequal numbers of blank lines, you can disregard the blank lines in their comparison by typing:

```
$ DIFFERENCES BURNS.JON BURNS.JEN/IGNORE=BLANKLINES
```

/MATCH = size

Controls the number of lines that must be identical for DIFFERENCES to consider them a match. For example, to see that five lines are included in a match between NANCY.TXT and DAVID.TXT, type:

```
$ DIFFERENCES NANCY.TXT DAVID.TXT/MATCH=5
```

By default, after DIFFERENCES finds unmatched lines, it assumes that the files match after it finds 3 sequential lines that match. The match size you specify overrides the default value of 3.

/MAXIMUM_DIFFERENCES = n

If you specify /MAXIMUM_DIFFERENCES, DIFFERENCES terminates after locating n differences. The output file lists differences only on lines compared until the maximum has been reached.

For example, you may need to compare two large data files (MARTHA.DAT and MUFF.DAT), but if more than four differences are found, any further comparison would not be worth your while. In this case you type:

```
$ DIFFERENCES MARTHA.DAT MUFF.DAT/MAXIMUM/DIFFERENCES=4
```

By default, DIFFERENCES stops after 300 differences are found.

/OUTPUT = file-spec

Stores the result of the comparison in a file rather than displaying it at your terminal. If you omit the /OUTPUT qualifier, the output is displayed at your terminal.

The /OUTPUT qualifier is especially useful if you are using a video terminal and want a listing of the output. For example, you can request that the output from comparing files RACE.A and RACE.B be stored in a file that you name RESULT.C:

```
$ DIFFERENCES RACE.A RACE.B/OUTPUT=RESULT.C
```

DIFFERENCES

If you specify /OUTPUT without a file specification, the output is directed to a file with the same file name as the first input file and a file type of DIF. If you had not assigned the file name RESULT.C in the previous example, it would have been named RACE.DIF.

When you specify /OUTPUT, the output file type defaults to DIF.

No wildcard characters are allowed in the file specification.

Allowing Access to Your Files: SET PROTECTION

This section describes what it means to assign protection codes to your files. Then it explains how to use the SET PROTECTION command and the /DEFAULT qualifier.

Using Protection Codes

A protection code determines who has access to a file.

The protection code consists of one to three decimal digits. This code determines the file's degree of protection on two levels:

- The actions — reading, writing, executing, and deleting — against which it is protected. (Anyone who can write in a file can also delete it, and vice versa.)
- The user or class of users against whom it is protected.

RSTS/E recognizes four classes of users for protection purposes:

1. Each user, known to the system by a PPN (such as [52,20]).
2. The project group, known to the system by a project number (the 52 in [52,20]).
3. Privileged users, known by a project number of 1 (such as [1,20]).
4. All other users on the system, known to the system by a project number that is different from yours.

For example, 60 is the default protection code that RSTS/E normally assigns. (The code is something other than 60 if your system manager changed the default, or if you use the SET PROTECTION/DEFAULT command described in the next section.) If one of your files has a protection code of 60, only you or a privileged user can read, edit, or delete it. A protection code remains fixed unless you change it.

When you create a file, DCL assigns it the default protection code 60:

```
$ CREATE SUE.DAT(RET)
One line files are fun.(RET)
(CTRL/Z)

$ DIR SUE.DAT(RET)

 Name .Typ   Size    Prot    Name .Typ   Size    Prot    SY:[52,20]
SUE   .DAT      1   < 60>

Total of 1 block in 1 file in SY:[52,20]
```

You assign a protection code with the SET PROTECTION command, as in:

```
$ SET PROTECTION=40 SUE.DAT(RET)
SUE   .DAT renamed to SUE   .DAT<40>

$ DIR SUE.DAT(RET)

 Name .Typ   Size    Prot    Name .Typ   Size    Prot    SY:[52,20]
SUE   .DAT      1   < 40>

Total of 1 block in 1 file in SY:[52,20]
```

A factor in assigning protection codes is whether a file is executable. An *executable* file is produced by the LINK command or by the COMPILE command of BASIC–PLUS or BASIC–PLUS–2. Such a file can be run; in other words, it is an executable program. All other files are considered nonexecutable.

A protection code is the sum of any of the numbers from Table 3–2.

Table 3–2: File Protection Codes

Code	Meaning
1	Protection against reading by owner.
2	Protection against writing by owner.
4	Protection against reading by owner's project number.
8	Protection against writing by owner's project number.
16	Protection against reading by anyone who does not have the owner's project number.
32	Protection against writing by anyone who does not have the owner's project number.
64	Executable program, which can be run.
128	A protected data file; overwritten with zeros when deleted. (This code is ignored if specified by a nonprivileged user.)
Individual codes added to the executable protection 64 have meanings different from those above. These compiled codes are:	
1	Execute protection against the owner.
2	Read and write protection against the owner.
4	Execute protection against the owner's project number.
8	Read and write protection against the owner's project number.
16	Execute protection against all others who do not have the owner's project number.
32	Read and write protection against all others who do not have the owner's project number.
128	Program with privileges. (This code is ignored if specified by a nonprivileged user.)

The protection code you set for a file is the total of the code numbers for the types of protection you want. For example, the usual system default (60) protects against reading, writing, and deleting by all users except its owner, because it is a sum of 4, 8, 16, and 32.

Table 3–3 lists common codes and their meanings.

Table 3–3: Common File Protection Codes

Protection Code	Sum of Codes	Meaning
124	64+32+16+8+4	Executable file that is protected against reading, writing, and execution by anyone except you.
104	64+328	Executable file that is protected against reading and writing by anyone except you. However, anyone can execute the file.
62	32+16+8+4+2	Protected against reading by anyone except you. Protected against accidental deletion or modification by you or anyone else.
60	32+16+8+4	Protection against reading and writing by anyone except owner or a privileged user.
48	32+16	Protection against reading and writing by anyone who does not have owner's project number. Can be read or written by any-one in your group.
40	32+8	Protection against writing by anyone except owner; readable by everyone.
42	32+8+2	Protection against writing by anyone including owner.
0		No protection (any user can read, write, and delete).

Note that the assignment of a PPN is related to file protection. The PPN is the test RSTS/E uses to grant or refuse access to a file. For this reason, assignment of account numbers must consider all security needs in a community of RSTS/E users.

SET PROTECTION

The SET PROTECTION command specifies the protection code of a file. You assign a protection code to determine who else, if anyone, can have access to your files. (To check the protection codes of your files, use the /PROTECTION qualifier with the DIRECTORY command.)

A file specification and protection code are required unless you use the /DEFAULT qualifier. You must include either an equal sign or a space between the command SET PROTECTION and the protection code you specify.

Format

SET PROTECTION[=]nn filespec[,...]

Qualifiers	**Defaults**
/DEFAULT	
/[NO]QUERY	/NOQUERY
/[NO]LOG	/LOG

For example, you set the protection code of a file named TIME.TRY to 42 by typing:

```
$ SET PROTECTION=42 TIME.TRY(RET)
```

If you omit information, you are prompted for a protection code and file name. For example:

```
$ SET PROTECTION(RET)
Protection code:   42(RET)
Files:     CRATER.WLK(RET)
CRATER.WLK renamed to CRATER.WLK<42>

$
```

Command Parameters

nn

The value you specify for nn can be any of the protection codes listed in the preceding section, depending on whether you are a privileged or nonprivileged user.

If you are a nonprivileged user and you attempt to specify a protection code which includes 128, the 128 code is ignored.

file-spec

You can include one or more file specifications in the command string. Wildcard characters are allowed.

Command Qualifiers

/DEFAULT

The /DEFAULT qualifier specifies the default protection code of each file you create. (You can override the default for individual files.) /DEFAULT stays in effect until you log out.

If you use SET PROTECTION/DEFAULT, RSTS/E assigns the protection code you specify to all files you create during the current session. Do not include a file specification when you use the /DEFAULT qualifier. Also, you must include either an equal sign or a space between the command and the protection code:

SET PROTECTION[=]nn/DEFAULT

You can override the default protection code for individual files by using the /PROTECTION qualifier.

For example, the following command line tells DCL that every file you create until the time you log out will have a protection code of 40:

```
$ SET PROTECTION=40/DEFAULT(RET)
```

If you do not include a protection code in the command line when you use the /DEFAULT qualifier, then the default code of 60 is assumed, unless your system manager has changed the default on your system.

/QUERY
/NOQUERY

The /QUERY qualifier prompts you to state whether to change a file's protection code. /NOQUERY is the default.

The /QUERY qualifier is useful only when you use wildcards in the file specification. You can enter one of the following responses to the question mark prompt:

Y	Yes, change the protection code.
N	No, do not change the protection code.
(CTRL/Z)	Return to command level.
(RET)	No, do not change the protection code.

For example, if you have several files named WOOLY, you can determine their protection codes as follows:

```
$ SET PROTECTION=40/QUERY WOOLY.*(RET)
[52,20]WOOLY .DAT? Y(RET)
WOOLY .DAT renamed to WOOLY.DAT < 40>
[52,20]WOOLY .LIN? N(RET)
[52,20]WOOLY .JIM? (CTRL/Z)

$
```

Do not use either the /QUERY or /NOQUERY qualifiers with /DEFAULT.

/LOG
/NOLOG

Determines whether DCL displays a message to confirm that a file's protection code was changed. /LOG is the default. To suppress the message when changing the protection code of a file (such as STRIPE.LIS), type:

```
$ SET PROTECTION=64 STRIPE.LIS/NOLOG
$
```

Unless you specify /NOLOG, a message is displayed:

```
$ SET PROTECTION=64 STRIPE.LIS/LOG
STRIPE.LIS renamed to STRIPE.LIS <64>
```

Do not use /LOG or /NOLOG with /DEFAULT.

System Operations 4

This chapter describes the DCL commands that allow you to perform common system operations, such as:

- Displaying system status
- Displaying and setting a terminal's characteristics
- Using physical and logical device names

The commands are listed in Table 4–1. Also included are the pages containing more information about each command.

Table 4–1: System Operation Commands

Using the System	
SHOW	Displays characteristics of the system, your job, or your terminal. (See page 4–2.)
SET	Changes certain default characteristics, which you can specify with an option. (See page 4–3.)
Displaying System Status	
SHOW USERS	Displays information about the status of attached jobs in use on the system. (See page 4–8.)
SHOW SYSTEM	Displays information about the status of all jobs (both attached and detached) in use on the system. (See page 4–10.)
Setting Your Terminal's Characteristics	
SHOW TERMINAL	Displays the characteristics of your terminal. (See page 4–15.)
SET TERMINAL	Changes the characteristics of your terminal. (See page 4–16.)
Physical Device Names and Logical Names	
ASSIGN	Establishes a relationship between a logical name and a physical device or a PPN. (See page 4–28.)
DEASSIGN	Cancels logical name assignments made with the ASSIGN or ALLOCATE commands. (See page 4–30. See also the ALLOCATE command description on page 5–8.)

The SHOW and SET commands let you display and change some of the characteristics of the system, your job, and your terminal.

SHOW

You use SHOW to display information about the current status of your job, the system, or devices on the system.

To specify the type of information you want displayed, use an option with the SHOW command. For example, SHOW TERMINAL displays the characteristics of your terminal.

Format

SHOW option

Options

DEVICES
NETWORK
QUEUE
SYSTEM
TERMINAL
USERS

The SHOW command options are summarized in Table 4–2. Also included in the table are the pages containing more information about each option.

Table 4–2: SHOW Command Options

Option	Displays
DEVICES	The status of devices in use on the system. (See page 5–18.)
NETWORK	The availability of network nodes. (See page 1–23.)
QUEUE	Printer or batch jobs that have been queued but are not yet completed. (See page 3–48. SHOW QUEUE for batch is also described on page 6–21.)
SYSTEM	Information about all jobs on the system. (See page 4–10.)
TERMINAL	The characteristics of a terminal. (See page 4–15.)
USERS	Information about attached jobs on the system. (See page 4–8.)

SET

You use SET and an argument to define or change certain default characteristics, such as the default protection codes the system assigns to your files. The functions of the available options are listed in Table 4–3.

Format

SET option[=]parameter

You must include either an equal sign (=) or a space between the option and the first parameter you specify. For example, you can type:

```
SET HOST = APS::
```

or

```
SET HOST APS::
```

Options

HOST
PROTECTION
QUEUE
TERMINAL

Table 4–3 summarizes the options available with the SET command and provides the pages to refer to for more information about each option.

Table 4–3: SET Command Options

Option	Function
HOST	Specifies the network node to connect your terminal to, so you can log in to that node. (See page 1–23.)
PROTECTION	Determines the protection code of a file. (See page 3–68.)
QUEUE	Modifies the status of a print job or a batch job in a queue. (See page 3–52. SET QUEUE for batch is also described on page 6–23.)
TERMINAL	Specifies the characteristics of a terminal. (See page 4–16.)

Displaying System Status

The commands SHOW USERS and SHOW SYSTEM let you display information about others who are logged in and what system resources they are using. This information might be useful if, for example, you want to know whether a particular user is logged in.

The following is a sample listing you can get with the SHOW USERS command. (For a complete explanation of SHOW USERS, refer to the command description in the next section.)

```
$ SHOW USERS(RET)

RSTS V8.0 *TARA* status at 01-Feb-83, 05:23 PM up 8:19:25

Job    Who     Where   What    Size    State   Run-Time     RTS
14    [SELF]   KB27    SYSTAT   11K    RN LcK     1.6      BAS4F
15     1,210   KB76*   ...MAC   20K     KB       26.2      ...RSX
20    160,45   POJ5    ...MAC   20K     RN     52:05.1     ...RSX

$
```

The elements you may find in a system status display, such as in the above example, are included in Table 4–4. You might want to refer to this table when you use the SHOW USER or SHOW SYSTEM commands.

Table 4–4: SHOW USER and SHOW SYSTEM Abbreviations

Abbreviation	Meaning	Description
Where		
DET	Detached	Job is detached from all terminals.
*	Dial-up	Job is running over a dial-up line.
Who		
,	–	Job is not logged in to the system.
OPR	Operator	Job runs under a system operator account.
SELF	Self	Job runs under your account.
Job status (state)		
RN	Run	Job is running or waiting to run.
RS	Residency	Job is waiting for residency. (The job has been swapped out of memory and is waiting to be swapped back in. Residency implies "memory resident.")
BF	Buffers	Job is waiting for buffers (no space is available for I/O buffers).
SL	Sleep	Job is sleeping (SLEEP statement).
SR	Send/Receive	Job is sleeping and is a message receiver.
FP	File Processor	Job is waiting for file processing by the system (opening or closing a file, file search).
TT	Terminal	Job is waiting to perform output to a terminal.
HB	Hibernating	Job is detached and waiting to perform I/O to or from a terminal. (Someone must attach to the job before it can resume execution.)
KB	Keyboard	Job is waiting for input from a terminal.
^C	CTRL/C	Job is at command level, awaiting a command. (In other words, the keyboard monitor has displayed its prompt and is waiting for input.)

(continued on next page)

Table 4–4: SHOW USER and SHOW SYSTEM Abbreviations (Cont.)

Abbreviation	Meaning	Description
CR	Card Reader	Job is waiting for input from a card reader.
MT, MM, or MS	Magnetic tape	Job is waiting for magnetic tape I/O.
LP	Line Printer	Job is waiting to perform line printer output.
DT or DD	DECtape	Job is waiting for DECtape I/O.
PP	Paper Tape Punch	Job is waiting to perform output on the high-speed paper tape punch.
PR	Paper Tape Reader	Job is waiting for input from the high-speed paper tape reader.
DK,DM,DB,DS,	Disk	Job is waiting to perform disk I/O. DP,DL,DF,DR
DX	Flexible Diskette	Job is waiting to perform flexible diskette I/O.
RJ	RJ2780	Job is waiting for RJ2780 I/O. (The RJ2780 is a device for communicating with an IBM system.)
??	–	Job's state cannot be determined by the SHOW command.
+ or –	Temporary Privileges	Job has temporary privilege. A plus sign (+) means that the temporary privilege is active. The minus sign (-) means that the program has temporary privilege, but that it is not active.
The following status descriptions may appear after one or more of the other job state abbreviations:		
Lck	Locked	Job is locked in memory for the current operation.
Nsw	No Swapping	A program has requested that the job not be swapped from memory.
Swi	Swapping In	Job is currently being swapped into memory.
Swo	Swapping Out	Job is currently being swapped out of memory.
Xnn	–	Job is swapped out and occupies slot nn in swap file X; file is denoted by A, B, C, or D to represent files 0 through 3 of the swapping structure. For example, A03 means slot 3 of swapping file 0.
Run-time System		
BASIC	BASIC	The BASIC–PLUS run-time system.
BASIC2	BASIC–PLUS–2	The BASIC–PLUS–2 run-time system.
DCL	DCL	The DIGITAL Command Language (DCL) keyboard monitor.
RT11	RT11	The RT11 run-time system.
RSX	RSX	The RSX run-time system.
...RSX	–	"Disappearing RSX," in use temporarily because you are running a program under the RSX run-time system.

When a privileged user types either SHOW USERS or SHOW SYSTEM, the display contains columns with additional information. For example, a privileged user who enters SHOW SYSTEM sees a display similar to:

```
$ SHOW SYSTEM[RET]
RSTS V8.0  *TARA*  status at 01-Dec-82, 05:23 PM   up 8:32:08

                                  (1)                        (2)
Job    Who      Where     What     Size    State  Run-Time   Pri/RB    RTS
 1     [OPR]    Det       ERRCPY   5/31K    SR       24.4      0/6     BAS4F
 2     [OPR]    Det       OPSRUN  15/31K    SL     2:47.0     -8/6     BAS4F
 .
 .
 .
14   [SELF]     KB27      SYSTAT  11/31K  RN LcK     1.6      -8/10    BAS4F
15     1,210    KB76*     ...MAC  20/31K  KB        26.2      -7/6     ...RSX

$
```

The display for privileged users contains the following information:

(1) The SIZE column includes a number following a slash, which indicates the size to which the job can expand. The size is shown in thousands of words — or K words — of computer storage space. (A *word* is a unit of storage in the computer that holds 16 bits, or 2 ASCII characters, of information.)

(2) A column headed Pri/RB prints a set of numbers that indicates the job's priority and run burst.

To understand what a *run burst* is, you need to know how RSTS/E divides its resources among users.

Each user's job gets a certain amount of time to use the computer all to itself, called a *time slice*. Since the computer can do a lot of work in just a few milliseconds, it seems to you at your terminal that you have the computer all to yourself. That is the central idea of the RSTS/E timesharing system: a lot of jobs can share the computer's resources at the same time.

A run burst is the length of time your job is allowed to run in terms of *time slices*, or uninterrupted computer time. Each time slice is comprised of *ticks*. A tick is about 1/60 of a second. You can compare a tick to a clock ticking away the seconds, only the computer divides time into much smaller increments. Therefore, a run burst is the maximum number of ticks the system gives you for your *slice*.

If your job is assigned a run burst of 6, that means that your job has the computer to itself for six ticks before another job gets a slice of the computer's time. Because timesharing happens so fast, you may not be able to tell that there are other jobs on the system.

Attached and Detached Jobs

Some of the command displays of system status involve attached and detached jobs. The following information describes both of these kinds of jobs:

1. An attached job is associated with a terminal, which is known as the job's console.

2. An attached job always communicates, or does I/O (input/output) with that terminal. The job may also do I/O with other terminals.
3. A detached job generally does not communicate (perform I/O) directly with terminals.
4. A job can run at a terminal and then become detached, either through its own actions (which requires privilege), another job's action, or because the job's console was a dial-in line that was hung up.
5. A detached job does not have a console. If it tries to perform I/O with its console, the job hibernates.

Restoring a Disconnected Dial-up Line

You can connect your terminal to a RSTS/E system by using a dial-up line. However, on rare occasions, the line can be disconnected because of electronic noise or other kinds of interference. The following example describes the actions you can take when this happens.

Assume you log in to a RSTS/E system by a dial-up line, and that the system identifies your session as Job 28. Then the line is hung up because of interference in the phone connection. Your job is now detached. The job attempts to do I/O to the console terminal, which is unsuccessful, so the job generally then hibernates.

Your next action should be to reestablish the connection by dialing up and logging back in. The dialogue at your terminal is:

```
Hello(RET)
User:  52,20(RET)
Password:(RET)

Job 28 is detached under this account.
Job number to attach to?
```

If you press the RETURN key, Job 28 stays detached (using up system resources), and you are logged in under another job number. If you instead type 28, RSTS/E puts you back to where you were in the terminal session before the line was disconnected:

```
Job number to attach to? 28(RET)
Attaching to Job 28
```

Sometimes the following message is displayed at your terminal:

```
?I/O to detached Keyboard
```

SHOW USERS

The SHOW USERS command displays information about the status of attached jobs on the system.

To use the command, type:

```
$ SHOW USERS(RET)
```

For example:

```
$ SHOW USERS(RET)

(1) RSTS V8.0  *TARA*  status at 01-Nov-82, 05:23 PM   up 8:19:25

(2)  (3)    (4)      (5)     (6)     (7)      (8)          (9)
 Job   Who     Where    What     Size    State    Run-Time     RTS
 14   [SELF]   KB27    SYSTAT    11K    RN LcK      1.6       BASIC
 15    1,210   KB76*   ...MAC    20K     KB        26.2       ...RSX
 20   160,45   POJ5    ...MAC    20K     RN      52:05.1      ...RSX

$
```

The display tells you the following:

(1) System status information – You are using RSTS/E Version 8.0 software. The node name is TARA. At the present day and time (01–Nov–82, 05:23 PM) the system has been operating for 8 hours, 19 minutes, and 25 seconds (up 8:19:25).

(2) Job — There are three attached jobs (14, 15, and 20) now in use on the system.

(3) Who — The jobs are in the following accounts: yours ([SELF]), [1,210], and [160,45].

(4) Where — The job's console terminals are KB27 and KB76. A pseudo keyboard (POJ5, which is pseudo keyboard zero) is controlled by job 5. The asterisk (in KB76*) signifies a dial-up line.

(5) What — The following operations are being performed: You are checking the system's status, which uses the SYSTAT program. The users of Job 15 and 20 are each running the MACRO assembler (...MAC, which refers to the program MAC.TSK).

(6) Size — The amounts of memory used by each job are 11K, 20K, and 20K words of storage, respectively. (Each word is 16 bits, or 2 bytes. The Size column is mainly to tell you the load each job places on the system.)

In computing, K approximates thousand-word units of memory storage (actually K = 1024 words). Therefore, 11K stands for roughly 11 thousand words.

(7) State — The status of each job is determined by the operation being performed. For example, your job's status is RN Lck. This means that the job is running or waiting to run (RN), and that it is locked in memory (Lck) during the SYSTAT operation. The SYSTAT operation is completed when the display appears at your terminal and is followed by the DCL prompt, which shows that your job is no longer locked in memory.

(8) Run-Time — The system keeps track of the CPU time each job uses. Your job, for example, has used a total of 1.6 seconds of CPU time since you logged in.

(9) RTS (Run-Time System) — RSTS/E provides a variety of run-time systems. The SYSTAT program uses the BASIC run-time system (BAS4F); the other two jobs are running programs that use *disappearing RSX* (...RSX).

Privileged users see additional information. Note the column labeled Pri/RB:

```
$ SHOW USERS(RET)

   RSTS V8.0  *TARA*  status at 01-Nov-82, 05:23 PM     8:19:25


   Job   Who     Where    What    Size     State    Run-Time Pri/RB  RTS
  14   [SELF]  KB27   SYSTAT   11K/31K  RN Lck      1.6      -8/6  BASIC
  15    1,210  KB76*  ...MAC   20K/31K   KB        26.2      -8/6  ...RSX
  20   160,60  POJ5   ...MAC   20K/31K   RN      52:05.1     -8/6  ...RSX

  $
```

The column labeled Pri/RB stands for "Priority/Run Burst." The numbers "-8/6" mean a priority of -8 and a run burst of 6 ticks.

SHOW SYSTEM

The SHOW SYSTEM command displays information about the status of all jobs, attached and detached, in use on the system.

An attached job is one that is associated with a terminal, and thus allows user input. By contrast, a detached job on RSTS/E is used to perform various system functions, such as printing files on the line printer.

For example, suppose your system manager invokes a program at system startup to keep track of jobs that run through any queue on the system. RSTS/E would consider that job to be detached because it is not associated with a specific terminal.

You may not recognize the names of programs that are run by detached jobs. They need not be standard RSTS/E programs — some could be created at your site. In addition, some of these programs are used primarily by your system manager. Programs for system management are described in the *RSTS/E System Manager's Guide.*

To use the SHOW SYSTEM command, type:

```
$ SHOW SYSTEM(RET)
```

The only difference between SHOW SYSTEM and SHOW USERS is that the SHOW SYSTEM command includes information about the status of detached jobs. (A detached job is identified by "Det" in the "Where" column of the display.) Refer back to the description of SHOW USERS for an explanation of the display.

The SHOW SYSTEM display looks similar to the following:

```
$ SHOW SYSTEM(RET)

RSTS V8.0  *TARA*  status at 01-Nov-82, 05:36 PM   up 8:32:08

Job   Who     Where    What    Size   State    Run-Time    RTS
 1    [OPR]   Det     ERRCPY    5K    SR         24.4      BAS4F
 2    [OPR]   Det     OPSRUN   15K    SL       2:47.0      BAS4F
 .
 .
 .
14   [SELF]   KB27    SYSTAT   11K    RN Lck     1.6       BAS4F
15    1,210   KB76*   ...MAC   20K    KB        26.2       ...RSX

$
```

The display for privileged users contains two additional columns:

```
$ SHOW SYSTEM[RET]

RSTS V8.0 *TARA* status at 01-Nov-82, 05:23 PM   up 8:32:08

                                  (1)                          (2)
Job   Who    Where   What    Size     State    Run-Time    Pri/RB   RTS
 1    [OPR]  Det     ERRCPY   5/31K    SR         24.4       0/6    BAS4F
 2    [OPR]  Det     OPSRUN  15/31K    SL       2:47.0      -8/6    BAS4F
 .
 .
 .
14   [SELF]  KB27    SYSTAT  11/31K    RN Lck     1.6       -8/10   BAS4F
15    1,210  KB76*   ...MAC  20/31K    KB        26.2       -7/6    ...RSX

$
```

The additional information is:

(1) The /31K in the Size column shows the maximum size of a job; 31K means roughly 31,000 words of storage space.

(2) The heading Pri/RB stands for "Priority/Run Burst." For example, the setting "-8/10" means that the priority assigned to the job is -8 (the default) and the run burst is 10 ticks.

Specifying a Terminal's Characteristics

You use the SHOW TERMINAL command to display information about your terminal and the SET TERMINAL command to change this information.

An example of a SHOW TERMINAL display is shown below. More detailed information about the command is included in the SHOW TERMINAL.

```
$ SHOW TERMINAL[RET]

Current settings for KB34:
        NoBroadcast          CRFill=0          Echo            HostSync
        LowerCase            NoParity          Scope           Speed=9600
        NoTab                TTSync            Width=80

$
```

These characteristics and others that can appear are described in Table 4–5.

Now, assume you want to change the NoBroadcast characteristic (which prevents messages from other users from appearing on your terminal) to Broadcast (which allows these messages). You use the SET TERMINAL command to change this characteristic:

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

The change you made with the SET TERMINAL command (replacing "NoBroadcast" with "Broadcast") is reflected in the display:

```
$ SHOW TERMINAL(RET)

Current settings for KB34:
        Broadcast           CRFill=0          Echo              HostSync
        LowerCase           NoParity          Scope             Speed=9600
        NoTab               TTSync            Width=80

$
```

Refer to the description of the SET TERMINAL command for complete information about changing characteristics of your terminal.

Table 4–5 contains a list of characteristics that can appear in a SHOW TERMINAL display. The characteristics that you can set with the SET TERMINAL command are indicated by a "Yes" in the "Settable?" column. A "No" in the "Settable?" column means that the characteristics are shown in the display but cannot be reset.

Many of the characteristics in Table 4–5 (such as CRFill=n, Parity, and HOSTSYNC) need to be set only once for the terminal, so you may never need to reset them. Usually your system manager sets these characteristics when the terminal is first connected to the system. By contrast, characteristics such as Broadcast and Width=n depend on your preference for using the terminal. The characteristics you can set are described in the description of SET TERMINAL.

There are no true "default" characteristics. That is, if you do not specify a characteristic, it is either implied by a terminal type (such as /VT100), or it remains constant when you change other settings. (In the command formats, many of the categories for "Defaults" are left blank because the defaults do not apply.) A characteristic remains the same unless you change it or specify a terminal type.

Table 4–5: SHOW TERMINAL Characteristics

Command	Settable?	Function
Broadcast	Yes	Allows messages from other users to be displayed at your terminal.
NoBroadcast	Yes	Suppresses messages from other users. You might specify NOBROADCAST if you do not want your session at the terminal to be interrupted by system messages.
CRFill=n	Yes	Sets the fill factor to n for this terminal, where n is between 0 and 6. Usually the fill factor is already set at your terminal, and it only needs to be set once. (Refer to the description of TTYSET in the *RSTS/E System User's Guide* if you need information on fill characters.)
Echo	Yes	Allows the terminal to display the characters you type. For example, when you type the DIRECTORY command and press RETURN, the characters in the command are displayed as you type them. The system displays its response at your terminal.

(continued on next page)

Table 4–5: SHOW TERMINAL Characteristics (Cont.)

Command	Settable?	Function
Noecho	Yes	Suppresses the display of characters you type at the terminal. The system still interprets your input, whether or not it is echoed at your terminal. For example, when you type the DIRECTORY command and press RETURN, the characters in the command are not displayed. However, the system's response is displayed at your terminal.
Hardcopy	Yes	Sets the terminal for hard-copy output. If you set a video terminal (such as a VT52 or VT100) to Hardcopy, the only change involves the display of deleted characters. Because hard-copy terminals are not designed to "erase" characters, they show deletions by redisplaying the character between backslashes (\).
Hostsync	Yes	Causes the terminal to recognize a CTRL/S from the system (which stops the input of text from your terminal) and CTRL/Q (which resumes input). This feature is possible because of special terminal hardware that allows the computer to instruct it to interrupt and resume transmission. HOSTSYNC complements TTSync, because HOSTSYNC governs the transmission of input (the CPU is the "host"), and TTSync governs output (the terminal is "TT"). VT100s require both HOSTSYNC and TTSync. VT52s require TTSync and NOHOSTSYNC.
NoHostSync	Yes	The terminal does not recognize CTRL/S and CTRL/Q.
Lowercase	Yes	The system allows both uppercase and lowercase characters as input from and output to the terminal.
NoLowerCase	Yes	The system interprets all input from and output to the terminal as uppercase characters.
Nolowercase Input	Yes	The system interprets all input from the terminal as uppercase characters.
Nolowercase Output	No	The system displays output to the terminal as uppercase characters.
Parity = Even	Yes	The system sends characters to the terminal with the parity bit set for even parity, but ignores the parity bit on characters received. (Refer to the description of TTYSET in the *RSTS/E System Manager's Guide* for more information about parity.)
Parity = Odd	Yes	The system sends characters to the terminal with the parity bit set for odd parity, but ignores the parity bit on characters received.
NoParity	Yes	The system ignores the parity bit on characters it receives and treats the parity bit on characters it transmits to the terminal as if it were a data bit.
Scope	Yes	The terminal is set as a video display device. (For example, VT52s and VT100s are video terminals.) When you set the Scope characteristic, TTSync and CRFill = 0 are assumed, although you can reset these characteristics.

(continued on next page)

Table 4–5: SHOW TERMINAL Characteristics (Cont.)

Command	Settable?	Function
Speed = n	Yes	Sets to n the rate at which the terminal's interface can receive or transmit characters. The value for n can be any legal speed for the interface, unless your system manager restricts the speeds you can set. See the *RSTS/E System Manager's Guide* for more information.
Speed = (i,o)	Yes	Sets the rate at which the terminal's interface can receive (i for input) or transmit (o for output) characters. (Only a privileged user can set speeds.) The values i and o can be any legal speeds for the interface, unless the system manager restricts the speeds you can set. See the *RSTS/E System Manager's Guide* for more information.
Speed 2741	No	The speed setting for an IBM 2741 type terminal.
Speed not settable	No	The speed cannot be set with the SET settable TERMINAL command. (No message is displayed if you try, but the speed you specify is ignored.)
Tab	Yes	When Tab is set, the system transmits tab characters to the terminal without translating them. Therefore, when the tab character is displayed, the terminal "sees" one tab character and moves the cursor to the next tab stop. (The actual cursor placement is determined by the terminal type.)
NoTab	Yes	When NoTab is set and a tab character is displayed, the system sends a number of blanks to the terminal instead of a tab character.
TTSync	Yes	The terminal stops output when you type CTRL/S and resumes it when you type CTRL/Q. In addition, the terminal hardware may transmit these characters to the system, in order that the terminal display can "catch up." VT52 and VT100–type terminals require TTSync.
NoTTSync	Yes	CTRL/S and CTRL/Q have no special meaning to the terminal.
Width = n	Yes	Sets the maximum width of the print line for the terminal to n, which can be between 1 and 254. If you type more characters on a line than the maximum (say, Width = 80 and you type 81 or more), the system automatically moves the cursor to the next line.

In addition to the commands described in Table 4–5, you can also set the characteristics for a specific kind of terminal. The description of SET TERMINAL lists the terminal types you can specify. The system assigns a set of predefined characteristics that apply to the terminal type. You can reset any of these characteristics. For example, suppose you type:

```
$ SET TERMINAL/VT100RET
```

The system then assumes that your terminal has a set of characteristics that correspond to a VT100. You can check these characteristics with the SHOW TERMINAL command, and change them to any others that you might want to set.

SHOW TERMINAL

The SHOW TERMINAL command displays the characteristics of your terminal. Users with privileged accounts can also display the characteristics of other terminals. Most of these characteristics can be changed with a corresponding option of the SET TERMINAL command.

Format:

SHOW TERMINAL [device-name[:]]

For example:

```
$ SHOW TERMINAL(RET)

Current settings for KB27:
        Broadcast       CRFill=0        Echo            Hostsync
        Lowercase       NoParity        Scope           Speed=9600
        Tab             TTSync          Width=80

$
```

Notice that the default is your own terminal (KB27 in this example). A privileged user can specify someone else's terminal, as in:

```
$ SHOW TERMINAL KB12(RET)

Current settings for KB12:
        Nobroadcast     CRFill=0        Echo            Hostsync
        Lowercase       NoParity        Noscope         Speed=2400
        Tab             TTSync          Width=80

$
```

Command Parameters

device-name[:]

Specifies a terminal for which you want to display the characteristics. Only a privileged user can specify the name of another terminal. If you are not privileged and you specify another terminal, the following message is displayed:

```
?Protection violation
```

If you do not specify a device-name, the default is your terminal.

SET TERMINAL

The SET TERMINAL command lets you specify the characteristics of your terminal. Privileged users can also set the characteristics of other terminals.

Format

SET TERMINAL [[=]device-name[:]]

Command Qualifiers

/[NO]BROADCAST
/CRFILL[= n]
/[NO]ECHO
/[NO]HARDCOPY
/[NO]HOSTSYNC
/LA34
/LA36
/LA38
/LA120
/[NO]LOWERCASE
/PARITY = EVEN
/PARITY = ODD
/NOPARITY
/[NO]SCOPE
/SPEED = n
/SPEED = (i,o)
/[NO]TAB
/[NO]TTSYNC
/[NO]UPPERCASE
/VT05
/VT52
/VT55
/VT100
/WIDTH = n

The SET TERMINAL command allows you to select characteristics for a particular application or to override characteristics that a system manager may have set.

SET TERMINAL is especially useful for dial-up lines. Your system manager usually establishes a set of characteristics for the terminals at your site. By contrast, a terminal that will be used with a dial-up line could be any kind of terminal. Therefore, unless you use SET TERMINAL to establish the correct characteristics, the computer could apply an inappropriate set.

You must include either an equal sign (=) or a space between the command SET TERMINAL and the device you specify. Also, the qualifiers beginning with /LA or /VT, which specify terminals, automatically define a set of characteristics. You can override these characteristics when you explicitly assign a qualifier. For example, if you set your terminal as a VT100, the system assumes the /SCOPE qualifier. If you then set the /NOSCOPE qualifier, you override the default. See Table 4–6 and the descriptions of each qualifier for specific information.

Table 4–6: Default Characteristics for Terminals

Terminal	LA36	LA120	VT05	VT5x	VT1xx	/VK100
Qualifier	**/LA36**	**/LA120**	**/VT05 /VT55**	**/VT52**	**VT100**	**/VK100**
BROADCAST	*	*	*	*	*	*
CRFILL	0	0	0	0	0	0
ECHO	yes	yes	yes	yes	yes	yes
HARDCOPY	yes	yes	no	no	no	no
HOSTSYNC	yes	yes	yes	yes	yes	yes
LOWERCASE	yes	yes	no	yes	yes	yes
PARITY	no	no	no	no	no	no
SCOPE	no	no	yes	yes	yes	yes
SPEED	*	*	*	*	*	*
TAB	no	no	no	yes	yes	yes
TTSYNC	yes	yes	yes	yes	yes	yes
WIDTH	132	132	72	80	80	84

* Indicates that the current setting is not changed by the qualifier.

For example, suppose you want to set your terminal as a VT100. (You can do this successfully only if your terminal is a VT100. The command on other terminals will succeed, but the terminal may not behave normally.) Type:

```
$ SET TERMINAL/VT100RET
```

The following defaults apply, as shown in Table 4–6:

- The qualifier is /VT100.
- The setting for /BROADCAST remains the same as before you specified /VT100.
- /CRFILL is set to 0.

- What you type is displayed on the screen because of the /ECHO qualifier.
- The /NOHARDCOPY qualifier is assumed with the /VT100 setting, because VT100s are not hard-copy terminals.
- /HOSTSYNC is assumed.
- The terminal displays uppercase and lowercase characters because /LOWERCASE is assumed.
- /NOPARITY is assumed.
- The speed settings are the same as before you specified /VT100.
- Tabs are interpreted as tab characters, and not as blanks, because /TAB is assumed. Therefore, when you press the TAB key, the system transmits a tab character to the terminal, which moves the cursor over one tab stop.
- The terminal responds when you press CTRL/S and CTRL/Q because /TTSYNC is assumed.
- The maximum width of a line is 80 characters, because /WIDTH=80 is assumed for /VT100s.

Command Parameters

device-name[:]
Specifies the name of the terminal whose characteristics you want to change. If you do not include a device-name, the default is your terminal.

Only a privileged user can specify another terminal. The SHOW USERS command displays the keyboard numbers of all jobs on the system.

Command Qualifiers

/BROADCAST
/NOBROADCAST
Controls whether the terminal can receive messages broadcast by other users. By default, a terminal receives any messages the system operator or another user sends.

Use /NOBROADCAST when you are using a terminal as a noninteractive terminal or when you do not want special output to be interrupted by messages.

/BROADCAST is the same characteristic as /NOGAG, which is described in the *RSTS/E System User's Guide.*

/CRFILL[=n]
Specifies whether the system must generate fill characters following a carriage return on the terminal. This qualifier prevents the system from sending out data before the terminal is ready to accept it.

The number n must be in the range of 0 through 6. This indicates the number of null fill characters required to ensure that the carriage return completes successfully before the next meaningful character is sent. The default is /CRFILL=0.

You may need to use this qualifier if you are using a non–DIGITAL terminal. Usually this characteristic is set by your system manager, and you should not need to change it.

/ECHO
/NOECHO

Controls whether the terminal displays what you type.

When /NOECHO is set, the terminal displays only data that the system or a program writes to it. It does not display user input.

/HARDCOPY
/NOHARDCOPY

Generally used to indicate a hard-copy terminal, as opposed to a video terminal. The /HARDCOPY qualifier also affects how the terminal interprets certain input keys.

/HARDCOPY establishes the terminal as a hard-copy device. Thus, the RUBOUT or DELETE key cannot accomplish backspace deletions. Instead, any characters being deleted are redisplayed between backslashes (\). /HARDCOPY is a synonym for /NOSCOPE.

HARDCOPY implies /NOTTSync, although you can override it by explicitly typing /TTSync.

/HOSTSYNC
/NOHOSTSYNC

When /HOSTSYNC is set, the terminal recognizes a CTRL/S from the system (which stops the input of text from your terminal) and CTRL/Q (which resumes input). This feature is possible because of special terminal hardware that allows the computer to instruct it to interrupt and resume transmission. /HOSTSYNC complements /TTSync, because /HOSTSYNC governs the transmission of input (the CPU is the "host"), and /TTSync governs output (the terminal is "TT").

VT100–type terminals require both /HOSTSYNC and /TTSync. VT52s require /TTSync and /NOHOSTSYNC.

When /NOHOSTSYNC is set, the terminal does not recognize CTRL/S and CTRL/Q.

/HOSTSYNC is the same characteristic as XON, which is described in the *RSTS/E System User's Guide.*

/LA34

Indicates that the terminal is an LA34 terminal. When you specify /LA34, the same characteristics as for LA36 terminals are set. (See Table 4–6.)

/LA36
Indicates that the terminal is an LA36 terminal. When you specify /LA36, the default characteristics for LA36 terminals are set. (See Table 4–6.)

/LA38
Indicates that the terminal is an LA38 terminal. When you specify /LA38, the same characteristics as for LA36 terminals are set. (See Table 4–6.)

/LA120
Indicates that the terminal is an LA120 terminal. When you specify /LA120, the default characteristics for LA120 terminals are set. (See Table 4–6.)

/LOWERCASE
/NOLOWERCASE
Indicates whether the terminal prints uppercase only or uppercase and lowercase characters. These qualifiers are most useful for terminals which allow both uppercase and lowercase characters.

If you specify /NOLOWERCASE, all alphabetic characters are translated to uppercase. This would be useful if you needed a long section of a document to be in all capital letters, such as a disclaimer, quote, or legal notice.

If you specify /LOWERCASE, lowercase characters are not converted to uppercase. However, uppercase characters remain uppercase.

/LOWERCASE is a synonym for /NOUPPERCASE.

/PARITY[=option]
/NOPARITY
Defines the parity for the terminal. *Parity* is a checking system that the terminal uses to be sure it is receiving characters properly.

Usually your system manager sets the parity for your terminal before you use it, so you should not have to reset it. If you need more information on parity, see the *RSTS/E System Manager's Guide.*

/SCOPE
/NOSCOPE
Indicates whether the terminal is a video terminal and, therefore, how it reacts to certain keys. Thus, when you press the DELETE key, the cursor moves one space to the left and any character displayed in that position is erased.

When you set /SCOPE, the /TTSYNC qualifier is assumed. When you set /NOSCOPE, the /NOTTSYNC qualifier is assumed. However, you can override either default by specifying /NOTTSYNC or /TTSYNC explicitly.

/SCOPE is a synonym for /NOHARDCOPY.

/SPEED=n
/SPEED=(i,o)
Specifies the rate at which the terminal receives and transmits data. (Only a privileged user can set speeds.)

The form /SPEED=n sets the input and output baud rates to the same speed. For example, you specify the transmit and receive speeds of a terminal to 9600 by typing:

```
$ SET TERMINAL/SPEED=9600RET
```

The form /SPEED=(i,o) allows you to specify different baud rates for input and output, in the format (i,o). You must include the parentheses; otherwise, RSTS/E displays an error message. To set the receive speed to 1200 and the transmit speed to 1800, for example, you type:

```
$ SET TERMINAL/SPEED=(1200,1800)
```

The valid values for input and output baud rates are as follows. (Values you can set depend on your terminal's interface; not all interfaces accept all speeds.)

50	300	2400
75	600	3600
110	1200	4800
134	1800	7200
150	2000	9600

The default transmission rates depend on both the installation and terminal type. (When you set speeds to connect the terminal to the system over a telephone line, the speeds you should set depend on the modem.) Not all terminals allow different input and output baud rates. Consult the appropriate hardware manual for details on a specific interface device.

/TAB
/NOTAB

Determines how tab characters are output to the terminal.

When /TAB is set, the system transmits tab characters to the terminal without translating them. Therefore, when the tab character is displayed, the terminal "sees" one tab character and is expected to move the cursor to the next tab stop. (The actual cursor placement is determined by the terminal type.)

When /NOTAB is set and a tab character is displayed, the system sends a number of blanks to the terminal instead of a tab character.

/TTSYNC
/NOTTSYNC

Controls whether the system responds to the CTRL/S and CTRL/Q combinations.

/TTSYNC causes the system to stop sending output when you press CTRL/S and resumes output when you press CTRL/Q. (VT52 and VT100–type terminals require /TTSYNC.) /NOTTSYNC causes the terminal to ignore CTRL/S and CTRL/Q.

You can also use the NOSCROLL key on VT100–type terminals for the effects of CTRL/S and CTRL/Q. Output stops the first time you press NOSCROLL and resumes the second time you press NOSCROLL.

/UPPERCASE
/NOUPPERCASE
Specifies whether the system translates all lowercase letters to uppercase. /UPPERCASE is synonymous with /NOLOWERCASE.

/VK100
Indicates a VK100 or GIGI terminal.

/VT05
Indicates that the terminal is a VT05 terminal. Use /VT05 to set the default terminal characteristics for VT05 terminals. (See the column labeled "VT05" in Table 4–6.)

/VT52
Indicates that the terminal is a VT52 terminal. Use /VT52 to set the default terminal characteristics for VT52 terminals. (See the column labeled "VT5x" in Table 4–6.)

/VT55
Indicates that the terminal is a VT55 terminal. Use /VT55 to set the default terminal characteristics for VT55 terminals. (See the column labeled "VT5x" in Table 4–6.)

/VT100
Indicates that the terminal is a VT100 terminal. Use /VT100 to set the default terminal characteristics for VT100, VT101, VT125, and other VT100–type terminals. (See the column labeled "VT100" in Table 4–6.)

/WIDTH=n
Specifies the maximum number of characters on each line that you type or that the system displays. The width, n, must be in the range of 1 through 254. You might want to set the width of the display of text to less than the maximum if, for example, you want a narrow column to appear in a report.

SET TERMINAL Error Messages

Table 4–7 lists the error messages you can get from the SET TERMINAL command.

Table 4–7: Error Messages for Terminal Characteristics

?Command error – <text>

The error denoted by <text> was encountered when the system was executing the command.

?Error – <text>

The error denoted by <text> was encountered when the system could not change the characteristic you specified.

?Illegal fill factor

The fill factor specified is not between 0 and 6.

?Illegal speed

The speed you tried to set is not allowed.

?Illegal width

The width specified is not between 1 and 254.

?Terminal open error – <text>

The error denoted by <text> was encountered while the system was executing the command.

%Warning – Cannot open $TTYSET.SPD file

The command could not access the terminal speed file and displays the specific error by printing the COMMAND ERROR message. Notify your system manager if you encounter this error.

%Warning – $TTYSET.SPD error

The system is warning you of possible corruption in the terminal speed file. The program then displays the specific error by printing the COMMAND ERROR message. Notify your system manager if you encounter this error.

Physical Device Names and Logical Names

This section explains what physical and logical device names are and what they mean to you as a DCL user. The ASSIGN and DEASSIGN command descriptions explain how you can relate logical names to physical devices, as well as to directories.

Physical Device Names

The system assigns physical device names to peripheral devices, such as:

1. Line printers (LP:)
2. Terminals (KB:, for "keyboard")
3. Magnetic tape drives (MM:, MT:, or MS:, depending on the kind of tape drive)
4. Disk drives (such as DB:, DM:, or DR:, depending on the kind of disk)

Use the SHOW DEVICES command for a list of the devices in use on your system. (A description of the command and a list of device names are included in Chapter 5.)

You can use physical device names in operations that allow you to specify a device. For example, you can display the characteristics of a specific terminal by including a physical device name, such as KB35:

```
$ SHOW TERMINAL KB35:RET
```

Similarly, you can determine which disk or magnetic tape you want to use in file transfer operations. For instance, assume you have a tape mounted on a tape drive named MT2:. To copy file RACE.ME in your directory to the tape on drive MT2:, type:

```
$ COPY RACE.ME MT2:RET
```

Another example of a physical device name is SY:, which refers to the public disk structure. You can therefore use SY: to refer to the public disk structure, regardless of the drive on which a disk is located. Such an assignment is useful if, for example, the drive on which the disk is usually mounted is put off line for maintenance, and the system disk is temporarily mounted on another drive. Users who reference accounts on the system disk do not need to adapt their commands or programs to fit hardware changes.

Physical device names are often used in logical name assignments. The following example assigns the logical name MARCH to DB2:, which is a disk drive:

```
$ ASSIGN DB2: MARCHRET
```

Logical names are described in the next section.

Logical Names

Logical names allow you to keep programs and batch control files independent of the physical locations of files. They also provide a covenient shorthand way to specify devices and directories that you refer to frequently.

You can assign a logical name to a physical device name or to a directory. The logical names that you assign are valid for your job only. For example, only your job can reference the logical name that you assign to a device. No other user can reference that logical name.

Advantages of Using Logical Names

Suppose you write a program that will require data from a magnetic tape. If your program specifies a physical name for the tape drive, such as MT2:, the success of the program depends on the availability of tape drive #2 at the time the program runs. However, you can assign a logical name to the tape drive before running the program. This ensures the system's recognition and access to that tape regardless of the drive on which it is mounted. Otherwise, you would have to change each occurrence of "MT2:" in your program to the physical name of another tape drive.

You can use logical names to increase the efficiency of a batch job, which does not require your presence at a terminal. (The batch facility is described in Chapter 6.) For instance, assume that you created a large batch job in the morning to be run by an operator at night, when there are fewer users on the system. If the batch control file accesses a disk, you can inform the operator of the logical name you assigned to the disk. Therefore, the operator need not be concerned about the availability of a specific disk drive. Any drive will do, since the system recognizes the logical name once the operator assigns it.

Logical names also make it easier to read and modify batch control files. (Batch control files contain commands that are used when you run a batch job.) For example, you can assign mnemonic names to any devices or files you include in the control file. The following command for a batch control file uses the ASSIGN command to assign the logical names BATCH1 to disk drive DB0:

```
$ASSIGN DB0: BATCH1
```

Logical Names and Devices

You can use logical device names to describe the nature of information on a disk, magnetic tape, or other medium. The logical name may be easier to remember than the physical name with its device number. You can choose a logical name without regard to what device drives may be available at some future time.

The logical names that you assign for devices do not depend on the physical device specifications. Unlike a physical name, a logical name is independent of the drive on which the medium is mounted. Logical device names make it easier to adapt a program for use on different drives. Thus, if you write a program referencing physical devices, you can give these devices logical names of your own choosing by issuing the ASSIGN command. This action associates your chosen names with the devices and makes the program independent of the devices' physical locations on the system.

System-Wide and User Logicals

When you assign logical names, you are the only one who can use them. Your logical name assignments are deleted when you log out. The logical names that you assign are called *user logical names*, to distinguish them from *system-wide logicals*.

There are some logical names that everyone on the system can use. Certain system-wide logicals are standard for most RSTS/E systems; others are assigned for your site by your system manager. These system-wide logicals are very much like user logicals, except that they remain assigned even when your system manager logs out and they are valid for all users.

Numbers of Logical Names

A job can have a maximum of four logical name assignments at a time if you specify only device-names. However, if one or more of the logical names is associated with a directory, the job is limited to three logical name assignments.

You can assign more than one logical name to a device. For example, you could assign to the tape on drive MM0: the logical names RACE1, RACE2, and RACE3, and use any of these three names as your application requires.

Physical File Specifications

A physical file specification describes the location of a file on the system. This information is useful in making logical name assignments.

The following example specifies file TSHIRT.PHU in account [3,24] on a disk in the DR3: disk drive:

```
_DR3:[3,24]TSHIRT.PHU
```

In the preceding physical file specification:

- The underscore (_) indicates a physical device (see the next section)
- DR3: is unit three of a specific kind of disk drive (DR:)
- [3,24] is a directory
- TSHIRT.PHU is the name and type of a file

Though there may be many ways to specify a given file, each file has one and only one "physical" file specification. In a physical file specification, none of the parts are omitted; nothing is allowed to default.

If the device is file-structured (as in DOS format magnetic tapes, which are discussed in Chapter 5), then the physical file specification must include an explicit PPN, a file name, a period, and a file type (which can be null).

How to Override Name Precedence

Sometimes you may have a logical name that conflicts with a physical name.

The system always recognizes and accepts a device's physical name, whether or not a logical name has been assigned to that device. Thus, you can specify a magnetic tape running on drive 2 and assigned that logical name STAR as either MT2: or as STAR:.

If you put an underscore before a physical name, the system will not try to interpret it as a logical name. A physical device name preceded by an underscore causes the system to access that device, regardless of any prior assignment.

RSTS/E evaluates the parts of a file specification in the following sequence. When a file specification contains a name followed by a colon, RSTS/E first determines whether the name is a user logical. If it is, then the system replaces the name with its expansion. (An *expansion* is the device name or PPN to which the logical name was assigned.) Otherwise, it checks to see if the name is a system-wide logical.

If the name is not a system-wide logical, RSTS/E then determines whether the name is one of the physical device names that your RSTS/E system software is equipped for. (Your system manager selects which devices to equip the system for at the time of system generation.)

Finally, if the device name is not a physical device name, the system displays the message:

```
"?Not a valid device"
```

ASSIGN

The ASSIGN command relates a logical name to a directory or a physical device. The names you assign stay in effect until you log out, or until you deassign the name. (Underscores show brackets that you must type.)

Format

ASSIGN device-name:[[ppn]] logical-name[:]

Prompts

Device: device-name:[[ppn]]

Logical name: logical-name

A device-name must end with a colon. Although a colon is optional when you assign logical names, it is required when you use it in a command string. When a logical name is used with a file specification, the colon separates the logical name from the rest of the file specification.

When you use the ASSIGN command, one of the following can occur:

- The command works as expected, and no message is displayed.
- The assignment statement is incorrect, and the following message is displayed:

```
?Not a valid device
```

- You exceed the maximum number of logical name assignments you can make, and the following message is printed:

```
?Too many logical names assigned
```

Command Parameters

device-name:[[ppn]]
Specifies the name of the device or PPN to be assigned a logical name. Brackets around the PPN are required, as in the following example:

```
$ ASSIGN MT1:[2,214] GRUNT:RET
```

logical-name[:]
Specifies a one- to six-character logical name to be associated with the device.

The ASSIGN command equates a logical name to a part of a physical file specification. The following example equates the logical name EXCER: to the directory DB2:[3,24]:

```
$ ASSIGN DB2:[3,24] EXCER:RET
```

The string "DB2:[3,24]" is called the "expansion" of the logical name EXCER.

Subsequently, you can refer to the directory by its logical name when you issue a DCL command. For example:

```
$ DIRECTORY EXCER:RET
```

When the system executes this DIRECTORY command, it replaces the logical name EXCER: with its expansion, DB2:[3,24], and lists all of the files in that directory on your terminal.

Another example is:

```
$ TYPE EXCER:RUNNER.DATRET
```

When the system executes this command, it replaces the logical name EXCER: with its expansion and displays the contents of the file DB2:[3,24]RUNNER.DAT on your terminal.

You can equate a logical name to a physical device name, to a directory, or to both. For example:

```
$ ASSIGN DB2: EXCER                  (DB2: is a physical device)
$ ASSIGN [3,24] EXCER:               ([3,24] is a directory)
$ ASSIGN DB2:[3,24] EXCER:           ([3,24] is a directory on DB2:)
```

When you assign a logical name, the partial file specification that you assign to the logical name may itself include an already assigned logical name. When the system executes the ASSIGN command, it replaces the already assigned logical name with its expansion. Therefore, if you later change the assignment of the first logical name, the expansion of the second logical name is not affected. For example, consider the following sequence of commands:

```
$ ASSIGN DB2: WORK:
$ ASSIGN WORK:[3,24] EXCER:
$ DEASSIGN WORK:
$ TYPE EXCER:RUNNER.DAT
```

In this example, the logical name EXCER: corresponds to DB2:[3,24]. The TYPE command displays DB2:[3,24]RUNNER.DAT even though the logical name WORK: has been deassigned.

You can also assign logical names to devices when you issue the ALLOCATE or MOUNT commands, which are described in Chapter 5.

DEASSIGN

The DEASSIGN command cancels logical name assignments you made with the ASSIGN or ALLOCATE commands.

Format

DEASSIGN [logical-name[:]]

Command Qualifiers	**Defaults**
/ALL	None

Prompts

Logical Name: logical-name[:]

Command Parameters

logical-name[:]

Specifies a one- to six-character logical name to be deassigned.

A logical name is required unless you specify /ALL. If you specify neither /ALL nor a logical name, you get an error message:

```
$ DEASSIGN(RET)
Logical Name:(RET)
?Missing parameter

$
```

Command Qualifiers

/ALL

Deletes all the logical names that you assigned. For example:

```
$ DEASSIGN/ALL(RET)
```

If you specify /ALL, you cannot also specify a logical name.

Working with Devices 5

All RSTS/E users share peripheral devices. These devices are referred to by either physical or logical device names. Because the number of devices on any system is limited, you need to know about other people's jobs when you reserve one of the devices for your own use.

This chapter describes the commands summarized in Table 5–1.

Table 5–1: Commands for Using Devices

Command	Description
SHOW DEVICES	Displays devices that are in use on your system. (See page 5–18.)
ALLOCATE	Reserves a device so that only you can use it. (See page 5–8.)
DEALLOCATE	Releases a device from your exclusive use, so that other users may access the device. (See page 5–9.)
MOUNT	Logically loads a magnetic tape or disk onto a device. (See page 5–10.)
INITIALIZE	Clears ("zeros") a tape for use. (See page 5–13.)
DISMOUNT	Logically unloads a magnetic tape or disk from a device. (See page 5–15.)
REQUEST	Sends a message to the system's operator. (See page 5–17.)

Physical Device Names

Each of the devices on a RSTS/E system has its own *device name.* A *physical device name* is also referred to as a device designator or device specification because you use it to specify a device.

A physical device name consists of two alphabetic characters, optionally followed by a unit number, and always terminated by a colon (:). The alphabetic characters are generally an abbreviation of the device's generic name (DT for DECtape, DK for disk, MT for magnetic tape, and so forth). The unit number uniquely identifies the device itself or the drive on which it is mounted.

For example, "KB0:", is a "keyboard," or terminal, and "0" is the *unit number.* Other terminals on your system have similar device names assigned to them, such as "KB7:" or "KB35:".

If you do not specify a physical device name, the system assumes the public structure. For non-file-structured devices (paper tape, line printer, and so forth), only the physical device name need be specified; the system ignores any file name, type, or directory that you specify.

Table 5–2 lists the RSTS/E physical device names.

Table 5–2: RSTS/E Physical Device Names

Designator	Device
DF:,DS:,DK:,DL:,DM:, DP:,DR:,DB:,DU:, or SY:	RSTS/E public disk structure
SY0:	System disk
DF0:	RF11 disk (all platters)
DS0: to DS7:	RS03/RS04 fixed head disk units 0 through 7
DK0: to DK7:	RK05 disk cartridge units 0 through 7
DL0: to DL3:	RL01/RL02 disk cartridge units 0 through 3
DM0: to DM7:	RK06/RK07 disk cartridge units 0 through 7
DP0: to DP7:	RP02/RP03 disk pack units 0 through 7
DR0: to DR7:	RM02/RM03/RM05/RM80 disk units 0 through 7
DB0: to DB7:	RP04/RP05/RP06 disk pack units 0 through 7
DU0: to DU7:	RA80/RA81/RA60/RC25/RD51/RX50 units 0 through 7
PP:	High speed paper tape reader and punch
CR:	CR11 punched or CM11 mark sense card reader
CD:	CD11 punched card reader
MT0: to MT7:	TE10/TU10/TS03 magnetic tape units 0 through 7
MM0: to MM7:	TE16/TU16/TU45/TU77 magnetic tape units 0 through 7
MS0: to MS3:	TS11/TU80/TSV05 magnetic tape units 0 through 3
LP0: to LP7:	Line printer units 0 through 7
DT0: to DT7:	TU56 DECtape units 0 through 7
DD0: to DD7:	TU58 DECtape II units 0 through 7
KB:	Your terminal
TT: or TI:	Your terminal (synonyms for KB:, the terminal that initiated the job)
KBn:	Terminal n on the system
TTn:	Terminal n on the system (synonym for KBn:)
NL:	The null device
PKn:	Pseudo keyboard n
DX0: to DX7:	RX01/RX02 flexible diskette (floppy disk) units 0 through 7
DU:	RA60, RA80, RA81, RD51, and RX50

Note

You can reference LPn:, DTn:, DXn:, KBn:, MMn:, MTn:, MSn:, and DDn: where n is between 0 and the maximum number of such units on the system. LP:, DT:, DX:, MM:, MT:, MS:, and DD: are each the same as specifying unit 0 of the related device.

If your system has only TS11, TU80, or TSV05 tape units, the designators MS: and MT: are synonymous. If your system has only TE16, TU16, TU45, or TU77 tape units, the designators MM: and MT: are synonymous. On systems that have the DC11 card reader, the designator CR: is synonymous with CD:.

If you do not specify a unit number, the defaults are:

Disk – Public structure
Tape – Unit 0
Keyboard – Yours
Line printer – Unit 0
DECtape – Unit 0

If you specify a device or type of device that is not part of your system's configuration, the following message is displayed:

```
"?Not a valid device"
```

An underscore before a name suppresses any interpretation of the name as a logical name. For example, if you assign the logical name MT0: to a disk, you can specify _MT0: when you want to refer to magnetic tape unit 0.

Devices You Can Reserve

Sometimes you may need to reserve, or *allocate*, a device for your exclusive use. Only one job at a time can use an allocated device.

Disks, public and private, generally cannot be reserved. That is, you cannot request one for your exclusive, temporary use. Even a private disk is usually shared by a number of users. To satisfy the frequent need for input and output media devoted to your work alone, RSTS/E provides an assortment of allocatable devices. You can reserve DECtapes and magnetic tapes, for example, for your own input or output. Card punches and paper tape punches can be reserved for your output.

You can only allocate devices that are not already in use. If a tape drive or punch is being used, you cannot immediately allocate it to yourself. Should you try, the system displays the error message "?Device not available".

Assigning and Allocating Devices

Sometimes the distinction between the ASSIGN and ALLOCATE commands is confusing, because the command names are not consistent among all RSTS/E keyboard monitors.

DCL is different from the other keyboard monitors on RSTS/E. In the other keyboard monitors, the ASSIGN command performs functions that both ALLOCATE and ASSIGN perform in DCL. This section explains the differences between the two commands in DCL.

The ALLOCATE command allocates a device to you. Also, ALLOCATE can assign a logical name for the device you are using.

Whereas ALLOCATE can assign logical names and allocate devices for your use only, the ASSIGN command only assigns logical names. In either case, you get an error message if you try to exceed the number of logical names you can assign.

The DEALLOCATE command releases a device from your exclusive use, but only the DEASSIGN command cancels the logical name assignment.

Device Independence

Device independence is a central idea in RSTS/E. This means that you can use a file on one device much as you can use a file on any other device. This idea applies both to the system user and to the programmer.

As much as possible, the system tries to hide the differences between devices from both the system user and the programmer. For example, you can run a program that is on a tape just as you run a program that is on a disk.

However, different devices offer different capabilities. RSTS/E cannot entirely hide these differences. The following sections discuss some of the differences among devices.

Public and Private Disks

A disk is either public or private, as determined by the system manager. Any user who has an account on the system can create files on a public disk. But if a disk is private, then you cannot place files there unless the system manager (or other privileged user) creates a directory for you on that disk.

Every RSTS/E installation has at least one public disk, which is known as the *system disk*. For most purposes, the public disks on a given system are treated as a unit. This unit is called the *public disk structure*, and has the device name SY: (which stands for system).

When you create a file and do not specify what device to create it on, or when you specify SY:, the system puts the file on one of the public disks. The system also decides which public disk to put it on if there is more than one public disk. When you specify an existing file and do not specify a device (or when you specify SY:), the system searches each of the public disks for the file.

You cannot have two files of the same name and type on two different public disks, just as you cannot have two files of the same name and type on a single disk. If you try to give a file the same specification as an existing file, one of two things can happen. Either you get the message "?Name or account now exists", or the existing file is replaced, depending on the command.

The Public Disk Structure

Users share computing time on the RSTS/E system. Each user is allocated a "slice" of the processor's time. Users may share another system resource as well: the array of devices. One set of devices (disks) is shared by a number of users. This shared set of devices is called the *public disk structure*, because it is always accessible to all users and because the system treats it as a unit. On some systems, it may include many separate disk packs. One of these, the system disk, contains the system code, language processors, and, possibly, the library of system programs. The other disk packs (the public disks) contain information created by the users. (The system disk also may contain user programs and data, which are stored in files.)

When you are logged in to the system and working with a disk file, you are usually not concerned about which disk in the public structure happens to contain that file. The particular disk is chosen by the system according to current time-sharing needs. Each of the disks contains a master list of users' accounts, and, for each account, a list of all files stored under that account. The system, by using these lists, is able to locate your file when you request it from your terminal.

Private Disks

A good deal of system file activity — such as creation, access, editing, and deletion — takes place on the public disk structure. Because it is the largest constantly available medium of file storage, it is generally the busiest. But not all the disks on the system need be in the public structure. Some of them may be private disks; disk packs or cartridges that belong to a single user account or perhaps to a few user accounts, in the sense that these accounts alone are on the disk. Only if a private disk already contains a directory can files be created in that directory on the disk. Without one of these private disk directories, you can read or edit a private disk file, but only if its protection code permits.

From your point of view, then, one important difference between a private disk and one on the public structure is that you can always create a file on the public structure, whereas you can do so on a private disk only if you have a directory there. On both types of disks, file protection codes govern your read and write access to existing files. Another difference is that a private disk can be mounted or dismounted, whereas a public disk must be mounted during timesharing. This difference is of special concern to the owner(s) of the private disk and to the system manager, who determines which disks on the system are public and which are private.

Magnetic Tapes and Files

Magnetic tape is a compact, relatively inexpensive medium that can provide large amounts of offline storage. One reel of magnetic tape can store many files.

Unlike disks, magnetic tape allows only sequential access to files.

To get files from a tape, you should:

- Allocate the drive. (This step is optional, but it provides security.)
- Physically mount the tape.

- Issue the MOUNT command. (This step is optional, if the tape has default density and format.) The MOUNT command also allocates the drive, if it was not already allocated to you.
- Copy files from the tape using the COPY command. (With the COPY command, you can copy a file that requires multiple tapes. User programs can access tape files if they are stored on only one tape, but system programs can access files stored on more than one tape.)
- Issue the DISMOUNT command. (This step is optional, but it deallocates the drive and rewinds the tape.)
- Physically dismount the tape.

Protection of Files on Tapes

Protection of files on tapes works differently from protection of files on disks. While you are using a tape, the device is said to be allocated to your job. While the device is allocated to you, you have complete control over the tape. You can read any file on it. You can create files with PPNs other than your own if the tape is in DOS format. No other user can access the tape while it is allocated to your job. The MOUNT command and the ALLOCATE command both allocate the tape device to your job. The DISMOUNT, DEALLOCATE, and LOGOUT commands free the tape device so that other jobs can allocate it.

Tape Density

When you initialize a tape on a device, you decide what the tape's density will be. The two considerations are (1) the amount of information to be stored on the tape, and (2) the degree of portability between systems.

For example, suppose you have a choice of specifying a density of 800 or 1600 bits per inch (bpi). You can get twice as much information on a tape with a density of 1600 bpi. However, although most tape drives support a density of 800 bpi, not all tape drives support 1600 bpi.

ANSI and DOS Format

RSTS/E has two file structures for tapes. One structure is called ANSI format, because it complies with the standards of the American National Standards Institute. The other structure is called DOS format, because RSTS/E adopted it from the Disk Operating System format.

The ANSI format is an industry standard. The advantages to using ANSI format tapes are:

1. Portability. If you expect to move the contents of a tape from one system to another, it is very possible that the other systems will recognize ANSI format.
2. Multiple tapes. The files you copy can be stored on more than one reel of tape.

Note

Where ANSI is used in RSTS/E documentation, it refers to the RSTS/E implementation of American National Standard X3.27–1978 – magnetic tape labels and file structure for information exchange. RSTS/E implements a subset of this standard. In addition, RSTS/E uses U (undefined) record format, which is not defined in ANSI standard X3.27–1978.

The advantage of using DOS tape is that files on DOS tapes are associated with PPNs. DOS tape files are also better for storing binary files that have no attributes, such as .OBJ (object) or .BAC (compiled BASIC–PLUS) files.

When you initialize a tape, you can select either ANSI or DOS format. Subsequently, when you mount the tape, you must again tell the system whether the tape is in ANSI or DOS format. If you do not specify a format, the system uses the default determined by your system manager.

There are two ways to tell if a tape containing data is written in DOS or ANSI format:

1. Look for a label on the tape. The person who put data on the tape may have placed a label on the tape and written "DOS" or "ANSI" on it.
2. Try mounting the tape as either DOS or ANSI. When you mount a tape on a drive, RSTS/E checks the format. If you entered the right format, you can proceed with tape operations. If the format you entered was incorrect, RSTS/E displays the error message "?Bad directory for device", in which case you can try again using the other format.

If the tape is in DOS format, then each file on the tape has a PPN. The PPN of a tape is like the directory on a disk file. For example, a DOS format tape on drive MT0: might contain a file whose specification is:

```
MT0:[1,2]LARD.RUN
```

Another file on the tape might have the specification:

```
MT0:[3,49]GOOD.RUN
```

The default PPN is yours, as with disk files, but note that PPNs on tape files have nothing to do with protection.

If the tape is in ANSI format, then files on the tape do not have PPNs. They have only names and types.

ALLOCATE

The ALLOCATE command reserves a physical device for your use during the current session and optionally establishes a logical name for the device. Once a device has been allocated, other users cannot access the device until you specifically deallocate it or log out. You can allocate a device only when it is not allocated by another job.

Format

ALLOCATE device-name[:] [logical-name[:]]

Prompts

Device: device-name

Command Parameters

device-name[:]
A logical or physical name.

logical-name[:]
Specifies a one- to six-character logical-name to be associated with the device. RSTS/E automatically translates any references you make to the logical-name to a physical device associated with the name. The logical-name parameter is optional.

DEALLOCATE

The DEALLOCATE command releases a device that you reserved for private use, so that other users may have access to it. (However, DEALLOCATE does not deassign any logical name you may have set up for the device; for that you need the DEASSIGN command.)

Format

DEALLOCATE [device-name[:]]

Command Qualifiers	**Defaults**
/ALL	None

Prompts

Device: device-name[:]

Command Parameters

device-name[:]

Specifies the name of the device to be deallocated. The device-name can be a physical device-name or a logical name. The device-name parameter is required unless you specify the /ALL qualifier.

Command Qualifiers

/ALL

Requests that all devices you have currently allocated be deallocated.

If you specify /ALL, you cannot specify a device-name.

MOUNT

The MOUNT command prepares a tape or disk for processing by system commands or user programs. (On some systems, mounting a disk requires privilege.) Issue the MOUNT command after you physically mount the tape or disk and put the drive on line.

Note

The MOUNT command has expanded capabilities for privileged users. See the *RSTS/E System Manager's Guide* for more information.

If the MOUNT command fails with the message, "?Disk pack needs REBUILDing", but you are not privileged, have your system manager mount the disk.

Format

MOUNT device-name[:] label

Command Qualifiers	**Defaults**
/[NO]WRITE	See description

Qualifiers for Disks	**Defaults**
/PRIVATE	/PRIVATE
/[NO]SHARE	/SHARE

Qualifiers for Tapes

/DENSITY=nnn
/FORMAT=ANSI
/FORMAT=DOS
/FORMAT=FOREIGN

Prompts

Device: device-name[:]

Label: label (for ANSI tapes only)

Pack-id: pack-id (for disks only)

The MOUNT command does the following, depending on whether you are mounting a tape or disk:

- Specifies a tape's density and format
- Checks that a tape or disk's device specification is correct
- Checks that a tape or disk has been initialized
- Verifies that a tape or disk drive has not been allocated to another user
- Allocates a tape drive (makes it accessible for your use)
- Verifies that a disk or tape is physically loaded on the device specified
- Verifies that the pack-id on a disk or the label on a tape (except for DOS tape) matches the label specified
- Allows you to mount a private pack for your own use

Command Parameters

device-name[:]
Specifies the physical or logical name of the drive on which a device is to be logically mounted.

label
pack-id
Specifies a label or pack-id, that identifies the tape or disk to be mounted. They can have from one- to six-alphanumeric characters.

The label is required when you mount a disk or a tape in ANSI format. It is not required (and is ignored) when you mount a tape in DOS or FOREIGN format.

Command Qualifiers

/DENSITY=nnn
Specifies the density in bits per inch (bpi) at which the tape will be read or written. This qualifier is valid only with tapes.

You can specify 800 or 1600 bpi, depending on the density supported by the tape drive. The default density depends on your installation.

/FORMAT=ANSI
/FORMAT=DOS
/FORMAT=FOREIGN
Indicates whether the tape is in a standard format used by the RSTS/E operating system. This qualifier is valid only with tapes.

ANSI and DOS formats are described earlier in this chapter, in the section named "ANSI and DOS Formats." A tape is FOREIGN if it is not in ANSI or DOS format.

If you mount a tape with /FORMAT=FOREIGN, the program you use to read the tape must be able to process any labels on the tape.

/PRIVATE
Allows you to mount a private disk, accessible only to those users who have accounts on the disk. (Your system manager designates whether disks are public or private.) /PRIVATE is the default if you are nonprivileged.

/SHARE
/NOSHARE
Determines whether the private disk you use is shared. /NOSHARE allows you to mount a private disk that is accessible only to the job that mounts it. /SHARE allows all users who have accounts on the disk to access it. /PRIVATE and /SHARE have the same meaning. If you specify /NOSHARE, you cannot specify /PRIVATE.

/SHARE is the default.

/WRITE
/NOWRITE
Controls whether you can write data to a tape or disk. The /WRITE qualifier allows you to modify the data contained on a tape or disk. If /NOWRITE is specified, you cannot modify the data.

You can specify /NOWRITE to provide read-only access to protect files. For a disk, this is equivalent to write-protecting it. For a tape, the drive itself must be write-protected; /NOWRITE simply generates an error message if the device is not write-protected.

/WRITE is the default for tapes when the device is not write-protected. /WRITE is also the default for disks (initialized as read/write) when the disk drive is not write-protected. However, /NOWRITE is the default if the drive is write-protected. Therefore, if you receive the warning, "%Device write protected", you have logically mounted your disk or tape with read-only access. /NOWRITE is also the default for disks initialized as read-only.

INITIALIZE

The INITIALIZE command deletes any data on a tape and writes a new label. INITIALIZE allocates the tape drive if it is not already allocated.

Use INITIALIZE to prepare a new tape or a tape that contains no useful files. Use MOUNT to prepare an already initialized tape that contains files you want to keep.

Note

See the *RSTS/E System Manager's Guide* for information on using INITIALIZE with disks.

Format

INITIALIZE device-name[:] [label]

Qualifiers

/FORMAT=ANSI
/FORMAT=DOS
/DENSITY=nnn

Prompts

Device:	device-name[:]
Label:	label
Proceed (Y or N):	

Note

The label prompt appears only for a tape in ANSI format or if /FORMAT=ANSI is specified. The proceed prompt asks you to confirm whether you want to initialize the tape. Type yes or no (Y or N) in response to the prompt.

Command Parameters

device-name[:]
Specifies the name of the drive on which the tape is physically mounted.

label
Specifies the identification to be encoded on the tape. You can specify a maximum of six alphanumeric characters. For an ANSI–format tape, the label is required. For a DOS–format tape, the label is ignored, because DOS tapes do not allow labels.

Qualifiers

/DENSITY = nnn
Specifies the density in bpi at which the tape is to be written. You can specify a density of either 800 or 1600 if the tape drive supports it.

If you do not specify a density, the system uses the default that your system manager has set.

/FORMAT = ANSI
/FORMAT = DOS
Specifies the tape format. If you do not specify a format, DCL uses the default determined by your system manager.

DISMOUNT

The DISMOUNT command releases a disk or tape previously accessed with a MOUNT command. You issue this command before you take the drive off line, or before you physically dismount the tape or disk.

Note

The DISMOUNT command has expanded capabilities for privileged users. See the *RSTS/E System Manager's Guide* for more information.

The DISMOUNT command deallocates the device if it was allocated to you. (On some systems, dismounting a disk requires privileges.) You cannot DISMOUNT a device if there are open files on it. If you try, RSTS/E displays the message:

```
?Account or device in use
```

Format

DISMOUNT device-name[:] [label]

Command Qualifiers (for tapes only)	**Defaults**
/[NO]UNLOAD	/UNLOAD

Prompts

Device: device-name[:]
Pack-id: pack-id label (for disks only, when you are nonprivileged)

Command Parameters

device-name[:]
Specifies the name of the device to be dismounted. If you omit a unit number, the default depends on the kind of device. (See the discussion on page 5–3.)

label
Specifies the identification label of the device. If you are nonprivileged, you must specify a disk's pack-id label to dismount it. For tapes, the label is optional (and is ignored).

Command Qualifiers

/UNLOAD
/NOUNLOAD
Rewinds and unloads the tape. This protects the tape against unauthorized access by other users.

/UNLOAD is the default.

REQUEST

The REQUEST command displays a message at a system operator's terminal.

Format

REQUEST ["]message["]

When you use the REQUEST command to send a message to an operator, the message is displayed at the operator services console.

Command Parameters

message

Specifies the text of a message to be displayed at the operator services console. For example:

```
$ REQUEST Do we need more paper for the line printer?RET
```

The message text can have a maximum of 128 characters.

SHOW DEVICES

The SHOW DEVICES command displays the status of devices that currently have disks mounted on them or that are allocated to jobs.

To use the command, type:

```
$ SHOW DEVICES
```

RSTS/E prints a message similar to the following:

```
Busy Devices:
Device   Job     Why
PK0      9       Open
PK1      22      Open
PK2      6       AS+Open
DMC-0    TRN     AS+Open
DMC-1    TRN     Open
DMR-2    TRN     Open
DMC-4    TRN     Open
DMC-5    TRN     AS+Open
Disk Structure:
Dsk Open    Size       Free    Clu  Err  Name    Level  Comments
DR1   70  131648   26556  20%   4    0  ARK      1.1   Pub, DLW
DR3   31  500352   42720   8%   8    0  W        1.1   Pri, DLW
DR4    0  242572   17796   7%   4    0  M        0.0   Pri, DLW
DR5    0  500352   76152  15%   8    0  H        0.0   Pri, R-O, DLW

$
```

Table 5–3 contains the abbreviations that can be included in a SHOW DEVICES display.

Table 5–3: Abbreviations in a SHOW DEVICES Display

Abbreviation	Meaning	Explanation
Busy device status		
AS	Assigned	Device is explicitly allocated to a job.
OPEN	Open	Device is open on a channel.
DOS	DOS	Magnetic tape is assigned with DOS format.
ANSI	ANSI	Magnetic tape is assigned with ANSI format.
NSP	Network Services Protocol	The monitor is using the device for DECnet communications.
Disk status		
Pub	Public	Disk is public.
Pri	Private	Disk is private.
NFS	Non-file-structured	Disk is open as a non-file-structured device.
R–O	Read-only	Disk unit is read-only (write-protected).
DLW	Date of Last Write	Date of last write (modify), rather than date of last access, is stored in file accounting entries.
Lck	Locked	Disk is in a locked state. (This means that nonprivileged users cannot gain access to the disk.)
NFF	New Files First	New files on the disk are put at the beginning of the directory.
Job n	Job n	Private disk is allocated to job n (reported with PRI and NFS).
Dirty	Dirty	Disk needs rebuilding (reported with PRI, LCK, and R–O).

Batch Processing 6

This chapter describes the use of the batch facility in DCL. For information about the non-DCL use of batch, see the *RSTS/E System User's Guide.*

A batch job consists of the execution of commands in a control file. You control the execution of the job with system commands.

The commands designed specifically for use in batch control files, as well as system commands for running a batch job, are listed in Table 6–1.

Table 6–1: Batch Processing Commands

Batch Commands for Control Files	
$JOB/DCL	Identifies the beginning of a batch job. (See page 6–11.)
$DATA	Signals the start of data in a batch job. (See page 6–13.)
$EOD	Signals the end of data in a batch job. (See page 6–13.)
$EOJ	Identifies the end of a batch job. (See page 6–13.)
$MESSAGE	Sends a message to the operator from a batch control file. (See page 6–14.)
Submitting and Controlling Batch Jobs	
SUBMIT	Enters a job into the batch queue. (See page 6–17.)
SHOW QUEUE/BATCH	Displays information about the batch queue. (See page 6–21.)
SET QUEUE/JOB/BATCH	Uses a job name to change the qualifiers (/AFTER, /PRIORITY, /HOLD, and /RELEASE) of a previously submitted batch job. (See page 6–23.)
SET QUEUE/ENTRY/BATCH	Same as SET/QUEUE/JOB, except that you refer to the job by number instead of by name. (See page 6–26.)
DELETE/JOB/BATCH	Removes a request to the batch queue by name. (See page 6–27.)
DELETE/ENTRY/BATCH	Same as DELETE/JOB, except that you refer to the job by number instead of by name. (See page 6–28.)

What Batch Is

Batch is a facility that allows you to process work without your having to be there. For example, you can submit a batch job to compute a list of figures while you use your terminal for other operations. Batch is useful when you do not want to tie up a terminal with a noninteractive job, such as running a payroll or transferring files onto a tape.

You run a batch job by first putting batch commands into a *control file* that you create. Then you use the SUBMIT command to enter the control file as a batch job.

When you submit the job, the batch processor executes the series of commands in the control file, in a sequence known as a *batch stream*. You can then check and modify the status of the job with DCL system commands. When the job is completed, you can verify that the job ran as expected by examining the *log file* that the batch facility creates for each job.

The batch job you submit is placed in a batch queue, which the system maintains. Several DCL commands (such as SHOW QUEUE and DELETE/ENTRY) can operate on either a print queue or a batch queue, depending on the qualifiers you specify. For example, SHOW QUEUE displays the status of jobs in a queue. You specify a batch queue by including either the /BATCH qualifier (as in SHOW QUEUE /BATCH) or a batch queue name (as in SHOW QUEUE BA1:).

You can include more than one control file with the SUBMIT command; these control files need not be related. The control files can each run under different accounts, depending on the parameter in the $JOB/DCL statement, if you are a privileged user.

If your system manager has provided multiple copies of the BATCH program, then more than one batch job can execute at the same time. Each copy of the BATCH program is identified by a queue name, such as BA0: or BA1:.

Batch and Pseudo Keyboards

A batch job uses a *pseudo keyboard*, which frees your terminal for other use while the job is running.

The concept of how a pseudo keyboard functions in batch procedures is often confusing to new users. For batch processing, you only need to know that the pseudo keyboard does your work for you. The pseudo keyboard "pretends" to be a terminal by interpreting the commands in the batch control file.

Similar to the way you use a terminal, BATCH uses a pseudo keyboard to interact with the system. See Figure 6–1.

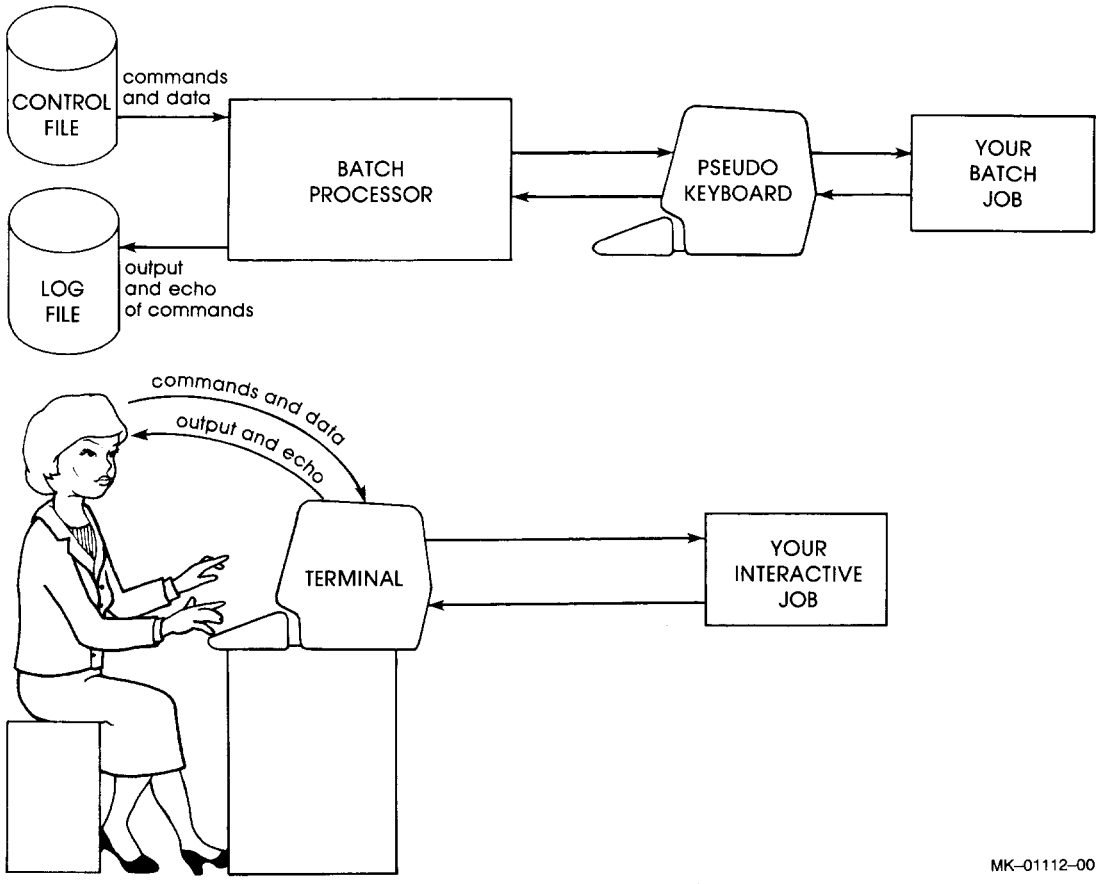


Figure 6–1: Comparison of Batch Processing and Interactive Processing

If you are interested in learning about pseudo keyboards, refer to the *RSTS/E Programming Manual.*

Using the Batch Facility

To create a batch control file and run a batch job, follow the steps described in this section. Refer to the examples in the next section and descriptions commands in this chapter for details about each step.

1. Create a control file. The file type .CTL is used by convention, and RSTS/E recognizes .CTL as the default file type for a control file.
2. Submit the job to the batch queue with the SUBMIT command.
3. Use the SHOW QUEUE/BATCH command to check the status of the batch job. (This step is optional.)

4. As your batch job runs, the system creates a log file with the same name as the control file, but with the file type .LOG. This *log file* records the dialogue that takes place at the pseudo keyboard when your job is running. The log file is important because batch does not depend on you being there to process commands.

   It is a good idea to read the log file when your batch job is complete, to be sure the procedure went as you expected.

The batch facility executes the jobs you submit in a two-step process:

1. The facility scans the control file for errors.
2. If no errors are detected, BATCH executes the commands. At the same time, BATCH also creates a log file containing its dialogue with the system. (If there are errors in the control file, the batch facility does not execute the commands.)

If program output is directed to KB: (a terminal), the output appears following the command in the control file that caused the program to execute.

Batch Examples

The following two examples show how to use the batch facility in routine operations.

The first example shows the use of batch to copy files from your account onto a magnetic tape. (The example assumes that there is a system operator to physically mount the tape onto a drive.) The second example shows how to use batch in running a payroll program.

Note that you can choose the way you create a control file. The first example uses the CREATE command; the second uses the EDIT command. It does not matter which method you use to create a control file. The CREATE command uses fewer steps, but the EDIT command allows you the versatility of the EDT editor.

EXAMPLE 1

1. Creating the Control File

   Create the control file to copy files from your default directory to magnetic tape as shown. Note that you must type a dollar sign directly before each command in the control file. The commands in the file are explained in the next section.

```
$ CREATE RUNTAP.CTL(RET)
$JOB/DCL/ERROR=WARNING/PRIORITY=-16/LIMIT=60(RET)
$MESSAGE/WAIT "Please mount tape BURNS on MT1:"(RET)
$INITIALIZE MT1: BURNS/FORMAT=ANSI/DENSITY=1600(RET)
$DATA(RET)
YES(RET)
$EOD(RET)
$MOUNT MT1: BURNS(RET)
$COPY RUNNER.* MT1:(RET)
$DISMOUNT MT1: BURNS(RET)
$MESSAGE "All done with MT1: now. Thanks!"(RET)
$EOJ(RET)
(CTRL/Z)
$
```

2. Explanation of Control File

CREATE is a DCL command that allows you to type input to the control file. You use commands within a control file by typing a dollar sign ($) directly before the command. The $JOB/DCL, $MESSAGE, $DATA, $EOD, and $EOJ commands are specifically for use in batch control files. The INITIALIZE, MOUNT, COPY, and DISMOUNT commands are standard DCL commands.

The $JOB/DCL command tells the batch processor to use the DCL form of batch (rather than one of the non-DCL forms) to accept commands from the batch stream that follows. The command qualifiers are:

/ERROR=WARNING — /ERROR determines the severity of errors the job will tolerate, if any errors occur. When you include =WARNING, the job will continue to run if it encounters warning errors. However, the job will stop if a fatal error occurs.

/PRIORITY=–16 — A nonprivileged user can set the job at a run priority of –120 to –8. –16 is in this range, and the job will run at a lower priority than the default –8.

/LIMIT=60 — /LIMIT specifies the time allowed for execution of the control file. When you include =60, the limit is 60 minutes. The job will stop if it takes longer than one hour to complete, because the time limit will have elapsed.

The $MESSAGE command asks the operator to mount a tape with the label BURNS on tape drive MT1:. The /WAIT qualifier holds the job from processing until the operator mounts the tape and responds that this is done.

The INITIALIZE command clears the tape on MT1: of any data it may contain. The tape label is BURNS, the format is ANSI, and the density is 1600 bits per inch.

The $DATA statement is included because RSTS/E issues a prompt in response to the INITIALIZE command:

Really zero MT1:BURNS . PARITY:ODD/DENSITY=1600 ?

The "data" supplied in the control file in response to the prompt is the answer YES. $EOD marks the end of the data.

The MOUNT command logically mounts the tape onto the drive, so files can be copied from your default directory to the tape. MT0: is the tape drive.

The COPY command copies all files named RUNNER in your directory onto the tape. The DISMOUNT command logically dismounts the tape; the next $MESSAGE command asks the operator to physically dismount the tape. $EOJ (end of job) signifies the end of the batch job. The $EOJ command is optional.

CTRL/Z notifies the system that you are finished including text to the control file.

3. Running the Job

You submit the job by typing:

```
$ SUBMIT RUNTAP.CTL[RET]
```

The job is placed in the generic batch queue (BA:) because you did not name a specific queue (such as BA1:). Therefore, if there is more than one copy of the batch program available on your system, your job will be processed by the first available copy. (If there is only one copy of the batch program on your system, the SUBMIT command merely puts the job in the batch queue.)

Although both the file name and type are included (RUNTAP.CTL), the .CTL file type is the default and is therefore optional. You could have typed instead:

```
$ SUBMIT RUNTAP[RET]
```

You can check the job's progress in the queue by typing:

```
$ SHOW QUEUE/BATCH[RET]

BA SHORT QUEUE LISTING 20-Dec-82 01:11PM
UNIT        JOB          S / P
 *      RUNTAP[2,214]/SE:4741/FO:NORMAL A/128/AF:19-Dec-82:10:00 PM
```

4. Reading the Log File

When the job is finished, the system places a log file named RUNTAP.LOG into your directory on the public structure. The log file records the dialogue between the BATCH program and the system, and the times when each interaction took place. It is a good idea to look at the contents of the log file to be sure the job ran as expected, as shown in Figure 6–2:

```
$ TYPE RUNTAP.LOG[RET]

------------$Job/DCL/ERROR=WARNING/PRIORITY=-16/LIMIT=60
01:11:49 PM  HELLO
01:11:50 PM  RSTS V8.0 RODEO Job 34  <Batch>  KB4  20-Dec-82  01:11 PM

01:11:50 PM  User: 2/214
01:11:51 PM  Password:
01:11:52 PM  4 other users are logged in under this account

01:11:52 PM  $
------------$MESSAGE/WAIT "Please mount tape BURNS on MT1:"
------------$INITIALIZE MT1: BURNS/FORMAT=ANSI/DENSITY=1600
01:12:35 PM  $ INITIALIZE MT1: BURNS/FORMAT=ANSI/DENSITY=1600
------------$DATA
01:12:36 PM Really zero MT1:BURNS .   /PARITY:ODD/DENSITY:1600 ? YES

------------$EOD
------------$MOUNT MT1: BURNS
01:12:39 PM  $ MOUNT MT1: BURNS

01:12:41 PM  Magtape is in ANSI format
01:12:41 PM  DENSITY IS 1600 , PARITY IS ODD

------------$COPY RUNNER.* MT1:
01:12:41 PM  $ COPY RUNNER.* MT1:
01:12:51 PM  [File RUNNER.DAT copied to MT1:[2,214]RUNNER.DAT]
01:12:51 PM  [File RUNNER.DOC copied to MT1:[2,214]RUNNER.DOC]

------------$DISMOUNT MT1: BURNS
01:12:51 PM  $ DISMOUNT MT1: BURNS

------------$MESSAGE "All done with MT1: now. Thanks!"
------------$EOJ
01:12:54 PM  $ BYE
01:12:55 PM  Saved all disk files; 648 blocks in use, 352 free
01:12:55 PM  Job 34 User 2,214 logged off KB4 at 20-Dec-82 01:12 PM
01:12:55 PM  4 other users still logged in under this account
01:12:55 PM  System RSTS V8.0 RODEO
01:12:55 PM  Run time was 2.5 seconds
01:12:55 PM  Elapsed time was 1 minute
01:12:56 PM  Good afternoon
```

Figure 6–2: Batch File RUNTAP.LOG

EXAMPLE 2

The following example uses an existing program to run a payroll. The name of the program is PAYOUT.TSK; the control file you create will be named EMPLOY.CTL.

1. Creating the Control File

   Use the EDT editor to create the control file:

```
$ EDIT/EDT EMPLOY.CTL<RET>
Input file does not exist
[EOB]
*INSERT<RET>
                $JOB/DCL<RET>
                $RUN PAYOUT.TSK<RET>
                $DATA ! The following is input to PAYOUT.TSK<RET>
                LAUREL,STAN;40;$30.00<RET>
                HARDY,OLIVER;40;$28.00<RET>
                $EOD<RET>
                <CTRL/Z>
*EXIT<RET>
File EMPLOY.CTL 6 lines

$
```

2. Explanation of Control File

   The EDIT/EDT command line starts the EDT editor to create a file. Because EMPLOY.CTL is a new file, EDT displays the message:

   Input file does not exist

   The end of buffer symbol ([EOB]) is displayed to show that the file does not yet contain text. The INSERT command after the line editing asterisk prompt (*) allows you to insert text.

   The $JOB/DCL command signals the beginning of the batch stream. RUN PAYOUT.TSK runs the program named PAYOUT, which has been compiled and linked (.TSK).

   $DATA indicates that the next lines contain data for the PAYOUT program. The exclamation mark (!) signifies a comment. Comments are useful because they can make a control file self-documenting.

   Next, salary information is displayed for the employees STAN LAUREL and OLIVER HARDY. (A real batch payroll run may include any number of employees, depending on your program and the system resources.) $EOD signals the end of data, and CTRL/Z signals to the editor that you are done inserting text.

3. Running the Job

You submit the job by typing:

```
$ SUBMIT EMPLOY(RET)
```

The command places the file EMPLOY.CTL in batch queue BA1:. To verify the status of the job, type:

```
$ SHOW QUEUE/BATCH(RET)

BA SHORT QUEUE LISTING 20-Dec-82 01:11 PM
UNIT        JOB                          S / P
 0          EMPLOY[2,214]/SE:3084/FO:NORMAL
 A/128/AF:19-Dec-82:10:00PM
```

4. Reading the Log File

When the job is finished, the system places a log file named EMPLOY.LOG in your default directory. Check the log file with the command shown in Figure 6-3:

```
$ TYPE EMPLOY.LOG(RET)

------------$JOB/DCL
12:22:09 PM  HELLO
12:22:10 PM  RSTS V8.0 RODEO Job 27  <Batch>  KB3  20-Dec-82 12:22 PM

12:22:10 PM  User: 2/214
12:22:11 PM  Password:
12:22:11 PM  2 other users are logged in under this account

12:22:11 PM  $
------------$RUN PAYOUT.TSK
12:22:11 PM  $ RUN PAYOUT.TSK
------------$DATA !The following is input to PAYOUT.TSK
12:22:12 PM  ? LAUREL,STAN;40;$30
12:22:13 PM  ? HARDY,OLIVER;40;$28

------------$EOD
12:22:14 PM  $ BYE
12:22:16 PM  Saved all disk files; 620 blocks in use, 380 free
12:22:17 PM  Job 27 User 2,214 logged off KB3 at 20-Dec-82 12:22 PM
12:22:17 PM  2 other users still logged in under this account
12:22:17 PM  System RSTS V8.0 RODEO
12:22:17 PM  Run time was .8 seconds
12:22:17 PM  Elapsed time was 0 minutes
12:22:17 PM  Good afternoon
```

Figure 6–3: Batch File EMPLOY.LOG

The log file shows the interaction between the BATCH program and the system in completing the batch job, as well as the time each command was processed.

Using Batch Commands

The syntax rules for batch control file commands are:

1. Batch commands should have a dollar sign ($) as the first character. The $ character causes DCL to recognize the command as part of a control statement.
2. The command name begins in the second character position, immediately following the $ character. A control statement must contain a command name (except in the case of the comment line "$!"). If you omit the command name, the command is ignored. An unrecognizable command name is invalid and causes DCL to print the error message "Invalid command".
3. No spaces are allowed after the dollar sign. For example, $DATA is a valid batch command, but $ DATA is not.
4. A command line can have a maximum of 120 characters.

As in the interactive use of DCL, you may include valid qualifiers after the command name. However, unrecognizable qualifiers make the control statements in which they appear invalid.

You can use CCL commands as well as DCL commands in batch streams. However, DIGITAL strongly recommends that you use the CCL prefix before each CCL command, because future releases of RSTS/E may include new DCL commands. The use of the CCL prefix ensures that there will be no conflict between your CCL commands and new DCL commands.

The proper notation of CCL commands in a batch stream is to type a dollar sign, the CCL prefix, a space, and then the CCL command. For example:

```
$CCL LIBR MYLIB=A,B,C<RET>
```

$JOB/DCL

The $JOB/DCL command marks the beginning of a batch job in DCL.

You cannot put a blank between $JOB/DCL and a qualifier, nor between the slash and the qualifier. For example, $JOB/DCL/QUE is legal, but $JOB/DCL/ QUE is not. Note that this is an exception to normal DCL rules. (Underscores show brackets that you must type.)

Format

$JOB/DCL [[proj,prog]]

Command Qualifiers	Defaults
/CPU=nnn	/NOCPU
/NOCPU	
/ERROR=operand	
/LIMIT=nnn	/LIMIT=10
/NOLIMIT	
/[NO]QUE	/NOQUE
/PRIORITY=n	/PRIORITY=-8

Parameters

[proj,prog]
A privileged user can include a project-programmer number to run the job under an account other than the account from which it was queued.

Command Qualifiers

/CPU=nnn
Assigns a CPU time limit to the job. The value of nnn, a decimal number, is interpreted as seconds. If you specify /CPU but not /LIMIT, no elapsed time limit is enforced, and the only time limit is on CPU time. If you specify both qualifiers, the job ends if either limit is exceeded. If you do not specify a time limit, the allowable CPU time is infinite.

You might want to specify a /CPU limit, for example, to prevent a "runaway" job in case of a bug in the program, such as in an infinite loop or logic error.

/NOCPU
Gives the job an unlimited amount of CPU time in which to complete. /NOCPU is the default.

/ERROR=operand
Specifies the level of error that the batch processor tolerates without ending the job. The level is indicated by operand, which may be FAT[AL], WAR[NING], or NON[E]. (See Table 6–2 for severity of error messages.) If FATAL, all errors are tolerated until completion. If WARNING, a fatal error ends the job, but warning errors are tolerated. If NONE, any error ends the job.

If a job is to be ended because the error level has been exceeded, termination occurs when the job next asks for input. A message is entered in the log file giving the reason for termination. Your system manager determines the default error level for the batch stream.

/LIMIT=nnn

Assigns an elapsed time limit to the job. The value of nnn, a decimal number, is interpreted as minutes. Note that the elapsed time taken to execute the job depends on overall system loading.

You might want to specify /LIMIT, for example, to prevent a "runaway" job in case of a bug in the program, such as in an infinite loop or logic error.

/NOLIMIT

Gives the job an unlimited amount of elapsed time in which to complete. If you give neither /LIMIT=nn nor /NOLIMIT, the job by default is given ten minutes elapsed (actual) time to complete execution before batch terminates it.

Common practice is to specify /NOLIMIT, once you have debugged the control file, so that the job will continue until it is completed.

/QUE

Queues the batch log file to LP0: (the default), or to the default device your system manager has specified.

/NOQUE

Suppresses printing of the batch log file. /NOQUE is the default.

/PRIORITY=n

Determines the execution priority of a batch job.

/PRIORITY=n sets the RSTS/E execution priority to n (or the next lowest multiple of eight) for the batch stream. For privileged users, n can be between –120 and +127. For nonprivileged users, n is limited to a value between –120 and –8. Unless otherwise altered by the /PRIORITY=n qualifier, all jobs run at –8 priority. Priorities are set in increments of 8.

Execution priority is different from queueing priority. You use the SUBMIT command to specify the *queueing priority*, which determines how quickly the job gets to the head of the queue. (The queueing priority of a job is included in the SHOW QUEUE display.) You use the $JOB/DCL command to specify the *execution priority*, which determines how quickly the job is executed after it gets to the head of the queue.

A nonprivileged user can set the execution priority of a batch job lower, but not higher, than the default of –8. (This prevents nonprivileged users from slowing down the processing of high-priority or interactive jobs.)

$DATA

The $DATA command signals the start of data in a control file. Usually the data is to be read by a program or command that the control file runs.

The $DATA command in a batch control file has the format:

$DATA

The $DATA statement is optional. $DATA ensures that if the program does not use all of its data, the remaining data is ignored, up to the next control statement (which begins with $).

$EOD

The $EOD command signals the end of data when a command or program is reading data from the batch command file. Place $EOD in the file after commands such as $CREATE, $DATA, and $RUN. (This corresponds to the interactive use of CTRL/Z.)

The $EOD command in a batch control file has the format:

$EOD

$EOJ

The $EOJ command marks the end of a batch job. Before $EOJ, the job should dismount all disks and tapes that it had mounted, and should use the $MESSAGE statement to inform the operator that the job is done with the devices.

The $EOJ command in a batch control file has the format:

$EOJ

The $EOJ command is not required in batch control files, although it is generally used in jobs submitted through a card reader.

The $EOJ command is implied if batch encounters either of the following:

- a physical end-of-file condition
- another $JOB control statement

$MESSAGE

The $MESSAGE command displays a message on the operator's console. $MESSAGE provides a way for the batch job to communicate with the operator.

The $MESSAGE command in a batch control file has the format:

$MESSAGE[/WAIT] message

Use the /WAIT qualifier to indicate a pause, so that the control file will wait for an action by the operator (such as mounting a tape). The space preceding the message is required. The message must be typed on one line. No blank can appear between the command name and the /WAIT qualifier.

When the BATCH program arrives at the /WAIT qualifier, the system pauses until the operator gives the appropriate command. For example, the following command halts the batch job until the operator takes action:

```
$MESSAGE/WAIT MOUNT SCRATCH TAPE ON MT0:
```

The WAIT condition remains in effect until the operator responds to the message on the operator services console. For information on operator response procedures, refer to the *RSTS/E System Manager's Guide.*

Batch Error Procedures

The batch facility scans control files for correctness before it begins to execute the batch job. Batch detects certain syntax errors during this scan.

When a syntax error is detected during the scan, the job is not executed. Instead, an error log is printed listing all commands and data scanned, along with the appropriate error message(s).

The batch log file always indicates all command lines scanned. If an error is found on a command line, the error message follows the command, marked with question marks (????????????). The batch facility then continues to scan the control file, but the job is not executed.

If no syntax errors are found during the scan, the times when lines are output from the batch job are indicated in the left margin of the log. All normal terminal interaction corresponding to the batch commands appears in the log.

Severity of Batch Errors

The first character in an error message indicates the severity of the error, as shown in Table 6–2. You use this information when you enter the /ERROR qualifier in the $JOB/DCL command line. For example, /ERROR=WARNING instructs the batch facility to tolerate any warning errors it may encounter, but not fatal errors. Warning errors are denoted by a percent sign (%).

Table 6–2: Severity Standard in Error Messages

Character	Severity	Meaning
%	Warning	Execution of the program can continue but may not generate the results you expect.
?	Fatal	Execution cannot continue unless you remove the cause of the error. No space or tab is allowed after the ?.
	Information	A message beginning with neither a question mark nor a percent sign is for information only.

Note

Some system programs available on RSTS/E were developed for other systems. An example is the FORTRAN–IV compiler. These other programs may not use the standard RSTS/E severity characters in error messages, so the batch facility cannot detect these errors. User programs that are coded to run under batch control should use standard severity characters.

Submitting and Controlling Batch Jobs

You use the SUBMIT command to execute the commands in a batch control file. The SET QUEUE and SHOW QUEUE commands let you monitor the status of jobs in the batch queue. To delete jobs from the batch queue, you can use either the DELETE/JOB or DELETE/ENTRY command.

The Batch Queue

When you submit a job, you normally want to select the generic batch queue (BA:, which is the default) to process the job.

The *generic batch queue* represents all jobs in all batch queues. Therefore, BA: selects whichever batch queue can take your job first. When your job gets to the head of the BA: queue, the system selects the copy of the BATCH program that is next available to process your job.

The names BA:, BA0:, BA1:, and so forth refer to queues, not to devices. You would request a specific batch queue (such as BA1:) if you have several jobs that need to be run in a particular order.

Each job entered into a batch queue is assigned a sequence number, according to the time the job is entered into the queue. The higher the priority of a job, the sooner it is handled by the queue. (Use the SHOW QUEUE/BATCH command to learn the sequence numbers of batch jobs in the queue.) Sequence numbers are consecutive; there are no duplicate sequence numbers for batch jobs.

The number of batch queues available is determined by the system manager; the maximum number of batch queues is eight. (The more batch queues, the greater the load on the system.)

Regardless of whether you are a privileged or nonprivileged user, your commands can affect only your own job in the batch queue. If you issue commands to change the status of another user's batch job, no error message is displayed, but there is no effect on the batch job.

SUBMIT

The SUBMIT command submits one or more control files for batch processing. This command can be used only with the large Spooler.

Format

SUBMIT file-spec[,...] [job-spec]

Command Qualifiers	Defaults
/AFTER = date-time	
/NAME = job-name	
/OWNER = [proj,prog]	
/PRIORITY = n	/PRIORITY = 128
/QUEUE = queue-name	/QUEUE = BA:

File Qualifiers

/[NO]DELETE	/NODELETE

Prompts

File: file-spec[,...]

A file or group of files queued by the SUBMIT command are considered a *job*. The system assigns a unique sequence number to each job in the queue. For example, if there are three jobs in the queue, numbered 4623, 4624, and 4625, the next job submitted will be numbered 4626.

When you submit batch control files for processing, all output is written to a file called name.LOG. The "name" is either the file name of the first batch control file in the job, the name you specify with the /NAME = name qualifier, or the job-name you specify in the job specification.

If you submit more than one batch control file, the job terminates if any of the files has an error or fatal error status, depending on the error level you specified on the $JOB statement. (For a description of Batch error message severity see Table 6–2.) The default error level for the batch stream is determined by the system manager at system start-up time.

The default file type for a control file is .CTL. For example, the following command puts the control file SMITH.CTL in the batch queue:

```
$ SUBMIT SMITH[RET]
```

You can check the status of your batch job by typing:

```
$ SHOW QUEUE/BATCH[RET]

BA SHORT QUEUE LISTING    20-Dec-82          01:11 PM
UNIT     JOB                                 S / P
 0       SMITH[2,214]/SE:3084/FO:NORMAL      A /128/AF:19-Dec-82:10:00 PM
```

Command Parameters

file-spec[,...]

Specifies one or more control files to be submitted as a single batch job. You must specify a file name; if you do not specify a file type, the SUBMIT command uses the default file type of .CTL. If you specify more than one file, separate them with commas (,) or plus signs (+). In either case, the files are processed as a single job. (Each control file must have its own $JOB statement.)

You can use wildcard characters in the file specification. This is useful with control files that run independently, but are related; that is, when their order is not important.

job-spec

queue-name:[proj,prog]job-name

Specifies a job identification to be submitted as a single batch job. Chapter 2 (Entering Job Specification Lists) contains more information. If you do not specify a queue-name, the default is BA:. Also, if you do not specify a project-programmer number, the default is your own. Only a privileged user can specify someone else's account. The name of your first file specification is the default if you do not specify a job-name.

You cannot use the job specification parameter if you specify /NAME, /OWNER, or /QUEUE.

Command Qualifiers

/AFTER=date-time

Requests that the job be held until after a specific date or time. If the specified date or time has already passed, the job is queued for immediate processing. For example, if you want a batch job named NIGHT.CTL to be run at midnight, type:

```
$ SUBMIT NIGHT/AFTER=11:59PM[RET]
```

Refer to Chapter 2, "Specifying Dates and Times," to use a date or time with the /AFTER qualifier.

/NAME=job-name

Defines a name of one- to six-alphanumeric characters to identify the job. You use the job-name with the SET QUEUE/JOB and DELETE/JOB commands. You cannot specify /NAME if you use the job specification parameter.

Each job has a job-name which is the name of the first input file or the name you specify with the /NAME qualifier.

For example, if your batch control file is named RUFUS, RSTS/E assumes that your job is named RUFUS unless you change it with the /NAME qualifier. The output file has the same name as the job, and a file type of .LOG. It is created in your directory on the public disk structure.

For example, to name the job that uses three files (ROUTE1.CTL, ROUTE2.CTL, and ROUTE3.CTL) as COURSE, type:

```
$ SUBMIT ROUTE1,ROUTE2,ROUTE3/NAME=COURSE↵
```

The job-name and the log file name are displayed by the SHOW QUEUE command:

```
$ SHOW QUEUE/BATCH/FULL↵

BA QUEUE LISTING                    20-Dec-82 01:11 PM
UNIT                                JOB   S / P      FILES
 0      COURSE[2,214]/SE:3084/FO:NORMAL/SKI
                          A /128/AF:19-Dec-82:10:00 PM/TY:EMB
                                    SY :[2,214]ROUTE1.CTL
                                    SY :[2,214]ROUTE2.CTL
                                    SY :[2,214]ROUTE3.CTL
```

In this example, the job would be named ROUTE1 if you did not use the /NAME qualifier.

/OWNER=[proj,prog]

Indicates the owner of the job to be printed. The project-programmer number is displayed on the job header page. If you are nonprivileged, you can specify only jobs with your PPN.

/PRIORITY=n

Specifies the queueing priority (as distinct from the run priority) for the job. Your job can move ahead of or behind other jobs in the queue, depending on the priority you set.

The priority, n, must be in the range of 0 through 255, where 0 is the lowest priority and 255 is the highest.

By default, jobs are queued at the priority of 128. You need a privileged account to give your job a priority higher than the default.

/QUEUE=queue-name

Specifies the batch queue that you want to process the job. You cannot specify /QUEUE if you include a queue-name in the job specification parameter.

Usually it is best to allow the generic batch queue (BA:, which is the default) to select a specific copy of the BATCH program. In other words, if you do not use the /QUEUE qualifier, then the generic queue BA: submits your job to the first available copy of the BATCH program. This way your job is queued the fastest.

You would request a specific batch queue (such as BA1:) if you have several jobs that need to be run in a particular order. To specify a particular copy of the BATCH program, use the format:

```
$ SUBMIT FINISH.DAT/QUEUE=BA1:↵
```

You should, however, be aware that you will not get an error message if you queue a job to a nonexistent queue. For example, if you queue a job to queue BA7: and your system only has queues BA0: and BA1:, no message is displayed and your job is not processed.

/DELETE
/NODELETE

Specifies whether a control file is to be deleted after processing. The default is /NODELETE.

SHOW QUEUE (Batch Only)

The SHOW QUEUE command displays information about the status of jobs in a queue. Two forms of SHOW QUEUE apply to batch operations. You can either use the /BATCH qualifier or specify the name of a batch queue.

Format

SHOW QUEUE/BATCH

OR

SHOW QUEUE [=] queue-name[:]

Qualifiers	**Defaults**
/ALL	
/BRIEF	/BRIEF
/FILES	
/FULL	

Using the /BATCH qualifier, for example, you can type:

```
$ SHOW QUEUE/BATCH
```

If you want to display a specific batch queue, you can type:

```
$ SHOW QUEUE BA1:
```

If you specify a queue-name, you cannot include the /BATCH qualifier in the command line.

SHOW QUEUE/BATCH is equivalent to SHOW QUEUE BA:.

Qualifiers

/ALL
Indicates that wildcard defaults should be used in the job-spec. This is an easier way to list all jobs. If you do not specify /ALL, the job specification defaults to all jobs with your project-programmer number.

/BRIEF
Provides a limited amount of information about jobs in the batch queue. This is the default display. The /BRIEF qualifier is especially useful when you want to know the order in which jobs in the queue will be processed. For example:

```
$ SHOW QUEUE/BATCH/BRIEF

BA SHORT QUEUE LISTING  25-Nov-82        01:11 PM
UNIT     JOB                                S / P
 0       PHYDOE[2,214]/SE:3084              S /128/TY:EMB
```

/FILES
Indicates that all file specifications and their qualifiers included in the job should be displayed. However, if you are nonprivileged, only files with your PPN are displayed. /FULL gives you other information as well. With the large Spooler, /FILES and /FULL display the same information.

/FULL
Indicates that full job status should be displayed. For example:

```
$ SHOW QUEUE/FULL[RET]

LP0 QUEUE LISTING          20-Dec-82        03:10 PM
UNIT     JOB               S / P FILES
 0       MUSIC [2,223]/SE:3747/FO:NORMAL/SKI
                           A /128/AF:19-Dec-82:10:00 PM/TY:EMB
                                     SY :[2,223]MUSIC.TXT
```

SET QUEUE (Batch Only)

The SET QUEUE command changes the qualifiers of jobs in a queue. (For complete information about the SET QUEUE command, refer to Chapter 3.) This command can be used only with the large Spooler.

You can change the qualifiers of a job in the batch queue either by name (SET QUEUE/JOB) or by number (SET QUEUE/ENTRY). You specify a job by number if you have more than one job with the same name in the queue. Otherwise, you may prefer to specify jobs by name.

SET QUEUE/JOB

The SET QUEUE/JOB command specifies one or more jobs in a queue by name.

Format

SET QUEUE/JOB [=] job-spec

Command Qualifiers	Defaults
/AFTER=date-time	None
/BATCH	
/HOLD	
/PRIORITY=n	
/RELEASE	

When you submit a batch job, the job is assigned a name, according to the first input file specification or the name you specify. You can use this name to modify the status of the job in the batch queue. For example:

```
$ SUBMIT SMITH.CTL(RET)
```

If you then decide to lower the job's queuing priority from the default 128 to 100, you type:

```
$ SET QUEUE/JOB=SMITH/PRIORITY=100/BATCH(RET)
```

Parameters

job-spec

queue-name:[proj,prog]job-name

Specifies the name of one or more jobs you want to modify in the queue. If you are nonprivileged, you can modify jobs only if they were queued from your account. (Your account is the default project-programmer number.) You must include the name of a queue in batch processing; otherwise, BA: is assumed. /BATCH is equivalent to BA:. (You cannot assign logical names to queues.)

If you use a wildcard (*) for job-name, this denotes all jobs in the queue that were submitted or printed from your account. If no jobs in the queue match the job-name you specify, the command has no effect; however, no error message is displayed. There is no default for the job-name parameter.

See Chapter 2 (Entering Job Specification Lists) for a description of the parameters.

Command Qualifiers

/AFTER = date-time

Requests that the specified job be held until a specific date and time and then released for printing or processing. For example, you type the following command line to queue a job named THANKS for processing after October 2 at 1:00 in the afternoon:

```
$ SET QUEUE/JOB=THANKS/AFTER=02-Oct-82-01:00PM/BATCH
```

If the specified date or time has already passed, the file is released immediately.

/BATCH

Requests that the specified job(s) be submitted to the Batch Processor. You may alternately specify the generic batch queue BA: in the job specification.

/HOLD

Requests that the specified job(s) be placed in a hold status. Jobs in a hold status are not processed until you release them with the /RELEASE qualifier. The /HOLD qualifier might be useful to queue a list of jobs, and decide at a later time when you need to process them (especially if other users have crucial time constraints for processing their jobs).

To put a job named FINISH in a queue, to be processed at a time that you will specify later, type:

```
$ SET QUEUE/JOB=FINISH/HOLD/BATCH
```

/PRIORITY = n

Changes the queuing priority of a job. The priority n must be in the range of 0–255 where 0 is the lowest priority and 255 is the highest. By default, the job's priority in the queue is unchanged.

For example, to set the priority of a job named SCAMP to 100, you type:

```
$ SET QUEUE/JOB=SCAMP/PRIORITY=100/BATCH
```

/RELEASE

Releases a job that was previously held from processing with the /HOLD qualifier.

For example, suppose you submit file MINDY.CTL for processing, and then put the job on hold in the queue:

```
$ SUBMIT MINDY.CTL[RET]
$ SET QUEUE/JOB=MINDY/HOLD/BATCH[RET]
```

Later, you release the job from its hold status by typing:

```
$ SET QUEUE/JOB=MINDY/RELEASE/BATCH[RET]
```

The /RELEASE qualifier causes the system to process file MINDY.CTL in its turn in the queue.

SET QUEUE/ENTRY (Batch Only)

SET QUEUE/ENTRY

The SET QUEUE/ENTRY command line specifies a job in a queue by number. (Use the SHOW QUEUE/BATCH command to determine the job's number in the queue.)

You can use two forms of the command with batch jobs. If you use the second form of the command listed below, you must include either an equal sign = or space between the command and queue-name you specify.

Format

SET QUEUE/ENTRY [=] job-number/BATCH

OR

SET QUEUE/ENTRY [=] job-number queue-name[:]

The /BATCH qualifier in the first form specifies the generic batch queue. The queue name you supply in the second form is the name of a specific batch queue, such as BA1:, or the generic batch queue BA:.

You might want to use the SET QUEUE command to change the time when a job in the queue can begin executing. For example, /AFTER=15:00 uses the 24–hour time format to have the job printed after 3:00 PM:

```
$ SET QUEUE/ENTRY=9476/BATCH/AFTER=15:00RET
```

Use the SHOW QUEUE/BATCH command to see the results:

```
$ SHOW QUEUE/BATCH/FULLRET

BA QUEUE LISTING          25-Nov-82    01:11 PM
UNIT     JOB              S / P        FILES
 *       TURKEY[1,9]/SE:9476
                          A /128/AF:03:00 PM/TY:EMB
                          SY :[2,9]TURKEY.CTL
```

As shown in the display, the job will be processed after 3:00 PM.

DELETE/JOB (Batch Only)

The DELETE/JOB command deletes one or more jobs from the batch queue by job name.

Format

DELETE/JOB=job-spec/BATCH

Parameters

job-spec

queue-name:[proj,prog]job-name

Specifies the name of one or more jobs you want to modify in the queue. You cannot use the /BATCH qualifier if you specify a queue-name in job-spec. If you are nonprivileged, you can modify jobs only if they were queued from your account. (Your account is the default project-programmer number.) If you do not specify a queue-name, the system assumes the default BA:. (You cannot assign logical names to queues.)

If you use a wildcard (*) for job-name, this denotes all jobs in the queue that were submitted or printed from your account. If no jobs in the queue match the job-name you specify, the command has no effect; however, an error message is not displayed. There is no default for the job-name parameter.

See Chapter 2 (Entering Job Specification Lists) for a description of the parameters.

You can choose either a specific queue or the generic batch queue, although you cannot specify both in one command line. For example, the following two command lines have the same meaning:

```
$ DELETE/JOB=BA:HURDLE<RET>
$ DELETE/JOB=HURDLE/BATCH<RET>
```

Both of these command lines delete the job named HURDLE from the generic batch queue.

DELETE/JOB deletes only jobs that were submitted from your account if you are nonprivileged. If you are privileged, you may specify jobs submitted from other accounts using the job-spec syntax. If there are two or more jobs in the queue that have the specified job name, the command deletes all of them.

DELETE/ENTRY (Batch Only)

The DELETE/ENTRY command removes jobs from a queue. There are two forms of the command you can use to delete batch jobs: one uses the /BATCH qualifier for the generic batch queue, and the other uses a specific batch queue-name.

Format

DELETE/ENTRY [=] job-number/BATCH

OR

DELETE/ENTRY [=] job-number queue-name[:]

DELETE/ENTRY requires you to use the sequence number of the job (instead of the job name, as in the DELETE/JOB command). For example, you delete a batch job with the sequence number 8922 by typing either of the following:

```
$ DELETE/ENTRY=8922/BATCH
$ DELETE/ENTRY=8922 BA:
```

Use the DELETE/ENTRY/BATCH command when you have more than one job with the same name in the queue. To find out the sequence number of your job, use the SHOW QUEUE/BATCH command.

The job(s) to be deleted must be your own. You can delete jobs that have not yet begun processing or that are currently being processed. You can specify either /BATCH for the generic batch queue (BA:) or a specific queue-name, but not both; otherwise, DCL displays the error message:

```
?Do not specify a queue-name with /BATCH
```

When you delete a job from the queue, no message is displayed to state that the job was deleted. Use the SHOW QUEUE/BATCH command to see that the job was deleted from the queue.

Command Parameters

sequence-number
Specifies the sequence number of the job to be deleted from the queue.

queue-name[:]
Specifies the name of the queue in which the job exists, such as BA2:.

Program Development 7

This chapter describes DCL commands for developing, linking, and running programs on RSTS/E. Refer to the appropriate programming manuals for information about using each language.

Note

This manual does not explain programming concepts. To use this chapter, you need to know how to write programs in the language you select.

Table 7–1: Program Development Commands

Programming Languages and Environments	
BASIC	Invokes the BASIC–PLUS or BASIC–PLUS–2 programming environment, depending on the qualifiers you use or the default set by your system manager. (See page 7–7.)
COBOL	Compiles COBOL–81 programs. (See page 7–8.)
DIBOL	Compiles a DIBOL–11 program. (See page 7–13.)
FORTRAN	Compiles a FORTRAN program. Two compilers are available: FORTRAN–IV and FORTRAN–77; depending on the qualifiers you use or the default set by your system manager. (See page 7–15.)
MACRO	Assembles a program written in MACRO–11. (See page 7–22.)
Programming Operations	
LINK	Links together object files to produce an executable file. (See page 7–24.)
RUN	Executes a system or user program. (See page 7–42.)

Developing Programs on RSTS/E

You can develop programs on RSTS/E using a variety of techniques and languages. BASIC–PLUS (supplied with all RSTS/E systems) is highly interactive, and provides a self-contained programming environment.

BASIC–PLUS is an interpreter. As you enter lines of source code into the system, BASIC–PLUS immediately translates them into code that the system can execute. Thus, you can run a program right after you finish entering it.

Other high-level languages (such as BASIC–PLUS–2, COBOL, DIBOL, and FORTRAN) are compilers, generating PDP–11 object code. To develop programs in these languages, you must create a source program, then compile it and link it before it can be run.

In addition, the MACRO–11 assembler (supplied with all RSTS/E systems) lets you write programs that correspond line-for-line with PDP–11 machine instructions. You follow the same general steps to develop MACRO programs as you do for compilers.

BASIC–PLUS and BASIC–PLUS–2 are *command environments* on RSTS/E. This means that you can switch from DCL command level to a BASIC environment, which has its own prompt for accepting your input. (The BASIC–PLUS prompt is "Ready", and the BASIC–PLUS–2 prompt is usually "BASIC2", depending on what your system manager has specified.) By contrast, you develop programs in other languages by using DCL commands.

The number and type of programming tools available on any given RSTS/E system (as well as the language you choose for any given application) depend on many factors. These factors include the size of the system, its cost, and the application environment.

Because almost any new program will contain errors, you will often need to use one or more steps of the process several times. The development process can be viewed as the *program development cycle.*

The following is an overview of the program development cycle for *compiled* languages — that is, those RSTS/E languages other than BASIC–PLUS. For information about programming in a specific language (including sample programs and procedures), refer to the appropriate language manual.

Overview

RSTS/E offers a number of different languages for program development. Although each language is unique, you use a similar process when developing any program that is compiled to machine language.

Briefly, the major steps are:

1. Paper – design, planning, and initial coding.
2. Edit – entering code into a *source file.*
3. Compile – compiling or assembling source code into an *object file.*
4. Link – combining object files into an executable file.
5. Test – making trial runs of an executable file and finding errors.

The next four sections introduce the tools you use for the edit, compile, link, and test steps. (The paper step does not require the programming tools available on RSTS/E.) The commands you use in each step depend on the programming language you select.

Editing

A source program is a file containing text; this is the form of your program that you actually write. The text in a source program consists of statements in BASIC, COBOL, DIBOL, FORTRAN, MACRO, or some other high-level language.

You may or may not first write out your source program (or your changes to it) on paper. In any case, you must create a file containing the text of your source statements. To do this, you generally use an editor, such as EDT (which you invoke with the EDIT command). An editor allows you to type lines of the program and save it as a file. When the program lines are entered into the file the way you want them, you are ready for the compile step.

Compiling

There is a command for compiling each high-level language. The compiler reads the source-code file you have created with your editor, compiles it into PDP–11 machine code, and writes the compiled machine code into an *object* file. On PDP–11 systems, the object output is called a "module;" it contains your program in the binary language that, when linked, is executable by a PDP–11 computer.

A single line of source code, especially in a high-level language, may be compiled into many PDP–11 instructions. The efficiency (and improved clarity) of not having to write machine instructions a line at a time is one of the important reasons for using a high-level language.

MACRO is called an "assembler." For this discussion, its function is the same as that of any compiler. MACRO turns source code into object code, and it fits into the program development cycle the same way as do the compilers.

In addition, the MACRO assembler can produce a listing that shows both your source code and the generated object code. This listing, like other listings prepared by compilers, is helpful during the debugging process.

Another listing that is useful in debugging is the "cross-reference" listing. This listing shows where in your source program each symbol is referenced. For the MACRO assembler, you can request this cross-reference as part of the assembler's program listing. For some of the other languages, there are separate programs that operate independently of the compilers to produce cross-reference listings. For example, BASIC–PLUS has BPCREF; BASIC–PLUS–2 has B2CREF.

Linking

A program that is ready for execution by the system is called an *executable file.* (Other terms for executable file are *save image*, *task*, or *load module.*)

To prepare an executable file from your (compiled) object module, you must *link* it. Linking a program into an executable file performs three major functions:

1. Combination of modules: Several separately compiled object files can be linked together into a single executable file.

2. Relocation of addresses: Because the assignment of actual memory locations for any object module must depend on the other object modules which go into the load module with it, the link step is the time when references to absolute memory locations are fixed.

3. Other object modules supplied by the system can be incorporated into your executable file. Each high-level language requires certain object modules to be linked into executable files for programs written in that language.

Combining Modules

There are two main reasons why you want to be able to link an executable file from separately compiled object modules. First, if your program is complex, it is very helpful to be able to compile it in several smaller pieces as opposed to one large program. During debugging, you can then recompile only the one smaller piece that you have changed before linking a new executable file for further testing. Second, you can link *subroutines* into a main program. You do so by maintaining a *library* of subroutines from which the link step can extract object modules. Therefore, you can reduce repetitious source coding in your main program.

The use of subroutines also allows you to write the main portion of your program in one language, and other portions in another. For instance, when writing a main program in FORTRAN, you might want to code a special or performance-sensitive subroutine in MACRO.

Relocating Addresses

The final location of an object module in a linked executable file cannot be determined until the object modules are combined in the link step. Therefore, object modules are *relocatable.* In addition to binary machine instructions, an object module contains tables of information that direct link-time operations such as address relocation.

The link step can optionally produce a listing of resolved addresses, called a *map.*

Testing

You can rarely create a program that does not contain at least one error, either in its logic or in its coding. You may discover errors while you are editing the program. Furthermore, the compiler may find errors during the compilation process. The compiler informs you of any errors it finds by using error flags in the listings. The link process may also catch certain errors and issue appropriate messages.

Often, however, it is not until execution (when you run your program) that you discover the program does not work properly. Some programming errors are difficult to find. For this reason, special debugging tools are supplied with most high-level languages. Refer to manuals about the programming language you are using for information about the debugging tools it has available.

The RT11 and RSX Tools

Most of the high-level languages available on RSTS/E are very similar to corresponding languages on RT–11 or RSX–11. Therefore, the program development commands you use on RSTS/E are similar to the RT11 and RSX program development commands.

For example, when you develop a FORTRAN–IV program on RSTS/E, you use commands similar to those on RT11. When you develop a program in other high-level languages (including FORTRAN–77, but excluding BASIC–PLUS), you use commands similar to those on RSX. You can develop MACRO–11 programs using either set of development commands.

The description of the program development cycle stated earlier applies to the commands for either RT11 or RSX. However, the appropriate command for each programming environment is unique: there is one set of commands for RT11, and another set for RSX. Table 7–2 shows the two sets of commands.

Table 7–2: Program Development Commands on RSTS/E

Generic Name	Command Used for RT11–based Language	Command for RSX–based Language
COBOL	–	COBOL
DIBOL	–	DIBOL
FORTRAN	FORTRAN/FOR	FORTRAN/F77
MACRO	MACRO/RT11	MACRO/RSX11
Linker	LINK/FOR LINK/RT11	LINK/F77 LINK/C81 LINK/DIBOL LINK/RSX11 LINK/BP2

The file types assigned to executable files you develop are .SAV for the RT11–based linker and .TSK for the RSX–based linker.

The file type of object files is ".OBJ" in either environment.

RT11–Based Programming

You can use RSTS/E's RT11–based commands to develop programs in either FORTRAN–IV or MACRO. Depending on your application, the programs you develop can be either:

1. Run directly on your RSTS/E system.
2. Debugged on RSTS/E, compiled, and run on an RT–11 system.

Using MACRO, you can write programs that use any feature of RSTS/E, or programs that use only RT11 monitor directives.

RSX–Based Programming

You can use RSTS/E's RSX–based commands to develop programs in MACRO or any of the high-level languages. Depending on your application, your developed programs can be either:

1. Run directly on your RSTS/E system.
2. Debugged on RSTS/E, compiled, and run on an RSX–11 system.

Using MACRO, you can write programs that use any feature of RSTS/E, or programs that use only RSX monitor directives. Certain features of RSTS/E are also available directly through some high-level languages (for example, the BASIC–PLUS SYS calls, which are also available in BASIC–PLUS–2).

If you want to use the RMS data management package in your application, you will be using RSTS/E's RSX–based languages. (Refer to documentation on RMS for more information.)

BASIC

The BASIC command places you in the BASIC–PLUS or BASIC–PLUS–2 programming environment, depending on the qualifiers you use and the system's default.

Format	
BASIC	
Qualifiers	**Defaults**
/BP2 /BPLUS	/BP2

These commands place you in the BASIC–PLUS or BASIC–PLUS–2 programming environment. In BASIC–PLUS, you can use BASIC–PLUS keyboard monitor commands (for program development), or CCL commands. In BASIC–PLUS–2, you can use program development commands.

To return to DCL from the BASIC–PLUS–2 environment, type $DCL or $SWITCH. To return to DCL from the BASIC–PLUS environment, type DCL or use the SWITCH program. Refer to the appropriate language manual for details on the commands you can use in these environments.

BASIC/BP2 invokes the BASIC–PLUS–2 environment to begin a programming session:

```
$ BASIC/BP2RET
```

A prompt is displayed to indicate the BASIC–PLUS–2 environment:

```
BASIC2
```

All subsequent command input is read by BASIC–PLUS–2.

BASIC/BPLUS invokes the BASIC–PLUS environment:

```
$ BASIC/BPLUSRET
```

A prompt displays the BASIC–PLUS environment:

```
Ready
```

Qualifiers

/BP2
Invokes the BASIC–PLUS–2 programming environment. The /BP2 qualifier is the default, unless your system manager has changed it.

/BPLUS
Invokes the BASIC–PLUS programming environment.

COBOL

The COBOL command compiles a COBOL–81 program. (You can compile one source file at a time with COBOL–81.)

Format

COBOL [source_file]

Qualifiers	Defaults
/[NO]ANSI_FORMAT	/NOANSI_FORMAT
/C81	/C81
/[NO]CHECK	/CHECK
/CHECK = [NO]BOUNDS	
/CHECK = [NO]PERFORM	
/CODE = [NO]CIS	See description
/[NO]CROSS_REFERENCE	/NOCROSS_REFERENCE
/[NO]DEBUG	/NODEBUG
/DIAGNOSTICS[= diagfile]	/NODIAGNOSTICS
/NODIAGNOSTICS	
/LIST[= listfile]	/NOLIST
/NOLIST	
/[NO]MAP	/NOMAP
/NAMES = aa	/NAMES = SC
/OBJECT[= objfile]	/OBJECT[= objfile]
/NOOBJECT	
/[NO]SHOW	/NOSHOW
/SHOW = [NO]MAP	
/[NO]SUBPROGRAM	/NOSUBPROGRAM
/TEMPORARY = device	/TEMP = SY:
/[NO]TRUNCATE	/NOTRUNCATE
/[NO]WARNINGS	/WARNINGS
/WARNINGS = [NO]INFORMATIONAL	

Prompts

File: source_file

Command Parameters

source_file

Specifies a COBOL program to be compiled. The source file must contain a file name. If you do not include a file type, the compiler uses the default type of .CBL.

No wildcards are allowed in the source file.

Qualifiers

/ANSI_FORMAT
/NOANSI_FORMAT
Indicates whether the source program is in ANSI COBOL format or in DIGITAL's terminal format.

An ANSI COBOL source file has 80–character records with Area A beginning in column position 8.

The recommended default is for the COBOL command to assume that the input records are in terminal format (Area A begins in column position 1 and the lines do not have line numbers). However, your system manager can change the default at installation.

/C81
Specifies the COBOL–81 compiler. This is the default.

/CHECK
/NOCHECK
/CHECK=[NO]BOUNDS
/CHECK=[NO]PERFORM
Controls whether the compiler produces extra code to check for program correctness at run time. /CHECK verifies subscripts, indexes, and perform statement ranges. /CHECK is the recommended default and causes COBOL–81 to check each subscript and table at run time against the ranges defined by its data-name's OCCURS clause. (Your system manager can change this default at installation.) /CHECK also causes COBOL–81 to determine if your program's PERFORM ranges are nested properly (if nested at all). If COBOL–81 detects improper nesting during execution, it issues an error message to that effect.

/NOCHECK suppresses all checking which can save execution time and decrease task image size. If /NOCHECK is used to suppress error checking, an out-of-range subscript or index does not generate an error message. /NOCHECK also stops the compiler from generating code needed for checking PERFORM statement ranges. If you use /NOCHECK and the program's PERFORM statements are nested incorrectly, the program does not produce valid results.

You can have the compiler generate code to check only subscripts and indexes by specifying /CHECK=(BOUNDS, NOPERFORM). Likewise, you can check only PERFORM statement ranges by specifying /CHECK=(PERFORM, NOBOUNDS).

/CODE=CIS
/CODE=NOCIS
Controls whether the compiler uses CIS (Commercial Instruction Set) in the object code it produces. CIS speeds program execution, but not all processors support CIS.

If your processor has CIS, the recommended default is /CODE=CIS; however, your system manager can change the default at installation. If your processor does not have CIS, the default is /CODE=NOCIS.

/CROSS_REFERENCE
/NOCROSS_REFERENCE

Controls whether the compiler listing includes a cross reference listing. By default, the compiler does not create a cross reference listing. Also, you will receive the error message, "?Additional qualifier needed" unless you specify the /LIST qualifier.

/DEBUG
/NODEBUG

Causes the compiler to use the COBOL–81 Symbolic Debugger. (The Debugger lets you control and monitor your program as it runs by referring to the source version rather than the object code produced by the compiler.) The compiler generates symbol information in the object module for all data-names and procedure-names. This increases the size of the object file, but when you finish debugging and no longer need the symbols, you can recompile without using this qualifier.

The recommended default is /NODEBUG; however, your system manager can change the default at installation.

If you use the Symbolic Debugger in your program, you must also use the LINK/DEBUG command to include the Debugger in your task image. (See the LINK command for more information.)

/DIAGNOSTICS[=file]
/NODIAGNOSTICS

Controls whether the compiler produces an output diagnostic file.

The diagnostic file is a subset of the list file. It contains only the diagnostics issued, along with the line of source code to which each diagnostic applies.

By default, the COBOL–81 compiler does not create a diagnostics file. If you specify /DIAGNOSTICS without a file specification, the compiler creates a diagnostic file with the same name as the object file, and in the same directory on the same device as the object file, but with a .DIA file type. If you also specify /NOOBJECT, then the file has the same name as the source file, and is in your directory on SY:.

/LIST[=listfile]
/NOLIST

Controls whether the compiler produces an output listing.

The COBOL–81 compiler, by default, does not create a listing file. If you specify /LIST without a file specification, the compiler creates a listing with the same name as the object file, and in the same directory on the same device as the object file, but with a .LST file type. If you also specify /NOOBJECT, then the file has the same name as the source file, and is in your directory on SY:.

If you include a file specification, the listing is written to that file or device. .LST is the default file type.

/MAP
/NOMAP

/MAP is equivalent to /SHOW=MAP. /NOMAP is equivalent to /SHOW=NOMAP. This qualifier exists for compatibility with previous releases of RSTS/E.

/NAMES=aa

Requests the compiler to generate program section names ending with the two-character alphanumeric suffix, aa. If you link several object files into one executable file, using segmentation, then each object file must have a different suffix for its program section names.

If the /NAMES qualifier is not specified, the default suffix SC is used.

/OBJECT[=objfile]
/NOOBJECT

Controls whether the compiler produces an object file and a skeleton file. These files are used as input to the LINK command, which in turn produces an executable program.

By default, (or if you specify /OBJECT without a file specification), the compiler produces an object file with the same file name as the input file in your directory on the public disk structure, with a file type of .OBJ. The compiler also uses the default file type of .OBJ when you include a file name but no file type with the /OBJECT qualifier. The skeleton file is in the same directory and has the same name as the object file, but has a .SKL file type.

You cannot use wildcard characters in the file specification.

/SHOW
/NOSHOW
/SHOW=MAP
/SHOW=NOMAP

Determines whether the compiler produces a Data Division map showing the memory addresses for Data Division entries. /SHOW and /SHOW=MAP perform the same operations. /NOSHOW and /SHOW=NOMAP also produce the same results and are the recommended defaults. However, your system manager can change the default at installation.

You can use this qualifier only in conjunction with /LIST.

/SUBPROGRAM
/NOSUBPROGRAM

/SUBPROGRAM indicates the compiler is compiling a subprogram. You need to use this qualifier only if the subprogram has not passed parameters from the calling program — that is, if the subprogram does not contain the PROCEDURE DIVISION USING header. /NOSUBPROGRAM is the default.

/TEMPORARY = device

Changes the storage area for temporary work files from SY: (the default) to the value you specify for "dev"; this is useful if you are running out of space on the system disk, or if you have a faster disk available. Because of the way the compiler names its temporary work files, two simultaneous compilations of source programs with the same file name, in the same account, (regardless of the types used) produces unpredictable results. This problem will not occur, however, if you use the /TEMPORARY = dev qualifier to specify a nondefault device for those files.

/TRUNCATE
/NOTRUNCATE

/TRUNCATE specifies that the compiler perform decimal truncation on the values of COMP data items.

The recommended default is for COBOL–81 to perform binary truncation rather than decimal truncation. (Your system manager can change the default at installation.) With binary truncation, the maximum value a COMP item can contain depends on its storage allocation. If you specify /TRUNCATE, the maximum value depends on the item's PICTURE character-string. (See the *COBOL–81 RSTS/E User's Guide* for more information.)

/WARNINGS
/NOWARNINGS
/WARNINGS = INFORMATIONAL
/WARNINGS = NOINFORMATIONAL

/NOWARNINGS and /WARNINGS = NOINFORMATIONAL stop the compiler from issuing informational diagnostics during the compilation. If you specify either of these qualifiers, only warning and fatal diagnostics will appear in the list file, diagnostics file, and diagnostic summary.

/WARNINGS and /WARNINGS = INFORMATIONAL, the recommended defaults, specify that the compiler issue informational diagnostics during the compilation.

DIBOL

The DIBOL command compiles a DIBOL–11 program. You can include up to six source file specifications to be compiled into a single object file with the DIBOL compiler. The default file type is .DBL.

Format

DIBOL filespec[,...]

File Qualifiers	Defaults
/[NO]DEBUG	/NODEBUG
/LIST[=listfile]	/NOLIST
/NOLIST	
/OBJECT[=objfile]	/OBJECT
/NOOBJECT	
/WARNINGS	/WARNINGS
/NOWARNINGS	

File Qualifiers

/DEBUG
/NODEBUG

Controls whether the compiler uses the DIBOL Debugging Technique (DDT).

If you specify /DEBUG, you must also use the /DEBUG qualifier when you compile the program. If your program consists of a main program and one or more separately compiled subroutines, and you want to use the debugging tool on the entire program, each subroutine must be compiled with the /DEBUG qualifier. You can mix sections of your program that have been compiled with /DEBUG with sections that have not been compiled with /DEBUG. However, the debugging tools will be used only in the sections compiled with /DEBUG.

If you use /DEBUG and it cannot be applied, you will receive the error message, "?Additional qualifier needed".

/LIST[=listfile]
/NOLIST

Controls whether the DIBOL compiler produces an output listing.

The compiler, by default, does not create a listing file. If you specify /LIST without a file specification, then the compiler gives a listing file the same file name as the object file (or, if you specify /NOOBJECT, as the first input source file). The file is placed in the same directory on the same device as the object file, but with a .LST file type.

You cannot use wildcard characters in the file specification.

/OBJECT[= objfile]
/NOOBJECT
Controls whether the compiler produces an output object file.

By default, or if you specify /OBJECT without a file specification, the compiler produces an object module that has the same file name as the first input source file, in your directory on the public disk structure, with a file type of .OBJ.

You cannot use wildcard characters in the file specification.

/WARNINGS
/NOWARNINGS
Controls whether the compiler produces diagnostic messages for warning conditions.

Use /NOWARNINGS to override the compiler default. (The default is to issue warning diagnostic messages.)

FORTRAN

The FORTRAN command compiles up to six FORTRAN source files into a single object file.

There are two FORTRAN compilers available on RSTS/E:

Command	Invokes
FORTRAN/F77	FORTRAN–77
FORTRAN/FOR	FORTRAN–IV

FORTRAN/F77

The FORTRAN/F77 command invokes the FORTRAN–77 compiler.

Format

FORTRAN/F77 file-spec[,...]

Command Qualifiers	Defaults
/[NO]CHECK	/CHECK
/CONTINUATIONS=n	/CONTINUATIONS=19
/[NO]D_LINES	/NOD_LINES
/[NO]I4	/NOI4
/[NO]IDENTIFICATION	/NOIDENTIFICATION
/LIST[=listfile]	/NOLIST
/NOLIST	
/[NO]MACHINE_CODE	/NOMACHINE_CODE
/OBJECT[=objfile]	/OBJECT
/NOOBJECT	
/[NO]WARNINGS	/WARNINGS
/WORK_FILES=n	/WORK_FILES=2

Prompts

File: file-spec[,...]

Command Parameters

file-spec[,...]

Specifies one or more FORTRAN source programs to be compiled. If you do not specify a file type, the compiler uses the default file type of .FTN.

You can specify more than one input file. The files are compiled as a single input file. The result is one object module and, if /LIST is specified, one listing.

You cannot use wildcard characters in the file specification.

Command Qualifiers

/CHECK
/NOCHECK

Controls whether the compiler produces extra code that checks for program correctness at run time. /CHECK causes the production of code to check that all array references are to addresses within the address boundaries specified in the array declaration.

/CONTINUATIONS=n

Specifies the maximum number of continuation lines to be permitted.

The number of lines, n, is a decimal number from 0 through 99. By default, the compiler accepts a maximum of 19 continuation lines.

/D_LINES
/NOD_LINES

Indicates whether the compiler reads and compiles debugging lines, which have a D in column 1 of the source program. (Debugging lines are used to check that the program is working, and they are not needed after the program is debugged.) If you specify /D_LINES, lines that have a D in column 1 are compiled.

The default is /NO_DLINES, which means the compiler assumes that lines beginning with a D are comments and does not compile them.

/I4
/NOI4

Controls how the compiler interprets INTEGER and LOGICAL declarations that do not specify a length.

/I4 allocates two words (four bytes) for the default length of integer and logical variables.

If you specify /NOI4, the compiler interprets the declarations as INTEGER*2 and LOGICAL*2, respectively. /NOI4 is the default.

/IDENTIFICATION
/NOIDENTIFICATION

Controls whether the FORTRAN–77 compiler identification and version numbers are typed on your terminal. /NOIDENTIFICATION is the default.

/LIST[=listfile]
/NOLIST

Controls whether a listing file is produced.

The compiler, by default, does not create a listing file. If you specify /LIST without a file specification, then the compiler gives a listing file the same file name as the object file, (or, if you specify /NOOBJECT, as the first input source file) and in the same directory on the same device as the object file, but with a .LST file type.

If you give a file specification, the listing is written to that file. The default file type is .LST.

No wildcard characters are allowed in the file specification.

/MACHINE_CODE
/NOMACHINE_CODE
Controls whether the listing produced by the compiler includes the machine language code that the compiler generated.

The default is /NOMACHINE_CODE, which omits machine language code in the listing. The /MACHINE_CODE qualifier is invalid if /LIST is not specified.

/OBJECT[=objfile]
/NOOBJECT
Controls whether the compiler produces an output object file.

By default, or if you specify /OBJECT without a file specification, the compiler produces an object module that has the same file name as the first input source file, in your directory on the public disk structure, with a default file type of .OBJ.

No wildcard characters are allowed in the file specification.

/WARNINGS
/NOWARNINGS
Controls whether the compiler produces diagnostic messages for warning conditions.

Use /NOWARNINGS to override the compiler default, which is to issue warning diagnostic messages.

/WORK_FILES=n
Determines the number of temporary disk work files that should be used during compilation. You can use from one to three files; the default value for n is 2.

If you increase the number of files, the size of the largest program to be compiled can be increased, but you may decrease compilation speed.

FORTRAN/FOR

The FORTRAN/FOR command invokes the FORTRAN–IV compiler.

Format

FORTRAN/FOR filespec[,...]

Qualifiers	Defaults
/CODE=EAE	
EIS	
FIS	
THR	
/[NO]D_LINES	/NOD_LINES
/[NO]I4	/NOI4
/[NO]LINENUMBERS	/LINENUMBERS
/LIST[=listfile]	/NOLIST
/NOLIST	
/[NO]MACHINE_CODE	/NOMACHINE_CODE
/OBJECT[=objfile]	/OBJECT
/NOOBJECT	
/[NO]OPTIMIZE	/OPTIMIZE
/[NO]WARNINGS	/WARNINGS

Command Parameters

filespec [,...]

Specifies one or more source files to be compiled into a single object file. The default file type is .FOR.

Command Qualifiers

/CODE=EAE
/CODE=EIS
/CODE=FIS
/CODE=THR

The /CODE qualifiers select the type of object code to be generated.

The compiler produces distinctly different types of object programs by generating either threaded code or inline code. The default is determined by your system manager at installation time.

The valid values are:

EAE – Selects EAE hardware
EIS – Selects EIS hardware
FIS – Selects EIS and FIS hardware
THR – Selects threaded code

When the compiler produces inline code (/CODE= followed by EAE, EIS, or FIS) the object program executes at greater speed and generally uses less physical memory. Inline code achieves this optimization, in part, by omitting instructions to detect or report certain error conditions. However, you can generate code for error checking by including the /CODE=THR option in the compiler command line.

When threaded code is generated (through the /CODE=THR option), the object program produced uses a symbolic library routine to perform each operation required for program execution. The executable program consists of a "threaded" list of the addresses of library routines and appropriate operand addresses. This type of code generation produces an object module that operates independently of hardware arithmetic configuration. It may be combined with any of the FORTRAN–IV OTS libraries to produce a valid executable program for each type of arithmetic hardware without any need for recompilation.

/D_LINES
/NOD_LINES

Indicates whether the compiler reads and compiles debugging lines. (Debugging lines have a D in column 1 of the source program.) If you specify /D_LINES, lines that have a D in column 1 are compiled.

The default is /NOD_LINES, which means the compiler assumes that lines beginning with a D are comments and does not compile them.

/I4
/NOI4

Controls how the compiler interprets INTEGER and LOGICAL declarations that do not specify a length.

/I4 allocates two words (four bytes) for the default length of integer and logical variables.

By default, or if you specify /NOI4, the compiler interprets these as INTEGER*2 and LOGICAL*2, respectively.

/LINENUMBERS
/NOLINENUMBERS

The /LINENUMBERS qualifier indicates that internal sequence numbers are to be included in the executable program for routine diagnostics.

The /NOLINENUMBERS qualifier suppresses generation of internal sequence numbers.

/LIST = [listfile]
/NOLIST

Controls whether a listing file is produced.

The compiler does not create a listing file by default. If you specify /LIST without a file specification, then the compiler gives a listing file the same file name as the object file. (If you specify /NOOBJECT, then the compiler gives the name of the first input source file.) The file also is in the same directory on the same device as the object file, but with a .LST file type.

If you give a file specification, the default file type is .LST. You cannot use wild-card characters in the file specification.

/MACHINE_CODE
/NOMACHINE_CODE

Controls whether the listing produced by the compiler includes the machine language code that the compiler generated.

The default is /NOMACHINE_CODE, which omits machine language code in the listing. The /MACHINE_CODE qualifier is invalid if /LIST is not specified.

/OBJECT[= objfile]
/NOOBJECT

Controls whether the compiler produces an output object file.

By default, or if you specify /OBJECT without a file specification, the compiler produces an object module that has the same file name as the first input source file, in your directory on the public disk structure, with a default file type of .OBJ.

If you give a file specification, the default file type is .OBJ. You cannot use wild-card characters in the file specification.

/OPTIMIZE
/NOOPTIMIZE

Tells the compiler whether to produce optimized code. The default is /OPTIMIZE.

The /NOOPTIMIZE qualifier should be used during a debugging session to ensure that the debugger has sufficient information to help you locate errors in your source program.

/WARNINGS
/NOWARNINGS

Controls whether the compiler produces diagnostic messages for warning conditions.

Use /NOWARNINGS to override the compiler default. (The default is to issue warning diagnostic messages.)

MACRO

The MACRO command invokes a MACRO–11 assembler. You can include up to six file specifications with the MACRO command.

On RSTS/E you can use either MACRO/RT11 or MACRO/RSX11. The default is MACRO/RSX11 unless your system manager has changed it. The default file type is .MAC with one exception: .MLB is the default file type if you use the RSX–based assembler with the /LIBRARY qualifier.

Format

MACRO/RT11 filespec[,...]

OR

MACRO/RSX11 filespec[,...]

Command Qualifiers	**Defaults**
/LIST[=listfile]	/LIST[=listfile]
/NOLIST	
/OBJECT[=objfile]	/OBJECT[=objfile]
/NOOBJECT	

File Qualifiers

/LIBRARY

Command Parameters

filespec [,...]

Specifies one or more source files to be compiled into a single object file. The default file type is .MAC with one exception: .MLB is the default file type if you use the RSX–based assembler with the /LIBRARY qualifier.

Command Qualifiers

/LIST[=listfile]
/NOLIST

Controls whether a listing file is produced. Unlike the other languages on RSTS/E, MACRO produces a listing file by default. (In other words, the /LIST qualifier is assumed.) The assembler gives a listing file the same file name as the object file and puts it in the same directory on the same device as the object file. If you specify /NOOBJECT, then it assumes the name of the first input source file, and is placed in your directory on the public structure. The default file type is .LST.

No wildcard characters are allowed in the file specification.

/OBJECT[= objfile]
/NOOBJECT

Controls whether the assembler produces an output object module.

By default, or if you specify /OBJECT without a file specification, the assembler produces an object module that has the same file name as the first input source file, on your directory on the public structure. The default file type is .OBJ.

No wildcard characters are allowed in the file specification.

/LIBRARY

Indicates that the file is a macro library.

When a macro call is encountered in the source program, MACRO–11 searches the user macro library for the named macro definitions. If necessary, MACRO–11 then continues the search with the system macro library.

LINK

The LINK command combines one or more of your compiled or assembled object files, including routines from appropriate libraries, into a single executable file.

Note

This section describes the DCL LINK command and its qualifiers. See the *RSTS/E Task Builder Reference Manual* for a complete description of linking concepts, overlays, and libraries.

Format for Simple (Non–Overlaid) Link

LINK file–spec1[,file–spec2,...]

Format for Overlaid Link

LINK /STRUCTURE

.
.
.

(LINK prompts for overlay structure)

Command Qualifiers

Qualifier	Type	Defaults
/BASIC or /BP2 /COBOL or /C81 /DIBOL /DIBOL/DMS /F77 /FOR /RSX11 /RT11	Language Qualifiers	/BP2 (unless your system manager has selected another)
/[NO]DEBUG	Debugging Qualifier	/NODEBUG
/DESCRIPTION	Description Qualifier	
/FMS /NOFMS	Forms Management System	/NOFMS

(continued on next page)

Format for Simple (Non–Overlaid) Link (Cont.)

Qualifier	Type	Defaults
/FMS = RESIDENT /FMS = NORESIDENT /OTS = RESIDENT /OTS = NORESIDENT /[NO]RMS /RMS = RESIDENT /RMS = NORESIDENT	Library Qualifiers	
/EXECUTABLE[= filespec] /NOEXECUTABLE /MAP[= file-spec] /NOMAP	Output File Qualifiers	EXECUTABLE /NOMAP
/STRUCTURE	Overlay Qualifier	(no overlay structure)

Prompt (No Overlays)

Files: file-spec1[,file-spec2,...]

Prompt (Overlays requested with /STRUCTURE qualifier)

ROOT files: file-spec1[,file-spec2,...]
Root COMMON areas: [common-area][,...]
Overlay: [file-spec[,...][+]]

.
.
.

(more prompts for overlays)

Overview

Two linker programs are available on RSTS/E: the RSX–based linker (the Task Builder) and the RT11–based linker (LINK.SAV). The LINK command invokes one or the other, depending on your source language. (You indicate the source language by including a language qualifier.) BASIC–PLUS–2 is the default.

Table 7–3 shows the language qualifiers and whether the RSX–11–based or RT11–based linker is run in each case.

Table 7–3: Relationship Between Language Qualifier, Source Language, and Linker

Qualifier	Source Language	Linker
/BP2 /BASIC	BASIC–PLUS–2	RSX–11–based
/COBOL /C81	COBOL–81	RSX–11–based
/DIBOL	DIBOL	RSX–11–based
/F77	FORTRAN–77	RSX–11–based
/FOR	FORTRAN–IV	RT11–based
/RSX11	MACRO Assembler	RSX–11–based
/RT11	MACRO Assembler	RT11–based

Input File List

The LINK command creates an executable file from one or more of your compiled or assembled object files. You specify the input files, that is, the object files, in one of two places in the LINK command (either beginning on the same line as the LINK command or in response to the prompt), depending on whether you use the /STRUCTURE qualifier to request overlays.

In any case, the standard RSTS/E defaults for file specifications apply. The default device is the public structure (SY:); the device must be disk. The default project-programmer number is your account. The default file type for input files (your object files to be linked) is .OBJ. Note that for COBOL programs, the LINK command expects to find a skeleton file (type .SKL) on the same device and account, with the same file name, as each .OBJ file listed. (A COBOL compile automatically produces .SKL and .OBJ files.)

Simple (Non-Overlaid) Linking

For a simple (nonoverlaid) link, you specify the object files beginning on the same line as the LINK command. If you have many files, you can continue a line by typing a hyphen (–) at the end of a line. If the continued command line is too long*, you get an error message: "?Command too long". The files you name are linked, together with routines from appropriate libraries, in the order you specify. That is, the first file you name occupies the lowest range of addresses in the program, the second file you name is linked to occupy a range of addresses immediately following the first, and so forth.

* The LINK command translates the command you specify into a command line to another program. The translated command cannot exceed 127 characters (80 characters for a COBOL program). If you need to link more files, you must run the Task Builder or the LINK linker directly (see the *RSTS/E Task Builder Reference Manual* or the *RSTS/E RT11 Utilities Manual*).

Overlaid Linking

For overlays, you do not specify the object files on the same line as the LINK command. Instead, you simply specify the /STRUCTURE qualifier (and other qualifiers, if desired). The /STRUCTURE qualifier causes LINK to prompt you for an overlay structure, which is described later in this chapter. Specify the object files in response to these prompts.

Language Qualifiers

The linking process includes routines in the executable file that are obtained from appropriate libraries. You can choose the libraries from which these routines are called. The choice depends on the qualifiers you use with the LINK command, and whether you choose to use the resident libraries installed on your system.

/BASIC
/BP2
Indicates that the object files listed in the command are BASIC–PLUS–2 object files. Unless your system manager has selected another default for your site, /BP2 is the default if you do not specify any language qualifier. (/BASIC and /BP2 are equivalent qualifiers.)

This qualifier provides an easy way to link BASIC–PLUS–2 programs. However, DIGITAL recommends that you use the /DESCRIPTION qualifier instead of /BP2 to link nonoverlaid BASIC–PLUS–2 programs. /DESCRIPTION allows you to use the defaults you specify at installation and with the BASIC–PLUS–2 BUILD command. You cannot use these defaults when you use the /BASIC and /BP2 qualifiers with the LINK command. Instead, you must specify what you would like to use each time you type the LINK command.

A BASIC–PLUS–2 program requires the resident library RMS (Record Management Services) if any OPEN statements include the ORGANIZATION clause or if the FMS (Forms Management System) qualifier is included. See the *PDP–11 BASIC–PLUS–2 Language Reference Manual* for more information.

/COBOL
/C81
Indicates that the skeleton and object files listed in the command are COBOL–81 files. (/COBOL and /C81 are equivalent qualifiers.)

The LINK command includes RMS if the COBOL program requires it.

/DIBOL
Indicates that the object files listed in the command are DIBOL object files.

/DIBOL/DMS
You can link a DIBOL program with either /RMS (Record Management Services) or /DMS (Data Management Services). RMS is the default and recommended I/O service. If you specify /DIBOL/DMS, the LINK command includes DMS instead of RMS in the object file.

/FOR
Indicates that the object files listed in the command are FORTRAN–IV object files.

/F77
Indicates that the object files listed in the command are FORTRAN–77 object files.

With FORTRAN–77, you can use I/O channels 1–14 in your program. The maximum record size for I/O is 512 bytes. See the *PDP–11 FORTRAN User's Guide* if you need a larger record size.

If your program stores a format specification in an array, the size of the specification is limited by the size of an internal buffer, which is 64 bytes. See the *PDP–11 FORTRAN–77 User's Guide* if your program requires more space.

The LINK command always includes RMS with this qualifier. You cannot specify /OTS=RESIDENT with this qualifier. FORTRAN–77 does not have an OTS (Object Time System) resident library.

/RSX11
Indicates that the RSX–11–based assembler (using the MACRO/RSX11 command) on RSTS/E systems (MAC.TSK) produced the object files listed in the command.

The LINK command include RMS only if you specify the /RMS qualifier. /NORMS is the default for MACRO programs.

/RT11
Indicates that the RT11–based assembler (using the MACRO/RT11 command) on RSTS/E systems (MACRO.SAV) produced the object files listed in the command.

The LINK command cannot use RMS with this qualifier.

Forms Management System (FMS) Qualifier

/FMS
/NOFMS
Indicates that you used the capabilities of the DIGITAL Forms Management System (FMS–11) in your program. You can use this qualifier in conjunction with any of the RSX–based language qualifiers. You cannot use it with the /FOR or /RT11 qualifiers.

/FMS=RESIDENT
Specifies the FMS resident libraries FDVRDB and FDVRES. FDVRDB is the FMS resident library that uses debug mode. FDVRES is the resident library without debug mode.

Your system manager can decide to let these libraries belong to a cluster. Cluster libraries are useful if you need to use more than one resident library. When they are clustered, resident libraries share the same virtual address space in your program. (See the *RSTS/E Task Builder Reference Manual* for more information on cluster libraries.)

Without cluster libraries, FMS uses 4K words of virtual address space in your program.

With cluster libraries, the LINK command clusters the FMS resident libraries with all resident libraries being used. However, the LINK command will not use cluster libraries with the /DIBOL qualifier.

/FMS=NORESIDENT
Specifies FMS without a resident library.

Debugging Qualifier

/[NO]DEBUG
Invokes the appropriate debugging tool for a program that is written in COBOL–81, DIBOL, or RSX–11–based MACRO, and/or uses FMS. The debuggers invoked with each qualifier follow. The default for each language is /NODEBUG.

Note that when you use the /DEBUG qualifier with either /C81 or /DIBOL, you must also use the /DEBUG qualifier when you compile the program. If your program consists of a main program and one or more separately compiled subroutines, and you want to use the debugging tool on the entire program, you must compile each subroutine with the /DEBUG qualifier. You can mix sections of your program that have been compiled with /DEBUG with sections that have not been compiled with /DEBUG. However, the debugging tools will be used only in the sections compiled with /DEBUG.

If you use /DEBUG and it cannot be applied, you receive the error message, "?Additional qualifier needed".

For a COBOL–81 program, /DEBUG invokes the COBOL–81 Symbolic Debugger. (See the *COBOL–81 RSTS/E User's Guide* for information on the debugging techniques used.)

For a DIBOL program, /DEBUG invokes the DDT (DIBOL Debugging Technique). (See the *CTS–500 DIBOL User's Guide* for information on the debugging techniques used.)

If FMS is being used, /DEBUG invokes the debug mode of FMS in addition to the other debugging tool. The debug mode of FMS is not available with COBOL–81. (See the *FMS–11/RSTS Reference Software Manual* for information on the debugging techniques used.)

For an RSX–11–based MACRO program, /DEBUG invokes ODT (Octal Debugging Tool). (See the *IAS/RSX–11 ODT Reference Manual* for information on the debugging techniques used.)

For a FORTRAN or RT11–based MACRO program, /DEBUG has no meaning.

Description Qualifier

/DESCRIPTION

Indicates that the input file describes the process for linking the program, including object files to link, the executable and/or map files to create, and various other options. A description file is sometimes called a "task builder command file."

If you program in BASIC–PLUS–2, you can specify /DESCRIPTION if you want to use CMD and ODL Task Builder command files created by the BASIC–PLUS–2 BUILD command. Although /DESCRIPTION and /BP2 can perform the same functions, unlike /BP2, /DESCRIPTION can be tailored to your system and generally results in a smaller program.

You cannot use this qualifier with any of the other LINK qualifiers. The default file type is .CMD. You can specify only one input file with /DESCRIPTION.

Address Space and Library Qualifiers

Two general types of libraries may be available on your system:

- Disk libraries (nonresident)
- Resident libraries

A library is a collection of software modules in a single file. For disk libraries, the Task Builder takes a copy of each routine that you reference in your program and builds it into your program.

Your system manager defines libraries as resident so that they can be shared by more than one user. Resident means residing in computer memory. Instead of building routines into your program (as is done with disk libraries), you share a copy of the library. The copy is resident in memory as long as you or someone else requires it.

Because they are shared, resident libraries occupy less space on the system. Your program uses less memory and processor time and therefore runs faster. However, because it is necessary to dedicate permanent physical memory for resident libraries, your system manager might choose not to install them. By default, the LINK command uses resident libraries if they are present on your system. Otherwise, LINK uses the appropriate disk libraries.

Because both resident and disk libraries take memory from your job, their use affects the amount of memory that is available for your program to work in. There is a 32K–word limit on job space for programs on RSTS/E. Your system manager may also have set a parameter called the SWAP MAX that can limit program size further. SWAP MAX has an upper limit of 32K; however, your system manager can reassign this upper limit to a lower value.

The BASIC–PLUS–2, COBOL–81, and RMS resident libraries can be clustered, which means they share virtual address space in your program. In general, 24K words of address space are available to your program when you use resident libraries that are clustered.

To learn which libraries are available on your system, type SYSTAT/L. Note that the defaults are to use the resident libraries on your system. Therefore, the resident libraries, described as follows, are of interest only if you want to change the defaults.

/OTS=RESIDENT
Specifies the resident library for a language OTS (Object Time System). You can use /OTS=RESIDENT only with BASIC–PLUS–2, COBOL–81, and DIBOL.

/OTS=NORESIDENT
Specifies the disk library for the language you are using.

/RMS
/NORMS
/RMS specifies that RMS will be used and is the default for all RSX–11–based langugages except MACRO. /NORMS is the default for MACRO programs. For more information, see the section on Language Qualifiers earlier in this chapter.

/RMS=RESIDENT
Specifies the RMS resident library, RMSRES.

/RMS=NORESIDENT
Specifies the RMS disk library.

Output File Qualifiers

Two qualifiers, /[NO]EXECUTABLE and /[NO]MAP, control the output files produced by the LINK command.

/EXECUTABLE[=file-spec]
/NOEXECUTABLE
Indicates that you want the LINK command to produce an executable file. /EXECUTABLE is the default.

If you use the /EXECUTABLE qualifier, you can specify a name for the executable file. If you omit the file specification, LINK produces an executable file on the public disk structure (SY:) in your account with a file name the same as the first file name in the input list. The default file type is .TSK (for RSX–based languages) or .SAV (for RT11–based languages).

If you use the /NOEXECUTABLE qualifier, the LINK command does not produce an executable file. You must use the /MAP qualifier if you use /NOEXECUTABLE.

/MAP[=file-spec]
/NOMAP
Use the /MAP qualifier to indicate that you want the LINK command to produce a memory map file. Use the /NOMAP qualifier, or (since /NOMAP is the default) simply omit the qualifier if you do not want a memory map file. You must specify /MAP if you use the /NOEXECUTABLE qualifier.

If you use /MAP, you can also assign a file specification for the memory map file. If you omit the file specification, LINK produces a map file on the same device, with the same account and name as the executable file. The default file type is .MAP.

For RSX–based languages, specifying the /MAP qualifier produces the Task Builder's memory map file. For RT11–based languages, the /MAP qualifier produces the LINK.SAV memory map.

Overlay Qualifier

/STRUCTURE
Indicates that you want to be prompted for an overlay structure (see the following sections).

When You Can Use /STRUCTURE

If your program was written in BASIC–PLUS–2, DIBOL, or FORTRAN–77, you can use the /STRUCTURE qualifier with the LINK command to specify an overlay structure. The /STRUCTURE qualifier presents a dialogue that makes it easy to work with overlays.*

COBOL programmers work with overlays within the language itself, by using the segmentation facility of the COBOL compiler. The *RSTS/E COBOL User's Guide* describes this facility. You do not need to use the /STRUCTURE qualifier if you are doing overlays in COBOL; the LINK command processes the skeleton file's output from the COBOL–81 compiler.

If you program in FORTRAN–IV or in MACRO under the RT11 run-time system, you must run the LINK.SAV program directly to specify an overlay structure. The *RSTS/E RT11 Utilities Manual* describes this program in detail.

When you use the /STRUCTURE qualifier, you are prompted for an overlay structure.

---

* Instead of using LINK and /STRUCTURE, you can specify overlays for these languages using the Overlay Description Language (ODL) recognized by the Task Builder. The *RSTS/E Task Builder Reference Manual* describes ODL. Using the ODL language is more complex than using the /STRUCTURE qualifier with LINK, but it is also more powerful.

When Must You Use Overlays?

If your program is too large to fit in the space available, you must specify an overlay structure for it. The discussion of the language qualifiers tell how much space is available for programs written in the various languages when you use the LINK command.

The easiest way to find out if your program is too large is to try to link it using a language qualifier. If you get the following error message, your program is too large:

```
TASK HAS INVALID MEMORY LIMITS
```

What Are Overlays?

The best way to explain overlays is by example. Suppose your program consists of a main program (called MAIN) and two separately compiled subroutines (called SUB1 and SUB2). Suppose further that MAIN calls both SUB1 and SUB2, and that neither SUB1 nor SUB2 contain any calls to separately compiled subroutines or to MAIN (see Figure 7–1).

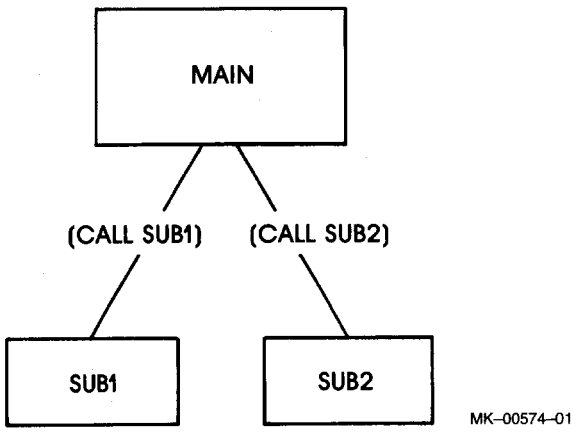


Figure 7–1: Outlining the Call Structure

You can specify an overlay structure such that MAIN is loaded when the program is first run. When MAIN calls SUB1, the loading code built into MAIN during the LINK process loads SUB1 for execution. Then, when control passes back to MAIN and it calls SUB2, SUB2 is brought in to memory overlaying SUB1 (see Figure 7–2).

Note that SUB1 and SUB2 do not call or use data from each other. This logical independence is necessary for program pieces that overlay each other. In this example, calls or references to data that are not currently in memory must be made from the *root*, which is the MAIN program.

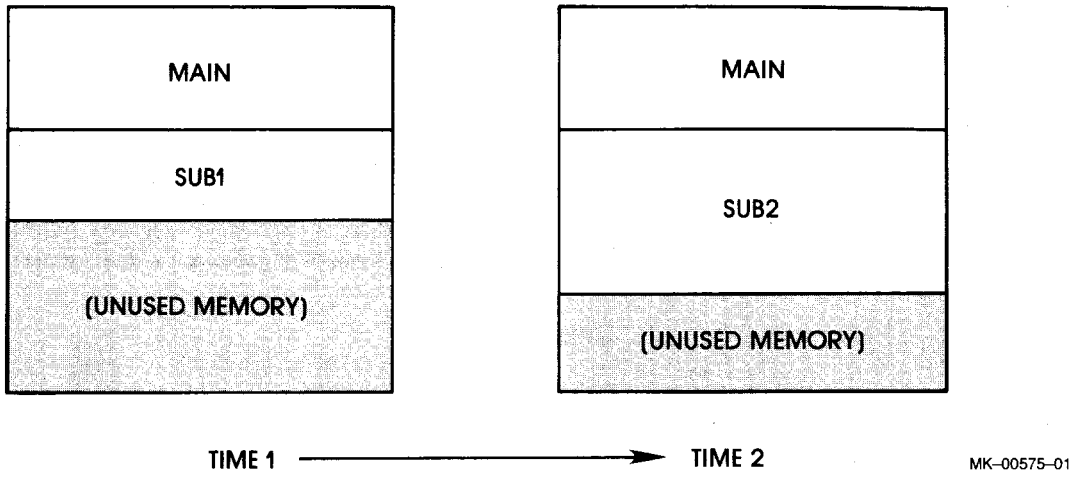


Figure 7–2: A Simple Overlay in Memory

Rules for Constructing Overlays

In general, you must structure an overlay *tree* so that calls or references to data take place between overlay pieces that are on the same *path*. A path is any route from the root of the structure that follows a series of branches to an outermost piece of the tree. Calls or references to data cannot take place between pieces that are along different paths.

Consider the overlay structure shown in Figure 7–3. The structure shows pieces that overlay each other as separate branches of a tree. C and D would start at the same virtual address, as would E, F, and G. The paths in this structure are A–B–C, A–B–D–F, A–B–D–E, and A–B–D–G. Calls may be made between pieces on any of these paths. However, F could not call G, E, or C; C could not call D, E, F, or G, and so forth.

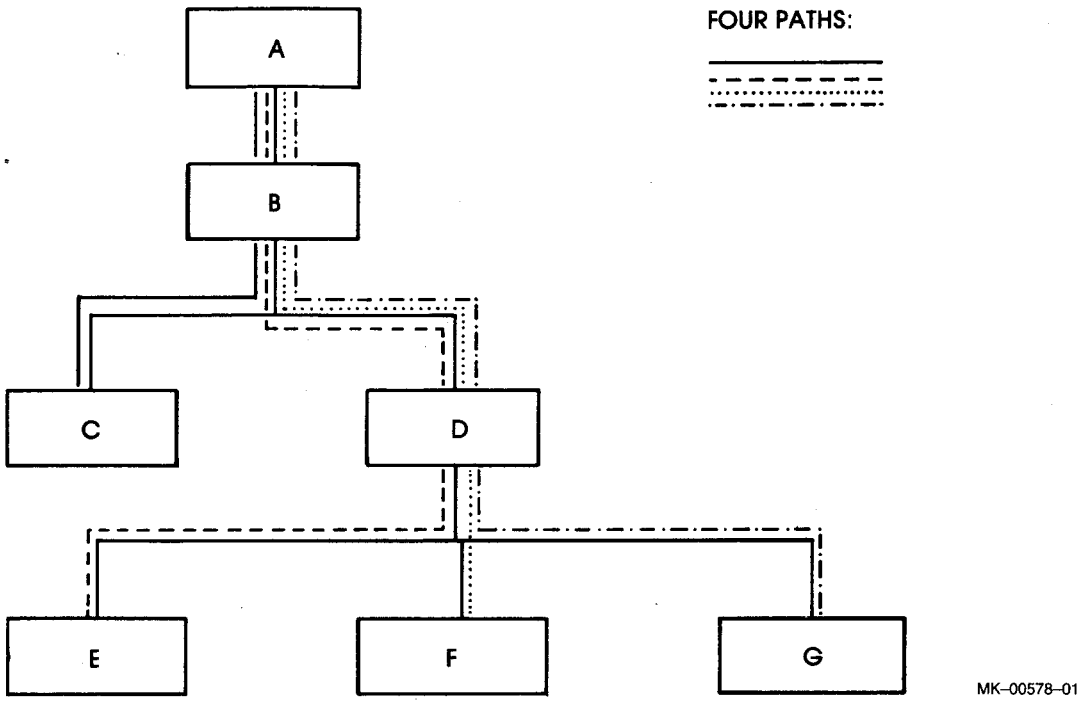


Figure 7–3: Separate Paths in An Overlay Structure

When you use the /STRUCTURE dialogue to specify an overlay structure, you must be careful to specify a structure that does not contain calls or references to data that cross paths.

You must also be careful when you work with common areas in an overlay structure. If a FORTRAN program and subprogram both define a common area A, for example, where is the space for A to be allocated?

Two examples are shown in Figure 7–4. The common block COMA is defined in the pieces A2 and A21. The space for COMA is allocated in A2 because that piece is closer to the root. The common block COMB, however, is defined in A0 and B0. These pieces are not on the same path, so the space for COMB is allocated in both A0 and B0. Note that A0 and B0 cannot communicate through COMB. When the overlay containing B0 is loaded, for example, any data stored in COMB by A0 is lost.

The /STRUCTURE dialogue that follows allows you to place COMB in the root of the overlay structure, where it can be accessed by A0 and B0 properly.

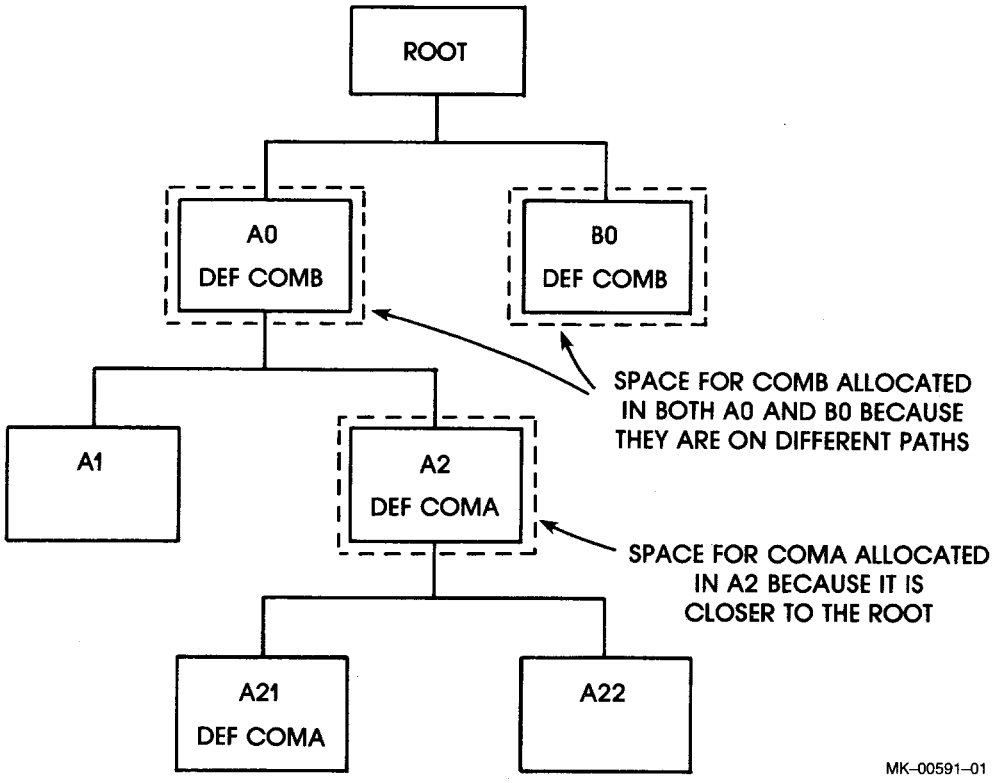


Figure 7–4: Allocating Space for Common Areas

The /STRUCTURE Dialogue

You can use the /STRUCTURE qualifier with /F77, /BASIC, or /BP2 and the /DIBOL qualifier. You cannot use it with the /FOR, /RT11, /COBOL, or /C81 qualifiers.

When you use /STRUCTURE, the LINK command prompts you for an overlay structure, as shown in the following example. This example shows how you would link the overlay structure shown in Figure 7–4.

```
LINK/F77/STRUCTURE
ROOT files: ROOT
Root PSECTS: COMB
Overlay: A0+
  Overlay: A1
  Overlay: A2+
    Overlay: A21
    Overlay: A22
    Overlay:RET
  Overlay:RET
Overlay: B0
Overlay:RET
```

"ROOT files:" Prompt

The first prompt presented is "ROOT files:". You respond to this prompt by typing the object files that are to form the root of the overlay tree. In this case, there is only one such file, named ROOT. If you have more than one such file, type them all on one line and separate them with commas. If you need to continue*, type a hyphen (–) at the end of a line. The LINK command displays "Continue:".

"Root PSECTS:" Prompt

The next prompt presented is "Root COMMON areas:". As a high-level language programmer, you will probably be interested in this question only if you define a common area in two pieces of your program that overlay each other, as in the example in the previous section. When you need to place such a common area in the root of the overlay structure, where it can be accessed properly as a common area, you specify the name of the common area in response to this question.

If you need to place more than one such common area in the root, you simply type the common area names, separated by commas. To continue a line, type a hyphen or a comma at the end of the line. If you type a comma, the LINK command assumes you have finished one PSECT name and the "Continue:" prompt is displayed for you to type more. If you type a hyphen, you can continue typing a PSECT name you began on the previous line.

There is no limit to the number of program sections you can place in the root of your program.

"Overlay:" Prompts

With further /STRUCTURE prompts, you define the overlay structure beyond the root section. You respond to the "Overlay:" prompts with file specifications for object files to be linked. You use two symbols, the plus sign (+) and the comma (,) to show how the files are to be overlaid. You can also use the exclamation point (!) to include comments in the overlay dialogue.

* There is a limit on the number of characters you can type for root files; the translated command line must be less than or equal to 127 characters.

The Plus Sign (+) Symbol

Consider again the LINK command to overlay the structure shown in Figure 7–4:

```
LINK/F77/STRUCTURE
ROOT files: ROOT
Root COMMON areas: COMB
Overlay: A0+
  Overlay: A1
  Overlay: A2+
    Overlay: A21
    Overlay: A22
    Overlay: RET
  Overlay: RET
Overlay: B0
Overlay: RET
```

As shown, you indicate a *branch* in the overlay structure by ending a line with a plus sign following the pieces from which the branch occurs. The next "Overlay:" prompt is indented two places, to show that you are at the next overlay level. For example, A1 and A2 will overlay each other; their addresses will begin immediately after the address taken by A0. Likewise, A21 and A22 will overlay each other after A2.

Repeat this process to obtain the desired overlay structure. When you want to return to the previous level in the overlay structure, you press the RETURN key in response to an "Overlay:" prompt.

Note that overlays can be nested in this fashion to seven levels.

The Comma (,) Symbol

You use the comma (,) operator in an "Overlay:" prompt between files that are to be concatenated in the overlay structure. For example, consider the structure in Figure 7–5.

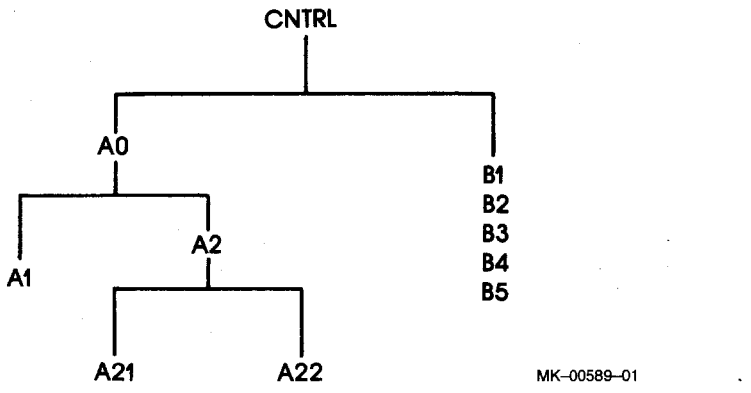


Figure 7–5: Overlay Structure Using Concatenated Files

This structure is similar to the one shown in Figure 7–4, except that the files B1, B2, B3, B4, and B5 are concatenated to form one overlay. Use commas to separate files that are to be concatenated. The following example shows how to use LINK with this structure:

```
LINK /F77/STRUCTURE
ROOT files: CNTRL
Root COMMON areas:RET
Overlay: A0+
  Overlay: A1
  Overlay: A2+
    Overlay: A21
    Overlay: A22
    Overlay:RET
  Overlay:RET
Overlay: B1,B2,B3,B4,B5
Overlay:RET
```

If you end a line with a comma, the LINK command assumes that you want to continue concatenating files on the same branch. A "Continue:" prompt at the same level is displayed for you to continue. For example, the following two lines are equivalent to the "Overlay: B1,B2,B3,B4,B5" line shown above:

```
Overlay: B1,B2,
Continue: B3,B4,B5
```

The Memory Map File

As noted in the section on output file qualifiers, the /MAP qualifier causes the LINK command to produce a memory map file. This file is a detailed listing of the virtual addresses that your program occupies. You can use the /MAP qualifier with any language qualifier. There are two maps available, depending on the language qualifier you use. The RSX–based languages produce a memory map that is described in detail in the *RSTS/E Task Builder Reference Manual.* The RT11–based languages produce a memory map that is described in the *RSTS/E RT11 Utilities Manual.*

You will probably be interested in the memory map only if you are working with overlays and want to examine in some detail what the LINK command has done for you. Figure 7–6 shows the first page of a memory map produced from the following LINK command sequence.

```
LINK /STRUCTURE/MAP=TRY2/EXEC=TRY2
ROOT Files: USER
Root COMMON areas:RET
Overlay: INTRO
Overlay: CRUNCH
Overlay: CHATR
Overlay:RET

TRY2.TSK Memory allocation map TKB 08.006 PAGE 1
                    22-NOV-82 13:42

PARTITION NAME : GEN
IDENTIFICATION : 16R
TASK UIC       : [1,196]
STACK    LIMITS: 001000 001777 001000 00512.
PRG XFR ADDRESS: 021154
TOTAL ADDRESS WINDOWS: 4.
TASK   EXTENSION : 512. WORDS
TASK IMAGE SIZE  : 6048. WORDS
TOTAL TASK SIZE  : 6560. WORDS
TASK ADDRESS LIMITS: 000000 027447
R-W DISK BLK LIMITS: 000002 000036 000035 00029.

TRY2.TSK OVERLAY DESCRIPTION:

BASE    TOP         LENGTH

000000  022127  022130  09304.       USER
022130  023707  001560  00880.            INTRO
022130  022577  000450  00296.            CRUNCH
022130  024317  002170  01144.            CHATR
024320  025613  001274  00700.       R3PUT
025614  027447  001634  00924.       R3UPDA
```

Figure 7–6: Sample from a Memory Map File

As might be determined from Figure 7–6, the memory map file is a bit obscure in terms of what you specify in the LINK command. However, you can determine some basic facts. The size of the program is listed in the line headed "TOTAL TASK SIZE." In this case, the program is 6560 words long.

The section headed TRY2.TSK OVERLAY DESCRIPTION shows the overlay structure constructed by LINK. The first two columns of information give the starting address (BASE) and ending address (TOP) of each major overlay piece in the structure. The next two columns (LENGTH) give the length of each piece in octal and decimal number of bytes, respectively.

The original LINK command dealt with four files: USER, INTRO, CRUNCH, and CHATR. You can see the overlay structure specified by the indentation in the memory map: INTRO, CRUNCH, and CHATR overlay each other following USER.

R3PUT and R3UPDA are overlay pieces from the disk library RMSLIB.OLB, which the LINK command has placed in two special overlay structures called *co-trees*. The *RSTS/E Task Builder Reference Manual* describes co-trees.

The Temporary Files Produced by LINK

When you use the LINK command with RSX–based languages, the LINK command processor creates two files for the Task Builder: a command file and an ODL file. These files tell the Task Builder what to do in creating the executable file and map file.

These files are left in your account. The .TMP file type indicates that they are temporary files. That is, they are deleted from your account when you log out. The command file is the same type of command file you would use with the /DESCRIPTION qualifier. (These temporary files are not created if you use the /DESCRIPTION qualifier.)

RUN

Runs an executable file.

Format

RUN file-spec

Prompts

Program: file-spec

Command Parameters

file-spec

Specifies an executable file.

If you do not specify a file type, the RUN command uses any of the following file types, which correspond to executable files:

.BAC A BASIC–PLUS compiled program. (Refer to the description of the BASIC command for more information.)

.SAV A "save image" file, which runs under the RT11 run-time system. (For example, a FORTRAN–IV executable file is a save image file.) Save image files are produced by LINK/FOR or by LINK/RT11.

.TSK A "task" file, which runs under the RSX run-time system (or a derivative of RSX). See the preceding LINK command description and the *RSTS/E Task Builder Reference Manual* for more information.

Your installation may have a different set of executable file types, depending on the run-time systems your system manager has installed.

If you omit a file type, the RUN command searches the specified directory for a file having each runnable file type. The order of the search depends on your installation. For example, at some installations the RUN command might search for a .TSK file first. If that fails, RUN then searches for a .BAC file, then a .SAV file. However, another installation might search for file types in a different order.

No wildcard characters are allowed in the file specification, and a file name must be included.

Examples of using the RUN command are:

```
$ RUN $SYSTAT(RET)
```

This example runs a system program named SYSTAT. The dollar sign ($ in $SYSTAT) indicates the directory [1,2], which is the system library.

```
$ RUN BANKER(RET)
```

This example runs a user program named BANKER in your account on the public structure.

DCL Command Summary A

This appendix contains a summary of the DCL commands available on RSTS/E.

The commands beginning with a dollar sign (as in $MESSAGE) are commands for use in batch control files.

Table A–1: DCL Command Summary

Command	Description
ALLOCATE	Reserves a device so that only you can use it. Optionally establishes a logical name for a device. (See page 5–8.)
APPEND	Appends data to the end of an existing file. (See page 3–38.)
ASSIGN	Lets you relate a logical name to a physical device or to a PPN. (See page 4–28.)
BASIC	Invokes the BASIC–PLUS or BASIC–PLUS–2 programming environment, depending on the qualifiers you use and the default set by your system manager. (See page 7–7.)
CCL	Allows the use of a CCL command from the DCL environment, when followed by the name of a CCL command. (See page 1–13.)
COBOL	Compiles a COBOL–81 program. (See page 7–8.)
COPY	Duplicates a file. Can also be used to concatenate (merge) files. (See page 3–26.)
CREATE	Allows you to enter text that you can save as a file. (See page 3–3.)
DEALLOCATE	Releases a device from your exclusive use, so that other users may access the device. (See page 5–9.)
DEASSIGN	Cancels logical name assignments made with the ASSIGN or ALLOCATE commands. (See page 4–30.)
DELETE	Deletes files. (See page 3–22.)
DELETE/ENTRY	Uses the sequence number of a job to cancel a request to the line printer or batch queue. (See pages 3–59 and 6–28.)
DELETE/JOB	Cancels a request to the print or batch queue by the job's name. (See pages 3–57 and 6–27.)
DIBOL	Compiles a DIBOL–11 source file. (See page 7–13.)
DIFFERENCES	Lists differences in the text of two files. (The output can be placed in a file or displayed at your terminal.) DIFFERENCES is often used to compare earlier versions of a file with its later versions. (See page 3–61.)

(continued on next page)

Table A–1: DCL Command Summary (Cont.)

Command	Description
DIRECTORY	Displays information about files in a directory. (See page 3–11.)
DISMOUNT	Logically unloads a magnetic tape or disk from a device. (See page 5–15.)
EDIT	Starts the EDT text editor. (See page 3–6.)
FORTRAN	Compiles a FORTRAN program. (See page 7–15.)
HELP	Provides you with DCL command descriptions. (See page 1–8.)
INITIALIZE	Clears ("zeros") a tape for input. (See page 5–13.)
LINK	Links together object files to produce an executable file. (See page 7–24.)
LOGOUT	Exits from RSTS/E and ends a session at the terminal. (See page 1–10.)
MACRO	Assembles a MACRO–11 source file. (See page 7–22.)
MOUNT	Logically loads a magnetic tape or disk onto a device. (See page 5–10.)
PRINT	Prints a "hard copy" of a file on the line printer. (See page 3–41.)
RENAME	Assigns a new name to a file. (See page 3–35.)
REQUEST	Sends a message to the system's operator. (See page 5–17.)
RUN	Executes a system or user program. (See page 7–42.)
SET	Changes characteristics that you specify with an option. (See page 4–3.)
SET HOST	Connects your terminal to a node, called a "host," on the network. (See page 1–23.)
SET PROTECTION	Specifies the protection code of a file to determine who, if anyone, can read, modify, execute, or delete the file. (See page 3–68.)
SET QUEUE/ENTRY	Uses the sequence number of a job to modify the status of a file that is queued for a printer or batch queue. (See pages 3–55 and 6–26.)
SET QUEUE/JOB	Uses the name of a job to modify the status of a file that is queued for a printer or batch queue. (See pages 3–53 and 6–23.)
SET TERMINAL	Changes the characteristics of a terminal. (See page 4–16.)
SHOW	Displays characteristics of the system, your job, your terminal, and the protection code of your files. (See page 4–2.)
SHOW DEVICES	Displays devices in use on your system. (See page 5–18.)
SHOW NETWORK	Displays the available nodes on the network. (See page 1–23.)
SHOW QUEUE	Displays the names of files in a print or batch queue, as well as files that are being processed. (See pages 3–48 and 6–21.)
SHOW SYSTEM	Displays information about the status of all jobs (both attached and detached) in use on the system. (See page 4–10.)
SHOW TERMINAL	Displays the characteristics of a terminal. (See page 4–15.)
SHOW USERS	Displays information about the status of attached jobs on the system. (See page 4–8.)
SUBMIT	Submits a batch job. (See page 6–17.)
TYPE	Displays the contents of a file. (See page 3–19.)

Table A–2: Batch Command Summary

Commands	Description
$DATA	Marks the start of data in a batch job. (See page 6–13.)
$EOD	Marks the end of data in a batch job. (See page 6–13.)
$EOJ	Marks the end of a batch job. (See page 6–13.)
$JOB/DCL	Identifies the beginning of a batch job. (See page 6–11.)
$MESSAGE	Sends a message to the operator from a batch control file. (See page 6–14.)

DCL Error Messages B

The following are error messages that you may encounter when using DCL on RSTS/E. The error messages indicate the cause of a problem in your use of a command or system program.

This appendix is divided into three sections: general system messages, messages that are specific to the use of the LINK command, and error messages for batch processing.

If you perform DECnet/E operations, you may receive messages that are described in DECnet/E documentation, but not in this appendix. In addition, the utility programs that support DCL can produce messages, not all of which are listed here. These messages are listed in the *RSTS/E System User's Guide.*

Note

Angle brackets (< >) surrounding text indicate a place holder for what the system inserts when an error message occurs.

Table B–1: General Error Messages

?Abbreviation too short

You shortened a keyword to fewer than the allowed number of characters.

?Account or device in use

Mounting or dismounting of the device cannot be done because the device is open or has one or more open files. This error can also occur when you attempt to access a non-shared disk from a job other than the one that mounted the disk.

?Additional qualifier needed

You did not include a qualifier required in a command.

?Argument not allowed

You used an argument with a qualifier that does not accept one.

?Bad directory for device

The directory of the device referenced is in an unreadable format. For example, the tape format differs from the format you specified with the MOUNT command.

?Can't find file or account

Either the directory or file does not exist, or you typed a file specification incorrectly.

(continued on next page)

Table B–1: General Error Messages (Cont.)

?Can't mount a private disk as public

This message occurs when a nonprivileged user tries to mount a private disk as public. A nonprivileged user can mount only private disks for his own use.

?Can't run <filename.type> system program

The DCL keyboard monitor cannot execute the command you typed because it cannot run a system program. The next line indicates what error occurred when DCL tried to run the program. See your system manager.

?Command not installed

You typed a DCL command that has not been installed on your system. The DCL keyboard monitor could not find the utility needed to carry out your command. This message can be displayed if you type a remote file specification in a command that allows remote file specifications, but your system does not have DECnet/E.

You can also get this message if you attempt to use the LINK command at an installation where LINK does not support the language you specified.

?Command requires operator privilege

You typed a spooler command that requires privileges, and you are nonprivileged.

?Command too long

The command's length, including continuation lines, exceeds 255 characters. Or, the length of the translated command string exceeds the maximum, which is 127 characters for some commands and 80 for others.

This message occurs with the LINK command in either of two cases.

1. The text you typed in response to the $ prompt and the Files: or Root files: prompt added up to more than 127 characters after translation.
2. With LINK /COBOL, the text was longer than 80 characters after translation.

?Conflicting qualifiers

You used two qualifiers that cannot be combined, or you specified the same qualifier more than once. (Sometimes this condition produces the message "?Syntax error", which covers a greater number of conditions.)

?Data error on device

One or more characters may have been transmitted incorrectly due to a parity error, bad punch combination on a card, or similar error.

?Data error or incorrect density

This message occurs for one of the following reasons:

1. You specified the incorrect density.
2. You did not specify a density, but your system's default density is incorrect for the tape.
3. The data on the tape is damaged.
4. The tape drive is faulty.

To correct the problem, specify the correct density. If that does not solve the problem, try mounting the tape on another drive.

(continued on next page)

?Device does not exist

The device name you specified does not exist on your system.

%Device hung or write locked
%Dismount will proceed as requested

You attempted to dismount a disk that has been physically dismounted. However, the logical dismount will succeed.

?Device hung or write locked

Check the hardware condition of the device requested. Possible causes of this error include a line printer out of paper or a disk drive being offline.

?Device must be disk

You specified a file on a nondisk device, where a disk is required.

?Device not available

The specified device exists on the system, but an attempt to allocate or use it is prohibited for one of the following reasons.

1. The device is currently reserved by another job.
2. The device requires privileges for ownership and you do not have privilege.
3. The device or its controller has been disabled by yoursystem manager.
4. The device is a keyboard line for pseudo keyboard use only.

?Device not file-structured

An attempt was made to access a device, other than a disk drive or magnetic tape drive, as a file-structured device.

?Device not write protected

You specified the /NOWRITE qualifier when you tried to mount a tape, but the device is not write protected. Write protect the device by removing the plastic ring from the tape hub.

?Device offline

You tried to use a tape or disk, but the device is off line.

%Device write protected

The tape or disk is protected against write access. Therefore, you can only read files on the device, but cannot write (perform output) to the device. If you want to write to the device, first write enable the device, next dismount the device, and then mount the device again.

?Device write protected

You requested write access to a device that is write protected. Write enable the device and then retype the command.

?Directory doesn't exist

A file specification indicates a directory that does not exist on the particular disk. Or, a wildcard directory specification failed to produce a match on the disk specified. This error can occur only with disk files.

(continued on next page)

?Disk error during swap

A hardware error occurs when your job is swapped into or out of memory. The contents of the job area are lost, but the job remains logged into the system and returns to DCL. Report such occurrences to your system manager.

%Disk is mounted read-only

This warning occurs if you are nonprivileged and do not specify either read or write access (/WRITE or /NOWRITE) when mounting a disk initialized as read-only. Therefore, you are granted read access only.

?Disk needs rebuilding but you are not privileged

This message occurs when a nonprivileged user tries to mount a private disk that was not logically dismounted. Your system manager can correct the situation by rebuilding the disk.

?Disk pack is locked out

This message occurs when a nonprivileged user attempts to open files on a disk restricted to privileged users.

?Disk pack is not mounted

The DISMOUNT command was attempted, but the disk pack was not mounted on the specified disk drive.

?Do not specify file name or type

This message can occur with the first parameter of the ASSIGN command. Remember to use a space between the first parameter (the string you assign) and the second parameter (the name you assign it). For example, "ASSIGN DR3: X", not "ASSIGN DR3:X".

?Equal sign required

You used a qualifier that requires an argument, but you did not give an argument.

?Error number nn (message text is not on line)

An I/O error occurred while the system was attempting to retrieve an error message. Possible causes could be that the device containing the system error file (ERR.SYS) is off line, or that the system error file contains a bad block.

This is a serious error, and should be reported to your system manager.

?Fatal disk pack mount error

Fatal disk mounting error. The disk is corrupt and cannot be successfully mounted with the MOUNT command.

?Fatal system I/O failure

An I/O error has occurred on the system level. The results of the last command are unpredictable. This error is caused by a hardware condition. Report such occurrences to your system manager.

?File does not exist

This message occurs for one of the following reasons:

1. An input file that must be present is not present.
2. A wildcard file specification does not match any files.
3. A file is not accessible because of its assigned protection code.

(continued on next page)

?File name needed

A file specification you typed does not include a file name, but one is needed.

?File specification needed

This error occurs if you specify node:: without a file specification (except with DIRECTORY). It also occurs if you have two commas with nothing between them in a file specification list.

?Files cannot be on different nodes

This message occurs with network operations. Each input file specification you include in a network command must be on the same node.

?Form does not exist

The form name you specified was not defined by your system manager. See your system manager to find out what form names are available.

?Forms Definition File does not exist

The forms definition file, [1,2]FORMS.SYS, could not be located. You may have accidentally deleted it.

%ID label ignored

You mounted a tape in DOS format and specified an ID label. Identification labels are not encoded on DOS tapes; therefore, the label you specified is not recognized by the MOUNT command.

%ID label should be specified when you mount an ANSI tape

You mounted a tape in ANSI format, but did not specify the tape's identification label. It is recommended that you specify the identification label, so the MOUNT command can verify that the tape you selected has been mounted.

?ID labels don't match

The identification label you specified does not match the identification label encoded on the tape. To correct the problem, specify the correct identification label. If you do not know what identification label is encoded on the tape, you can omit the ID label from the MOUNT command.

?Illegal switch usage

A CCL command contains an error in an otherwise valid CCL switch (qualifier). For example, the /SI:n switch was used without a value for n or a colon; or more than one of the same type of CCL switch was specified.

?Impossible density for this device

You tried to mount or initialize a tape, but specified a density not available on the tape drive you used. To correct the problem, use another drive or specify a different density.

?Incorrect density

You specified a density different from the tape's density. This message appears only if you are using a tape drive that determines the tape's correct density. You do not need to specify the tape's density with the MOUNT command.

(continued on next page)

Table B–1: General Error Messages (Cont.)

?Incorrect density or uninitialized tape

This message can occur for the following reasons:

1. You tried to mount a tape that has not been initialized.
2. You specified an incorrect density.
3. You did not specify a density, and your system's default density is incorrect for the tape.

If the tape has not been initialized, use the INITIALIZE command. Otherwise, specify the correct density.

?Invalid argument

This message can occur when you use a qualifier in the form /qualifier=argument. The qualifier you specified is spelled properly, and the equal sign (=) or colon (:) is present, but the argument is either missing or syntactically invalid.

This error message is displayed when there is not a more specific message to describe the syntax error.

?Invalid CCL command

You used the CCL prefix followed by a command that is not installed as a CCL command on your system.

?Invalid character

You typed an invalid punctuation character.

?Invalid command

The command name you gave is not a DCL command and is not defined on your system as a CCL command. Or, the line begins with a punctuation character rather than with a keyword.

?Invalid date

A date either has improper syntax, represents a nonexistent date (like 30–Feb), or represents a date before 1970 or after 1999.

?Invalid density – n

You tried to mount or initialize a tape, but specified a density other than 800 or 1600 bpi. The "n" is the density you specified.

?Invalid file specification

A local file specification has improper syntax.

?Invalid form definition

The definition of the specified form contains an invalid keyword or keyword argument.

?Invalid form name

The form name you specified is either longer than six characters or consists of one or more nonalphanumeric characters.

?Invalid job name

The job-name you specified does not consist of alphanumeric characters or is too long. Job-names can be one to six characters for the large Spooler and one to nine characters for the small Spooler.

(continued on next page)

Table B–1: General Error Messages (Cont.)

?Invalid keyword

The keyword is not recognized. This error occurs with keywords that are not qualifiers and are not command names. (For example, it can occur with the options of SET and SHOW and with qualifier values that are keywords.)

?Invalid PPN

The project-programmer number (PPN) you specified does not have valid syntax.

?Invalid print device

The device you specified is not a valid print device because it is not a line printer or terminal.

?Invalid qualifier

The qualifier keyword is not valid in the command you typed. (This message may indicate an error in spelling or typing.)

?Invalid qualifier for disk

This message occurs when you specify an invalid qualifier for disks. For example, /FORMAT=ANSI (applies to tapes).

?Invalid qualifier for tape

This message occurs when you specify an invalid qualifier for tapes. For example, /PRIVATE (applies to disks).

?Invalid queue name

Either the queue name you typed does not consist of alphanumeric characters or there is no such queue.

The only valid queue name for the small Spooler is PRINT:.

?Invalid time

A time either has improper syntax or represents a nonexistent time (like 25:00 or 13:00PM).

?Invalid with network file specification

You gave a network file specification in one of the commands that accepts them (RENAME, COPY, and so on), but you also gave a qualifier that can only be used with local operations.

?I/O to detached keyboard

This message can result from two actions:

1. You tried to perform I/O with a terminal line that is used for dial-up terminals, but nobody was dialed in.
2. Your job became detached (perhaps because you were dialed in and your line was later hung up) and then tried to perform I/O with the terminal. The second situation either causes the job to hibernate or causes this error condition, after which the job hibernates. You see this message when you subsequently attach to the job.

?Job #<job-number> does not exist

The job-number you typed was not found in the specified queue.

?Job <job-specification> does not exist

No jobs were found that matched the job-specification you indicated.

(continued on next page)

Table B–1: General Error Messages (Cont.)

?Job name needed

The command you typed requires a job-name.

?Keyword needed

You typed nonalphanumeric characters when a keyword is needed instead. (If you type alphanumeric characters without a valid keyword, you receive the message, "?Invalid keyword".)

%Logical name has not been assigned

This warning occurs when a nonprivileged user specifies a logical name. The mount succeeds, but the logical name has not been assigned.

?Magtape record length error

When performing input from magnetic tape, the record on tape was found to be longer than the buffer designated to handle the record.

?Magtape select error

When access to a magnetic tape drive was attempted, the selected unit was found to be off line. This message can occur when you transfer data to or from a tape.

?Maximum memory exceeded

This is a nonrecoverable RSTS/E message caused by the following conditions:

1. During an OLD operation, the job's private memory size maximum was reached.
2. While running a program, the system required more memory for string or I/O buffer space, and the job's private memory size maximum or the system maximum (16K words for BASIC–PLUS) was reached.

?Missing closing bracket

This error can occur in a local or remote file specification. There is a left square bracket ([) or left angle bracket (<), but no right square bracket (]) or right angle bracket (>).

?Missing closing quote

A quotation mark (") is not matched with another quotation mark. (This message can occur in remote file specifications.)

?Missing parameter

You were prompted for a required command parameter and you pressed the RETURN key instead of giving the parameter.

?Name or account now exists

You attempted to COPY or RENAME to an existing file. This message can occur with RENAME if you do not specify /REPLACE and the output file already exists. It can also occur with COPY if you specify /NOREPLACE and the output file already exists.

?No buffer space available

The system is overloaded and cannot complete your command because small buffers are currently unavailable. Try the command again later.

?No logins

This message can be displayed when you try to log in, for one of two reasons:

1. The system is full, so it cannot accept additional users.
2. Your system manager has disabled logins. (Possibly, logins are disabled because the system will be shut down shortly.)

(continued on next page)

?Non-executable file

This message occurs if the file you are trying to run is a source file, for example, .BAS. You need to compile and link the file before you run it. (Note that an executable file includes the value 64 in its protection code.)

?Non-printable character

You typed a control character.

?Non-res run-time system

This message generally indicates hardware problems. The following are examples:

1. The run-time system referenced has not been loaded into memory (and cannot be loaded for some reason), and is therefore nonresident. Report this error to your system manager.
2. With the BASIC or SWITCH command, the run-time system you are trying to switch into is not currently available, although it is installed on the system.
3. With the RUN command, the program you are trying to run requires a run-time system that is nonresident.
4. At other times, your job keyboard monitor became unavailable (perhaps because the disk it was on malfunctioned). The error message is displayed, and then control is returned to the default keyboard monitor. If the default keyboard monitor is also unavailable, control is returned to the primary run-time system.

?/NO prefix not allowed

You used the /NO prefix improperly.

?No room for Spooling package

There are currently no RSTS/E job slots available, so the Spooler cannot be started.

?No room for user on device

In attempting to create a file on a device, or write information to a device, you exceeded some size limit. If it was a nondisk device, this error indicates that the device was full. If it was a disk device, this error indicates one of three conditions:

1. The disk as a whole is full.
2. You attempted to create a contiguous file, and there is not enough space on the disk. (However, the /CONTIGUOUS qualifier for the COPY and CREATE commands produce only a warning if there is not enough contiguous space.)

   If you attempted to create a contiguous file but were unable to, try to create a noncontiguous file of the same size. If you are able to create a noncontiguous file, then you can conclude the problem is lack of contiguous space.
3. If you have eliminated the first two conditions, then the disk directory has reached its capacity. This capacity is independent of your disk quota. The number varies, depending on the directory cluster size your system manager assigned when creating the directory and the sizes of the files in it. Large files fill up a directory faster than small files, especially if the large files have small cluster sizes. Also, if the directory itself has a large cluster size, it can hold more and larger files.

(continued on next page)

Table B–1: General Error Messages (Cont.)

?No run-time system

This error can occur if the program you are trying to run requires a run-time system that is not installed.

?Not a queueable device — <queue-name> :

You specified a queue-name that is valid syntactically, but does not represent a queue that exists on your system.

?Not a valid device

The device name that you gave is invalid for any of the following reasons:

1. It is not assigned as a system-wide or user-logical name.
2. It is not a physical device name of any device that is installed on your system.
3. The device name is valid, but not with this command. For example, "INITIALIZE TT:" (you cannot initialize a terminal).

?Not enough available memory

An attempt was made to load a nonprivileged executable program that is too large to run, given the job's private memory size maximum. Either the program must be made privileged to allow it to expand above a private memory size maximum, or your system manager must increase the job's private memory size maximum to accommodate the program.

?Not your own PPN

You are nonprivileged and specified an account that was not your own.

?Number too big

You typed a number where one is allowed, but the number is too large. Refer to the command description to find out the largest acceptable value.

?Number too small

You typed a number where one is allowed, but the number is too small. Refer to the command description to find out the smallest acceptable value.

?Pack-id labels don't match

The identification code for the specified disk pack does not match the identification code already on the pack. This message can occur when you try to mount or dismount a disk.

?Parameter or argument too long

This message occurs if a file specification, text string, or qualifier argument exceeds 127 characters.

?PPN does not exist

The project-programmer number (PPN) you specified as part of the job-specification does not exist.

?PPN needed

The command you typed requires that a PPN be specified.

?Printer already initialized

The device you specified has been already initialized as a spooling device.

(continued on next page)

Table B–1: General Error Messages (Cont.)

?Printer busy

The command you typed could not be completed because the spooling device you specified is currently processing a job.

?Printer not initialized

The device you specified has not been initialized as a spooling device.

?Program failure in <program-name>

This message reports a problem in the software. It is followed on the next line by an explanation of the problem. Your system manager should verify that the failing program is correctly installed. If necessary, he should then submit an SPR. The SPR should show the dialogue that preceeded the message, the exact text of the message, and a list of patches that have been installed in the failing program.

?Protection violation

This error can occur for reasons similar to the following:

1. You typed a CCL command that your system manager has protected against general use.
2. You tried to run a program to which you do not have execute access.
3. You tried to read a file to which you do not have read access.
4. You tried to write to or delete a file to which you do not have write access.

If this message occurs because of a protection violation with one of your own files, you can use the SET PROTECTION command to change the file's protection code. (However, this error can also occur because of other conditions.)

?Qualifier conflicts with parameter

You typed a parameter and a qualifier that should not be present together. For example, with the DEASSIGN command, you specified the /ALL qualifier and a logical name.

Queue file being reorganized – please wait...

The queue file is currently being reorganized. The command you typed will proceed when the reorganization is complete.

?Queue file does not exist

The queue file could not be located. Have your system manager restart the Spooling package to create the queue file.

?Reorganization still not complete – please try again later

The queue reorganization process did not complete within a reasonable amount of time. Have your system manager submit an SPR if this warning occurs frequently.

?Specify a map or executable file

With LINK, you specified /NOEXECUTABLE and did not specify a map file.

?Spooling package already started

The Spooling package has been already started.

?Spooling package not started

The command you typed cannot be processed until the Spooling package is started.

(continued on next page)

Table B–1: General Error Messages (Cont.)

?Stack overflow

This message indicates a system problem. Have your system manager send in an SPR, giving the dialogue that preceded the message, the text of the message, and a list of patches that have been installed.

?Syntax error

The command has improper syntax. This occurs when there is not a more specific message describing the syntax error.

?Too many arguments

You used the notation /qualifier=(arg,arg,...) with a qualifier that accepts only a single argument.

?Too many items in list

In a list of file specifications or other items (separated by commas or plus signs), you indicated more file specifications than are allowed. For example, you exceeded one of the following limits:

1. The DIBOL, RENAME, DELETE, and SET PROTECTION commands allow six file specifications.
2. The COPY command allows six input file specifications and one output file specification.
3. The PRINT and SUBMIT commands allow up to 11 file specifications.

?Too many logical names assigned

With the ASSIGN command, you exceeded the maximum number of logical names. You can only assign up to four logical names (only three logical names if any of the logical assignments includes a PPN).

?Too many open files on unit

You specified the same magnetic tape or DECtape drive both as input and output files with the COPY or APPEND commands.

?Too many parameters

This message occurs if you specify more command parameters than the command can accept.

?Too many printers initialized

The device you specified could not be initialized because the maximum number of spooling devices has been already initialized.

?Unit number needed

This message occurs when you specify a device-name without a device-number. For example, DM: instead of DM0:.

?Wildcard job name not allowed

The command you typed does not permit wildcard characters in the job-name.

?Wildcard PPN not allowed

The command you typed does not allow wildcard characters in the PPN.

?Wildcard queue name not allowed

The command you typed does not allow wildcard characters in the queue-name.

(continued on next page)

Table B–1: General Error Messages (Cont.)

?Wildcards not allowed

You included a wildcard in a file specification, where wildcards are not allowed.

?You must be privileged to dismount a public disk

This message occurs when a nonprivileged user tries to dismount a disk initialized as public. A nonprivileged user can dismount only private disks.

?You must be privileged to mount a public disk

This message occurs when a nonprivileged user tries to mount a disk initialized as public. A nonprivileged user can mount only private disks.

?You must be privileged to rebuild the disk

This message occurs when a nonprivileged user attempts to rebuild a private disk. If you are nonprivileged, have your system manager rebuild the disk.

The following messages are specific to the use of the LINK command. However, this is not a complete list. If you do an RSX–based link, refer to the *RSTS/E Task Builder Reference Manual* for other messages. In addition, if you use LINK/COBOL, refer to the COBOL documentation for messages.

If you do an RT11–based link, refer to the *RSTS/E RT11 Utilities Manual* for other messages.

Table B–2: LINK Error Messages

Task Builder messages for RSX–based link

?ALLOCATION FAILURE ON FILE file-specification

There is not enough disk space to store the executable file, or you do not have write access to the account or disk that was to contain the file.

?FILE file-specification HAS INVALID FORMAT

You tried to link an RT11–based object file, but specified an RSX–based language (or an RSX–based language was the default). Or, you tried to link something that was not an object file (such as a source file).

?LOOKUP FAILURE ON FILE file-specification

This message occurs in one of two cases:

1. One of the input files you specified was deleted while the link was in progress (for example, from another terminal).
2. One of the system files required for the particular language you are linking under is not present.

In either case, verify that all of the input files exist. If they do, then contact your system manager.

(continued on next page)

Table B–2: LINK Error Messages (Cont.)

?MODULE file-name MULTIPLY DEFINES SYMBOL sym-name

Two input files on the same path either contain routines or define global variables that have the same name.

?MODULE file-name MULTIPLY DEFINES XFR ADDR IN SEG segment-name

There are two or more main programs linked into the root.

?Null PSECT name

You typed a null program section name. For example, you typed two commas with nothing between them in response to the "Root COMMON areas:" prompt.

?OPEN FAILURE ON FILE file-specification

You specified an input file to which you do not have read access. Or, a system file needed by the LINK command may be protected against read access. Check the protection codes of the input files you specified. If you have read access to all of them, contact your system manager.

?Overlay tree has too many levels

You were already seven levels deep, and your answer to the Overlay: prompt included a trailing plus sign.

%Resident library not installed

You attempted to link against a resident library that was not installed.

If your system manager later installs the library, your program will run. If you try to run the program without the library being installed, you will get the message, "?Can't find file or account", because the library cannot be found.

?TASK HAS INVALID MEMORY LIMITS

Your program is too big. Check the map. Overlay it more heavily.

?TASK REQUIRES MORE THAN 8 WINDOW BLOCKS

Your program is too big.

?n UNDEFINED SYMBOLS SEGMENT seg-name

This message is caused by one of the following:

1. You specified the wrong language on the LINK command line.
2. You allowed LINK to default to the wrong language.
3. You are trying to link object files that were compiled under different languages.
4. One of the object files references an external routine or variable that is not present in any of the other object files.

(continued on next page)

Table B–2: LINK Error Messages (Cont.)

LINK.SAV messages for RT11–based link

?LINK–F–Invalid record type in <file-spec>

The object file is in the wrong format; it may have been produced by an RSX–based compiler. (The notation <file-spec> represents a file specification that is supplied in the message.) See the *RSTS/E RT11 Utilities Manual* for more information.

?LINK–W–Multiple definition of symbol

Two input files either contain routines or define global variables that have the same name.

?LINK–W–Transfer address undefined or in overlay

You did not include a main program.

?LINK–W–Undefined globals:

You specified the wrong language in the LINK command line, or allowed LINK to default to the wrong language. Or, one of the object files references an external routine or variable that is not present in any of the other object files.

The following messages are generated by the batch facility.

Table B–3: BATCH Error Messages

?Batch being shut down

The batch processor is going off line and the job is terminated.

?Cannot increase priority

A /PRIORITY:n qualifier appeared in the $JOB/DCL command. The user was privileged, but specified a value for n greater than 127. Or, the user was nonprivileged and specified a value greater than –8.

?Cannot use that account

An account parameter appeared on the $JOB/DCL command, but the request did not come from a privileged user.

?Do not specify a queue name with /BATCH

You specified the /BATCH qualifier and included a queue-name in the job specification. You may specify either /BATCH or a queue-name, but not both.

?Invalid specification field

The parameter (specification) given in a $JOB or $MESSAGE statement is in the wrong format.

?Invalid switch

The qualifier (switch) used in the command is either invalid, in the wrong format, or privileged.

?No batch jobs possible at this time

The batch processor requires a pseudo keyboard to execute a job, but no pseudo keyboard is available. Use the SUBMIT command to reenter the job.

(continued on next page)

Table B–3: BATCH Error Messages (Cont.)

?No such account

The account specified in the $JOB/DCL command does not exist.

?Time limit exceeded

The time specified in the $JOB command is insufficient to execute the job. Specify a larger limit by using the /LIMIT=nnn or the /NOLIMIT qualifier.

?Unable to log in batch job

To execute a request, the batch processor logs a job into the system using the account under which the job was queued or the account specified in the $JOB command. For some reason, the log-in procedure failed, for example, logins are disabled. The job will be requeued for later execution.

?Unmatched parentheses

An opening (left) parenthesis appears in a command, but an accompanying closing (right) parenthesis is not found.

?Unmatched quotation marks

Quotation marks (") and apostrophes (') must be paired in a control statement.

Differences Between DCL on RSTS/E and VMS

In general, RSTS/E DCL is a subset of VMS DCL. This means that most DCL commands on RSTS/E work and have the same effect on VMS.

This appendix lists exceptions to the general rule.

Table C–1: Differences Between DCL on RSTS/E and VMS (Cont.)

Condition	Solution
Defaults for qualifiers are sometimes different.	Be explicit. For example, specify /LIST or /NOLIST, depending on whether you want a listing. Do not rely on the defaults.
Error messages are often different.	No solution.
Displays with the SHOW and DIRECTORY commands are different.	No solution.
RSTS/E uses numeric protection codes. VMS uses symbolic protection codes.	No solution.
RSTS/E does not use controller letters to specify device names, for example, DB0: and DB2:. VMS uses controller letters, such as, DBA0: and DBB0:.	Use logical names.
RSTS/E uses the name SY: to designate the public structure. VMS does not have a public structure, but the system disk has the name SYS$SYSTEM:.	No solution.
On RSTS/E, directories and users are identified by PPN (project-programmer number), which is a combination of decimal numbers. On VMS, they are identified either by name or by UIC (user identification code), which is a combination of octal numbers.	No solution.
On RSTS/E, even if a logical name includes a PPN, you can override that PPN. Therefore, you can specify, "LB:[1,2]BARBSH.PRO", where LB: is equated with "DR2:[2,5]". You cannot do this on VMS.	Do not override PPNs on RSTS/E.
RSTS/E does not let you specify version numbers for files. VMS lets you specify version numbers.	Do not specify version numbers. Always work with the highest-numbered version. An exception is the DELETE command, where VMS requires you to specify a version number and RSTS/E does not allow one.

(continued on next page)

Table C–1: Differences Between DCL on RSTS/E and VMS (Cont.)

Condition	Solution
RSTS/E does not require quotation marks in remote file specifications. VMS requires you to enclose remote file specifications in quotation marks if they do not conform to VMS file specification syntax. For example, RSTS::"NOTICE.TXT".	Use quotation marks around remote file specifications.
RSTS/E has fixed defaults for file specifications. VMS has temporary defaults for file specifications.	Do not rely on the defaults. Explicitly specify a device and directory for each file specification.

RSTS/E and VMS prompt for command parameters, if you omit the parameters, but not in the same way. For example,
On RSTS/E:
`$ ALLOCATE`
`Device: LP0`
On VMS:
`$ ALLOCATE`
`$_Device: LPA0`
`$_Log_name: PRINTER`
	Type the command on a single line to avoid using the prompts.

RSTS/E has the /STRUCTURE qualifier for the LINK command. VMS does not have this qualifier.	No solution.
RSTS/E has the SHOW USERS command. VMS does not have this command.	Use the SHOW SYSTEM command instead.
RSTS/E has the DELETE/JOB and SET QUEUE/JOB commands. VMS does not have these commands.	Use the DELETE/ENTRY and SET QUEUE /ENTRY commands instead.
For batch streams, RSTS/E has the $DATA and $MESSAGE commands;VMS does not. Both systems have the $JOB command, but it has a different meaning on each system.	Use $REQUEST instead of $MESSAGE, and do not use $DATA. There is no solution for $JOB.
In batch streams, RSTS/E treats data lines as commands. VMS flushes extra data lines that do not begin with a dollar sign ($) when a command is needed.	Do not put extra data lines in a batch stream. Make sure that each program uses all of the data intended.
On RSTS/E, if you type SET PROTECTION /DEFAULT with no arguments, your job default protection code is returned to the system-wide default (normally 60). On VMS, this command has no effect.	No solution.
Queue names are different on RSTS/E. This affects the PRINT, SUBMIT, DELETE/ENTRY, and SET QUEUE/ENTRY commands. RSTS/E uses names like BA0: for batch queues and LP1: for printer queues. VMS uses names like SYS$BATCH and SYS$PRINT.	Use the default queues on both systems to avoid having to specify queue names. This helps with PRINT and SUBMIT, but not with DELETE /ENTRY, where VMS requires you to specify a queue name and does not have a default.

(continued on next page)

Table C–1: Differences Between DCL on RSTS/E and VMS

Condition	Solution
On RSTS/E, the SHOW QUEUE, DELETE /ENTRY, and SET QUEUE/ENTRY commands default to the LP0: line printer queue. On VMS they have no defaults.	No solution.
RSTS/E accepts AM/PM times with the /BEFORE, /SINCE, and /AFTER qualifiers. VMS does not accept these formats.	Use 24-hour notation. For example, 13:00 rather than 1:00PM.
When you use DECnet to access a file on another node, RSTS/E prompts for accounting information for that node. On VMS, you must supply the accounting information as part of the file specification.	Supply the accounting information as part of the file specification. For example, node "user password account"::filespec.
The COPY command on RSTS/E queries you before replacing an existing file. The COPY command on VMS does not do this.	Specify /REPLACE or /NOREPLACE to avoid the prompt.
RSTS/E commands for file manipulation accept the /QUERY qualifier that lets you select files to be affected. The corresponding qualifier on VMS is /CONFIRM.	No solution.
RSTS/E may accept shorter keyword abbreviations than VMS, or vice versa.	Spell out keywords fully. If you abbreviate them, use at least the first four letters. Do not count the prefix NO as part of these letters. If you specify underscores (_) when you use abbreviations, do not apply them to the four letter minimum.
The BASIC command on RSTS/E accepts the /BPLUS and /BP2 qualifiers. VMS does not accept them.	Use the /BP2 default.
On RSTS/E, you can prefix a DCL command with the keyword "DCL". You cannot do this on VMS.	Do not use the DCL prefix.
RSTS/E has CCL (Concise Command Language) commands. VMS does not have CCL commands.	Do not use CCL commands.
The RSTS/E REQUEST command accepts quotation marks, but it does not require them. On VMS, the parameter of the REQUEST command must be enclosed in quotation marks if the command contains spaces or special characters.	Use quotation marks.
On RSTS/E, the INITIALIZE command for tapes allocates the device to your job. The INITIALIZE command does not do this on VMS.	Always use the ALLOCATE command to allocate a device before you initialize it.

Glossary

Absolute Time — a specific date or time provided in a command line.

Account — what you log in to with a Project-Programmer Number (PPN) and a password on a RSTS/E system.

Account Number — a number that identifies the project group and the programmer. It is also known as the Project-Programmer Number (PPN, such as [3,25]). You need an account number (which new users receive from the system manager) to log in to the system. The system uses the account to charge you for computer time and arrange, find, and protect your stored files.

Alphanumeric — a contraction of alphabetic-numeric; the set of characters that compose text. Characters include letters and numerals, and exclude special characters.

ANSI — American National Standards Institute. Creates standards to ensure consistency in all aspects of computer technology: command languages, keyboards, codes, and so forth.

ASCII Code — American Standard Code for Information Interchange, a standard 7–bit code representing the 128 characters in which textual information is recorded.

Batch Processing — a way of scheduling programs that allows you to enter jobs to be run without your input at a terminal. Batch processing can be used, for example, in data processing operations that need no interaction.

Baud Rate — the speed at which information is transferred between devices on a system.

Bif — contraction of "binary digit." A bit is the smallest unit of information in a binary system of notation.

Block — a set of consecutive machine words, characters, or digits handled as a unit, particularly in input and output operations. On the RSTS/E system, a block is equal to 256 16–bit words or 512 8–bit bytes.

BPI — bits per inch. A measure of tape density. Common densities for magnetic tapes are 800 or 1600 BPI.

Buffer — a temporary storage area used to contain data. Buffers hold data being passed between processes or devices that operate at different speeds or times.

Byte — a group of eight binary digits (bits) processed as a unit. Each ASCII character is stored in one byte.

Carriage Return — a keyboard operation that causes the terminal print head or cursor to return to the left margin; usually combined with a line feed to allow input from a terminal to start a new line.

Central Processing Unit — the portion of a computer system that controls the interpretation and execution of instructions. Also called CPU.

Channel — a logical concept that helps you manage the flow of data to and from a running program.

Character — an element in a set of symbols. A human readable symbol can be a letter from A to Z, a number from 0 to 9, or a special symbol. A machine intelligible symbol is made of a group of binary digits. Some machine-readable symbols are "nonprinting" characters (such as tabs or control characters), because they cannot be displayed on a line printer listing or a video screen.

Cluster Size — the unit of size in which files are allocated. Cluster size is measured in blocks.

Command — a user-initiated order to a computer system that causes the performance of a predefined operation. The part of an instruction that specifies the operation to be performed.

Command Environment — an interface through which you can communicate with the computer. DCL and BASIC–PLUS are examples of command environments.

Command Level — the part of a session at a terminal when the system awaits your input. For example, you can recognize DCL command level by its prompting symbol ($).

Command Line/Command String — a line (or set of continued lines), normally terminated by pressing the RETURN key. A command line or string contains a command and (optionally) information modifying the command. The full form of the command string contains a command, its qualifiers, and its parameters (file specifications, for example) and their qualifiers.

Compile — to translate a source program (one written by a user) into machine language.

Concatenate — to unite in a series; to link together many items into one.

Concise Command Language (CCL) — a way to call a system program and provide it with input on a single command line. Many system programs (such as PIP on RSTS/E), use the name of the program as the command name, followed by the command line required by that program. CCLs can be different for each RSTS/E system.

CPU — see Central Processing Unit.

Crash — an unexpected computer system shut-down, caused by problems such as power failures.

Creation Date — the date on which a file on RSTS/E is created or updated.

CRT — Cathode Ray Tube. Used in video display terminals, such as VT52s and VT100s. (Sometimes video terminals are referred to as "CRT devices.")

CTRL (Control Key) — the keyboard character that causes a control action. It is used in combination with an alphabetic key. For example, if you hold down the control key and press the "U" key on a RSTS/E system, the system responds by deleting characters from the cursor to the beginning of the line.

DCL — DIGITAL Command Language.

DECnet — a family of hardware/software products that creates distributed networks from DIGITAL computers and their interconnecting devices. DECnet allows you to do such operations as copying files from one computer (node) to another, and connecting your terminal to a remote computer.

Default — to omit information in a computer operation and let the computer make a predefined assumption about the operation; the assumption made by a program when you do not provide a value.

Density — the number of bits per inch that can be stored on a magnetic tape.

Device — a peripheral hardware unit used to perform input and output operations.

Dial-up Line — a communications circuit that connects a terminal and a computer over a telephone line.

Disk — a mass storage device that holds data on rotating, electromagnetic platters.

Disk Quota — the capacity of a disk to store information. Each user of a computer system is allowed a finite amount of storage space. On RSTS/E, the system notifies you if you try to exceed your disk quota.

Display — the printing of text at a hard-copy terminal, or the output of text at a video screen.

DOS — Disk Operating System, used primarily to refer to a tape format.

Echo — the display of characters you type at a terminal. When you log in on a RSTS/E system, for example, generally the PPN is echoed, but the password is not.

Editing Session — the time you spend at a terminal using a text editor.

Editor — a program that allows you to modify data, programs, or files.

Elapsed Time — the amount of time you are logged in to a computer system.

Error Message — a notice from the computer that indicates an error and usually contains recovery information; a program message indicating the presence of a mistake. For example, RSTS/E displays an error message if you type in data that it cannot process.

Field — one or more characters treated as a unit; a specified area of a record used for a single type of data. For example, a file name is occasionally referred to as the "name field" of a complete file specification.

File — an ordered collection of stored text or data; a collection of related information. For example, you could create files on RSTS/E to contain data for a program or text for a report.

File Specification — the name used to identify the node, device, directory name, file name, and file type under which a file is stored.

File Type — the part of a file specification that follows the file name. A file type (sometimes called an "extension" on RSTS/E) consists of a period (.) followed by up to three alphanumeric characters.

Hard Copy — computer output printed on paper, such as from a line printer or a hard-copy terminal.

Hardware — the physical equipment or machinery of a computer system. (Devices are examples of hardware.) Compare with Software.

Help Facility — a set of on-line messages that describes the functions of available programs on a system.

Host — the computer system that processes your commands and responds to them. The network node that you are currently logged in to.

Input — information that is transmitted to a computer; information to be processed by the computer; a device involved in gathering data. For example, when you type in a program at a terminal that is connected to the computer, you are providing input. The terminal is, among other things, serving as an input device.

Installation — the location of a computer system. For example, this manual refers to your installation as the computer system at your site.

Interactive — the process of communication between a person and a computer system. For example, the computer prompts you for input and you type in a response.

Interface — a shared boundary between two elements of a computer system, such as between two programs or devices.

Job — everything a user does from beginning to end of a terminal session: commands, programs, processing, output. Also, a pending request to print files or to run a batch job.

Job Number — the number RSTS/E assigns to your job when you log in. Your job number is deleted when you log out.

Journaling — the recording of user input during an editing session. For example, if you edit a file using EDT and the system crashes, journaling allows you to recover your work when the system resumes operation.

Keyboard — the set of keys that you use to type at a terminal to provide input to the computer.

Keyboard Monitor — a part of the system software that provides communication between a person at a keyboard and the operating system. The keyboard monitor interprets the commands you type at the keyboard.

Keyword — a word that is an integral part of a command or qualifier.

Line Feed — the keyboard operation that shifts from one line position to the same horizontal position on the next line. Often the line feed is combined with the carriage return to start a new line.

Link — to combine or connect two independent entities; a reference to another element in a set of elements. Generally used to mean combining two or more object files into an executable file.

Literal — characters that are interpreted literally. For example, in BASIC–PLUS programming, you can enclose a character string in quotation marks so that the characters are used as literals.

Local Node — the node a user first logs in to on the system; generally used in discussions of network operations.

Logical Name — a name that a user or system manager can assign to a physical device or to a PPN.

Log In — the process of gaining access to a computer system; an identification procedure.

Log Out — the process of ending a session with the computer.

Magnetic Tape — plastic tape coated with magnetic material on which data is recorded in narrow tracks.

Memory — internal computer storage. Also, a device on which data can be stored and from which it can be accessed.

Merge — in files, to join the contents of one file to the contents of another file. (See also Concatenate.)

Microsecond — one millionth of a second.

Millisecond — one thousandth of a second.

Mnemonic — an aid to memory, as in mnemonic file names, where the name of the file reminds you of the contents of the file.

Mode — a state of being, such as "interactive mode," in which the system or a program prompts you for your input.

Network — a set of interconnected computer systems.

Node — an individual computer system in a network that can communicate with other computer systems in the network.

Non-File-Structured Device — a device that does not separate data into distinct files and does not maintain directories.

Nonprivileged Account — an account that allows you to work with your own files, but that has limited access to system-wide operations.

Null — an element that is explicitly left blank. For example, a null file type consists of no alphanumeric characters. (In other words, the file specification includes a file type, which is null.)

Operating System — software that controls the execution of computer programs and performs system functions. An integrated collection of programs that supervise computer operations.

Output — the device involved in transferring data from memory; the data created as the result of transfer; processed data. For example, when your terminal displays the contents of a file, the terminal serves as an output device.

Paper Tape — a long, narrow strip of paper on which holes are punched in several rows. You can use tape for input, using the paper tape reader; for output, using the paper tape punch; and for offline storage.

Parameter — the object of a command. A parameter can be a file specification or a keyword option.

Parity Bit — a binary digit appended to a group of bits to make the sum of all the bits always odd (odd parity) or always even (even parity). (Parity pertains to the way data is transmitted between the system and its devices.) Used to verify data storage.

Password — the character string assigned to each user that verifies user access privileges to a system; a way of assuring file confidentiality on time-sharing systems. You need a PPN and a password to log in to a RSTS/E system.

Physical Name — a name for a peripheral device; identifies the drive, control circuit, and medium together. Compare to Logical Name.

Priority — the level of importance assigned to an operation in computing. For example, you can set the priority of a print request so that your job is printed before or after other jobs in the print queue.

Privileged Account — an account that allows you access to more system-wide operations than a nonprivileged account.

Programming — the process of planning, writing, testing, and correcting the steps required for a computer to solve a problem or perform a desired operation.

Project-Programmer Number — see Account Number.

Prompt — a symbol indicating that the system is prepared for input from a user.

Protection Code — a feature that allows you to decide who can have access to your files.

Pseudo Keyboard — a part of RSTS/E software, primarily used in batch jobs. The pseudo keyboard operates like a real terminal, except that the "typist" is a program rather than a person.

Qualifier — a command language keyword that modifies the operation of a command. Qualifiers are always preceded by slash (/) characters. (On some systems, a qualifier is referred to as a "switch.")

Queue — a line-up of jobs to be performed in sequence, as in a batch or print queue, (in which certain jobs may be given priority over others).

Read — a process that performs input from a file or a device.

Read-Only — the state a device or file is in when it can be used for reading data from a file or device, but not for writing.

Real Time — synonym for time as we know it, in 24–hour days.

Rebuilding — a process that corrects corrupt file structures on a disk (previously called "cleaning"). This is a privileged operation and applies to disks only.

Remote Node — any node in the network other than the local node. That is, any node except the one you logged in to.

RETURN Key — a key at the terminal that ends a line that you type and inputs it to the computer. See also Carriage Return.

RSTS/E — an acronym for Resource-Sharing Timesharing System/Extended, a DIGITAL operating system.

Run Burst — the length of time, measured in ticks, that a job is allowed to run uninterrupted in the CPU of a computer.

Run Time — the time in which a program is executed; the actual amount of CPU time required for a program to complete execution.

Run–Time System — system software that manages a programming language. The number of run-time systems can differ from one system to another.

Software — the set of programs that controls the operation of a computer system.

SPR — Software Problem Report. Submitted by a user (generally the system manager) to report a possible problem in the software.

Swapping — the transfer of a job from memory to disk or vice versa. Lets the system run more jobs than its memory can hold at any one time.

Switch — see Qualifier.

Syntax — the rules governing the structure of statements in a computer language; the structure of a language.

System — a combination of hardware and software that performs processing operations; a collection of components that forms a functional unit.

System Manager — the person or group of people who schedule the access and use of a computer system.

Temporary File — a file that is deleted from your account when you log out.

Terminal — a device, consisting of a keyboard and display mechanism, used to enter and receive data to and from a computer. LA36s (hardcopy) and VT100s (video) are examples of terminals.

Tick — a fraction of a time slice. (A tick is about 1/60 of a second.)

Timesharing — a method of computer operation in which more than one user shares the system at the same time. The central theory of timesharing is that the system processes each user's job in slices of time so rapidly that the user is often unaware that processing time is shared.

Time Slice — the amount of time (in fractions of a second) that the computer assigns to each user's job during timesharing.

Translator — software that converts human-readable programs into machine-readable code.

Utility — a program or set of programs that performs a set of commonly used functions, such as copying or deleting files. The PIP program and the EDT editor are examples of utilities on RSTS/E.

Wildcard — a symbol that allows you to refer to more than one file with a single file specification. The asterisk (*) is a commonly-used wildcard symbol in DCL, as in *.DAT to specify all names of data files.

Word — a unit of storage in the computer. A word holds 16 bits, or two ASCII characters, of information.

Write — a process that performs output to a file or a device.

Write-Locked — a protection feature that does not permit you to write data into a file or device. (Same as read-only.)

Write Protected — same as write-locked.

Index

A

C

D

F

G

H

I

J

K

L

M

N

Q

R

S

T

U

V

W

X

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA and Puerto Rico call **800–258–1710**

In Canada call **800–267–6146**

In New Hampshire, Alaska or Hawaii call **603–884–6660**

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
A&SG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

*Any prepaid order from Puerto Rico must be placed with the Local Digital Subsidiary:
809–754–7575

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number. _____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

Do Not Tear - Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: Commercial Engineering Publications MK01-2/E06
RSTS/E Documentation
DIGITAL EQUIPMENT CORPORATION
CONTINENTAL BOULEVARD
MERRIMACK, N.H. 03054

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line