RSTS/E
System Manager's Guide

Order No. AA–2762E–TC

March 1983

This document describes how to manage a RSTS/E system.

OPERATING SYSTEM AND VERSION:	RSTS/E	V8.0
SOFTWARE VERSION:	RSTS/E	V8.0

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

Contents

Chapter 7 System Utility Operations

Chapter 10 Updating RSTS/E Software

Chapter 11 DCL Commands for Disk and Tape Handling

Chapter 12 DCL Commands for Using the Micro–RSTS Spooler

Appendix A Auxiliary System Program Files

Appendix B Number Conversion

Appendix C Manual Patching with ONLPAT and CPATCH

Appendix D DCL Error Messages

Appendix E Disk Device Sizes

Figures

Tables

Preface

This guide tells you how to operate and manage a RSTS/E system. You should be familiar with the structure and programming of RSTS/E and know time-sharing software and hardware. The basis for the manual is a program set with privileged status and privileged features. Only the RSTS/E system manager and privileged users should have access to this guide.

For more information on RSTS/E guides and manuals, consult the *RSTS/E Documentation Directory*.

This manual uses the following conventions:

(RET)	Indicates pressing the RETURN key. Unless otherwise noted, user input is terminated by pressing RETURN.
(LF)	Indicates pressing the LINE FEED key.
(CTRL/X)	Indicates holding down the CONTROL key and pressing another key, as in the CTRL/C combination.
`color`	A contrasting color in examples indicates user input.

Summary of Technical Changes

The *RSTS/E System Manager's Guide* is revised for Version 8.0. Known errors are fixed and several descriptions are revised. There are changes to update the manual for the hardware and software that RSTS/E currently supports. Wherever possible, examples are revised for DCL.

In addition, the following paragraphs list the significant changes for RSTS/E.

File Structures (Sections 1.3.2 through 1.3.5)

RSTS/E now supports two kinds of file structures. Sections have been added to describe the structures and to explain the differences between the two.

REACT (Sections 4.1 through 4.1.3)

The ENTER function of the REACT program is expanded to allow you to locate the User File Directory (UFD) at a specific device cluster on the disk. You can also preallocate space for the UFD, and check to see if a directory is contiguous, both of which allow you to optimize the disk.

MONEY (Section 4.2)

The MONEY program now lets you obtain data using wildcard accounts, in addition to specific project-programmer numbers, or an entire device. The dialogue has changed to eliminate some questions. Minor changes have been made to the format of headings in MONEY's printed report. In addition, you can create a CCL command to allow a single line format for the MONEY command.

STALL and UNSTALL (Section 7.1.2.5)

The new STALL and UNSTALL qualifiers of the UTILTY command let you temporarily suspend system operations.

DSKCVT (Section 7.1.3.3)

The new DSKCVT program lets you convert disks to Version 8.0 (RDS1) format.

Changing Quotas and Passwords (Section 7.1.3.7)

RSTS/E now enforces disk quotas you set for user accounts on any disk.

EMT Logging (Section 7.2)

The EMT logging feature allows you to record system activities. This section provides an overview of how to use EMT logging and why it can be useful to your site.

DSKINT (Section 7.5)

The DSKINT program is completely rewritten for Version 8.0 to allow you to initialize a disk on-line, whether or not the disk was previously initialized. DSKINT writes patterns to the disk and checks them. If DSKINT finds any bad blocks, they are noted in the "bad block file" (BADB.SYS) so they will not be used to store data. Formerly, this operation could be done only by the DSKINT option of INIT.SYS, requiring that the system be shut down.

Updating RSTS/E Software (Chapter 10)

This new chapter consists of two parts. Part I provides background information about updating software and emphasizes the use of automated patching from update kits. Part I also describes the use of prebuilt tasks and replacement modules, which are new for Version 8.0.

Part II begins with a table that lists programs for updating RSTS/E software. The sections following the table describe each program in more detail and give examples of their use.

The description of manual patching is now in Appendix C.

DCL (Chapters 11 and 12)

New and expanded Digital Command Language (DCL) commands are now available to the system manager. Chapter 11 describes disk and tape handling:

- The INITIALIZE program exercises a disk and creates a RSTS/E file structure on it. Like the new DSKINT program, INITIALIZE writes patterns to the disk, checks for bad blocks, and adds any bad blocks it finds to the file BADB.SYS.

- MOUNT now automatically rebuilds a dirty or corrupt disk (one that has been physically dismounted without being logically dismounted with a DISMOUNT command).

Chapter 12 describes the commands available to manage the new micro-RSTS line printer spooling package, also known as the "small spooler," available with Version 8.0.

The small spooler requires only one job, unlike the existing spooler described in Chapter 5, which uses OPSER. However, the small spooler handles only print spooling, not BATCH processing. New DCL commands let you start up and shut down the small spooler, start printing on a particular device with a specified form, stop printing on a device, and delete jobs from the queue.

System managers can choose to install one of two versions of DCL: one for the small spooling package and the other for the existing spooler. The choice is determined at system generation; see the *RSTS/E System Generation Manual* for DCL installation procedures.

DCL Error Messages (Appendix D)

This new appendix describes DCL error messages for privileged and non-privileged users.

Disk Device Sizes (Appendix E)

This new appendix lists device sizes and device cluster sizes for each disk that RSTS/E supports.

Chapter 1
RSTS/E System Structure and System Management

RSTS/E runs on a PDP–11 computer and allows simultaneous, time-shared access to PDP–11 hardware and to RSTS/E software components through either local or remote asynchronous terminals. The *RSTS/E Software Product Description*, which is part of the RSTS/E documentation kit, describes the hardware and software RSTS/E supports. This manual contains the description of the software a system manager uses to control the operation of a RSTS/E system. The following sections introduce you to some important RSTS/E concepts.

1.1 Glossary of RSTS/E Terms

To understand more fully how RSTS/E works, you need to know these terms:

CCL (Concise Command Language)

A shorthand way to run a RSTS/E system program, such as UTILTY. The CCL syntax allows you to run a program without the RUN command and, unlike the RUN command, allows you to place the entire command on one line. After the program finishes executing, control returns to your job keyboard monitor. The system manager chooses the CCL commands for a particular RSTS/E system. (Refer to Chapter 7.)

DCL (DIGITAL Command Language)

A set of commands available on many different DIGITAL systems. These commands perform basic tasks like copying files, printing files, and running programs. On RSTS/E, the DCL command environment is managed by the DCL run-time system, which has a keyboard monitor like BASIC-PLUS. (Refer to the *RSTS/E DCL User's Guide* for comprehensive information about DCL on RSTS/E. See Chapter 11 of this manual for DCL commands for system management.)

Default Keyboard Monitor

The main keyboard monitor that you work in on a RSTS/E system. You enter the default keyboard monitor after you log in. The system manager chooses the default keyboard monitor for a particular system.

Job

The unit that RSTS/E uses to keep track of you (and other users) during a terminal session. When you log in, the system creates a job for you and assigns it a job number. The system uses your job number to keep track of everything you do from the beginning to the end of your terminal session.

Job Keyboard Monitor

The keyboard monitor that manages a job. Your job keyboard monitor is the same as the default keyboard monitor, unless you change it with the SWITCH program. After you change your job keyboard monitor, you remain under its control until you log out or use SWITCH again to change your keyboard monitor.

Keyboard Monitor

The part of a run-time system with which you communicate. When you work in the DIGITAL Command Language (DCL) environment, for example, you type commands that the DCL keyboard monitor receives and interprets. Each RSTS/E keyboard monitor has an identifying "prompt" that it displays to indicate when it expects command input. Common keyboard monitor prompts on RSTS/E are: dollar sign ($) for DCL, "Ready" for BASIC-PLUS, angle bracket (>) for RSX, and dot (.) for RT11.

Monitor

The master control system software that observes, supervises, controls or verifies the operation of a computer system. The collection of routines that controls the operation of user and system programs, schedules operations, allocates resources, and performs I/O.

Operating System

The collection of programs, including a monitor or executive and system programs, that organizes a central processor and peripheral devices into a working unit for the development and execution of application programs.

Primary Run-Time System

The run-time system that is permanently resident and that was installed when you established the defaults for your current monitor with the DEFAULT option of INIT. At system start-up, the primary run-time system also serves as your default keyboard monitor. Once under timesharing, you can use the DEFAULT KBM command in

UTILTY to change your system default keyboard monitor if you want users to issue commands from a keyboard monitor other than that of your primary run-time system.

Program Development

The process of writing, entering, translating, and debugging source programs.

Public Structure

The set of all disks that are public. When you do not include a device name in your file specification, the system by default accesses one of the disks on the public structure. The logical SY: represents the name for all disks in the public structure. Thus, if you do not have any public disks other than the system disk, SY0: and SY: are equivalent. If you have more than the system disk in the public structure, SY: refers to the aggregate of all public disks.

Run-Time System

System software that manages part of the RSTS/E system. For example, the BASIC-PLUS run-time system manages the BASIC-PLUS programming environment.

System Disk

The disk that is required by the RSTS/E monitor to start the system and thereafter to allow the system to run properly under timesharing. The system-wide logical SY0: is assigned to the system disk.

System Program

Any general-purpose program included in an operating system to perform common functions.

1.2 System Management

Management of RSTS/E begins with providing properly tailored hardware and software configuration, proceeds through initializing the software at system generation time, and continues with the daily functioning of timesharing. To make sure you perform these steps efficiently, you should be familiar with time-sharing concepts and practices or have a close working relationship with a senior programmer or analyst who is experienced with time-sharing systems.

To begin managing a system well, you must have knowledge of the processing requirements of the system and know the capabilities and structure of the system. Information in the *RSTS/E System Generation Manual* describes important aspects of hardware and software options and provides memory requirements to assist in configuring a RSTS/E system.

After you generate your RSTS/E system, many initialization options are available to structure and control system elements. A few of the initialization options are complex and their use at system generation time is important. Once timesharing begins, certain restructuring capabilities become more difficult to employ. You, as the system manager, should be particularly careful about initializing your system for the first time. However, no prepackaged approach can give you a properly initialized RSTS/E system. The *RSTS/E System Generation Manual* provides detailed information about all the initialization options.

To ensure important information reaches the right persons within your organization, you should make someone responsible for current documentation of the system. This responsibility includes both locally generated and DIGITAL-supplied procedures and guides. The responsible person should make sure that delegated members of the staff receive the most current system information. In too many cases, improper use of resources occurs when responsible individuals are not aware of the latest information.

1.3 Disk Organization

The organization of files on disks is important for system managers to understand. It is with this knowledge that they can make the best use of all system resources. This section introduces you to the concepts of disk and file structures and how to use them to best advantage.

1.3.1 Types of Disks

RSTS/E supports two types of disks as part of its logical disk structure: private and public. The public disk structure consists of a system disk and additional public disk packs or disk cartridges. The system disk must be physically on-line and logically mounted whenever the system is running; this is the only way users can gain access to the system disk during time-sharing operations. All other public disks that users may need to access should also be physically on-line and logically mounted.

The system disk contains:

- Monitor code
- Initialization code
- The primary run-time system
- The system start-up program
- The control files for system start-up and crash recovery

Some installations may also use the system disk for storing active user jobs that are temporarily swapped out of memory.

Remaining space on the system disk and other public disks is used for:

- Auxiliary run-time systems
- Run-time system files

- Optional system files
- System programs

This space is also available for storing user programs and data files.

Any disk drives not devoted to the public structure can be used for private disk packs or disk cartridges. Unlike public disks, you can physically and logically mount private disks and move them to other drives during time-sharing operations. In addition, private disks make it possible to restrict disk storage to a defined set of users, a distinct advantage when numerous projects are in progress. The file structure of a private disk is the same as that of a public disk.

1.3.2 File Structures

The RSTS/E file structures allow the monitor to access system and user data in an organized manner. RSTS/E currently supports two kinds of file structures:

1. A structure comprised of an MFD (Master File Directory) and a UFD (User File Directory). This structure, known as RDS0 (for RSTS/E Disk Structure 0), is for RSTS/E disks created before Version 8.0.

   RDS0 is supported only so you can use disks initialized prior to Version 8.0.

2. A structure comprised of an MFD, a GFD (Group File Directory), and a UFD. As of Version 8.0, you can initialize disks only in this second format. This structure is known as RDS1.

You can position and preextend a UFD in either structure to optimize the use of system resources. (See the description of the REACT program in Chapter 4 for the techniques.)

The following three sections describe the two kinds of structures. If you have a new RSTS/E Version 8.0 system with all new disks, you can skip Section 1.3.3. Furthermore, if you have disks that were created before Version 8.0, DIGITAL recommends that you use the new DSKCVT disk conversion program to convert them to the new disk structure. (The DSKCVT program is described in Chapter 7.)

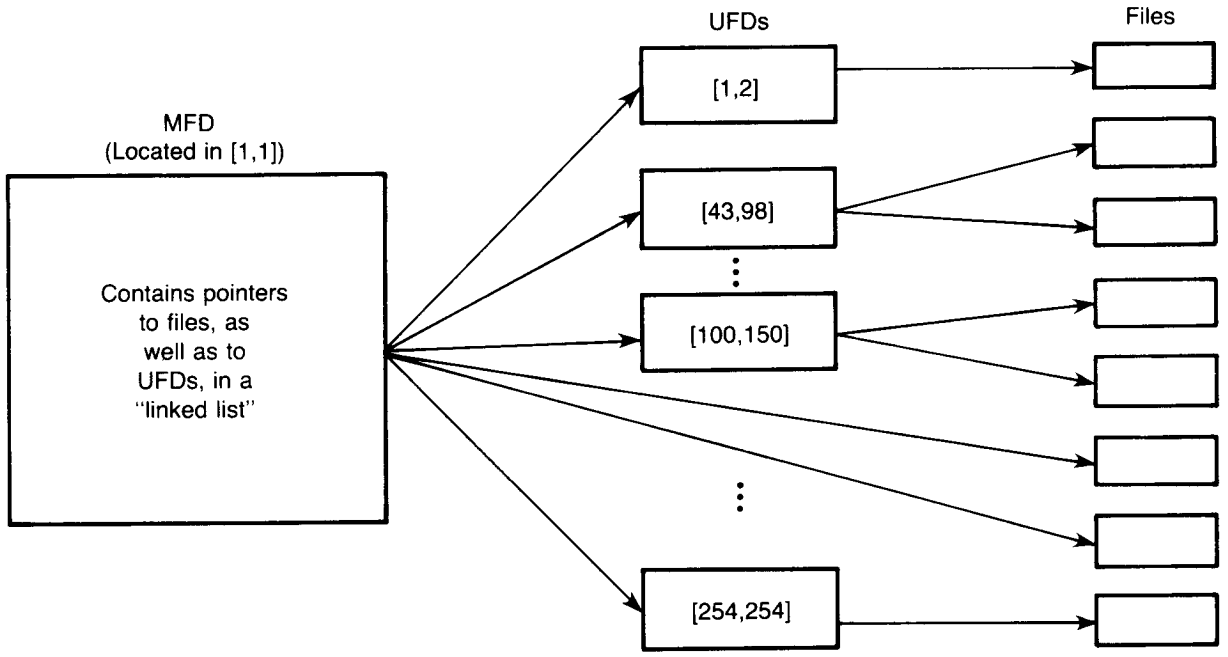
To see which structure applies to disks on your system, type SY/D as shown. The "Level" column indicates whether a disk is in RDS0 (0.0) or RSD1 (1.1) format.

```
$ SY/D (RET)

Disk Structure:
Dsk Open     Size        Free    Clu  Err  Name     Level  Comments
DR1   66   131648    29424  22%    4    0  BOWER     1.1   Pub, DLW
DR2    0   242576    33040  13%    8    0  CRONIN    0.0   Pri, R-O, DLW
DR3   33   500352    57744  11%    8    0  WOJTAS    1.1   Pri, DLW
DR4    0   242572    17528   7%    4    0  MODNE     0.0   Pri
DR5    0   500352    76152  15%    8    0  HOGAN     0.0   Pri, R-O, DLW

$
```

1.3.3 RDS0 Disk Structure (Prior to Version 8.0)

Before Version 8.0, users gained access to files on a RSTS/E system by an MFD and by UFDs. These structures are shown in Figure 1–1.

Figure 1–1: MFDs and UFDs in RDS0 File Structure



In the structure shown in Figure 1–1, each disk you initialize for use on a RSTS/E system contains an MFD, located in [1,1]. The system uses the MFD to catalogue other accounts on the disk. The MFD on the system disk lists the accounts that can be used to log in to the system. On a private disk, the MFD contain entries of accounts that can create files on that disk. Any user can access any file on any disk if the protection code of the file permits. Only users whose accounts are entered in the MFD on a private disk can create files on that disk.

The system creates one UFD for each user account on a disk when the manager sets up the account (or optionally when a user creates the first file for the account). The UFD catalogues all program and data files under an account and maintains accounting and access information for these files. The UFD contains all retrieval information for the files because each file is pure data and has no linkage or structural information.

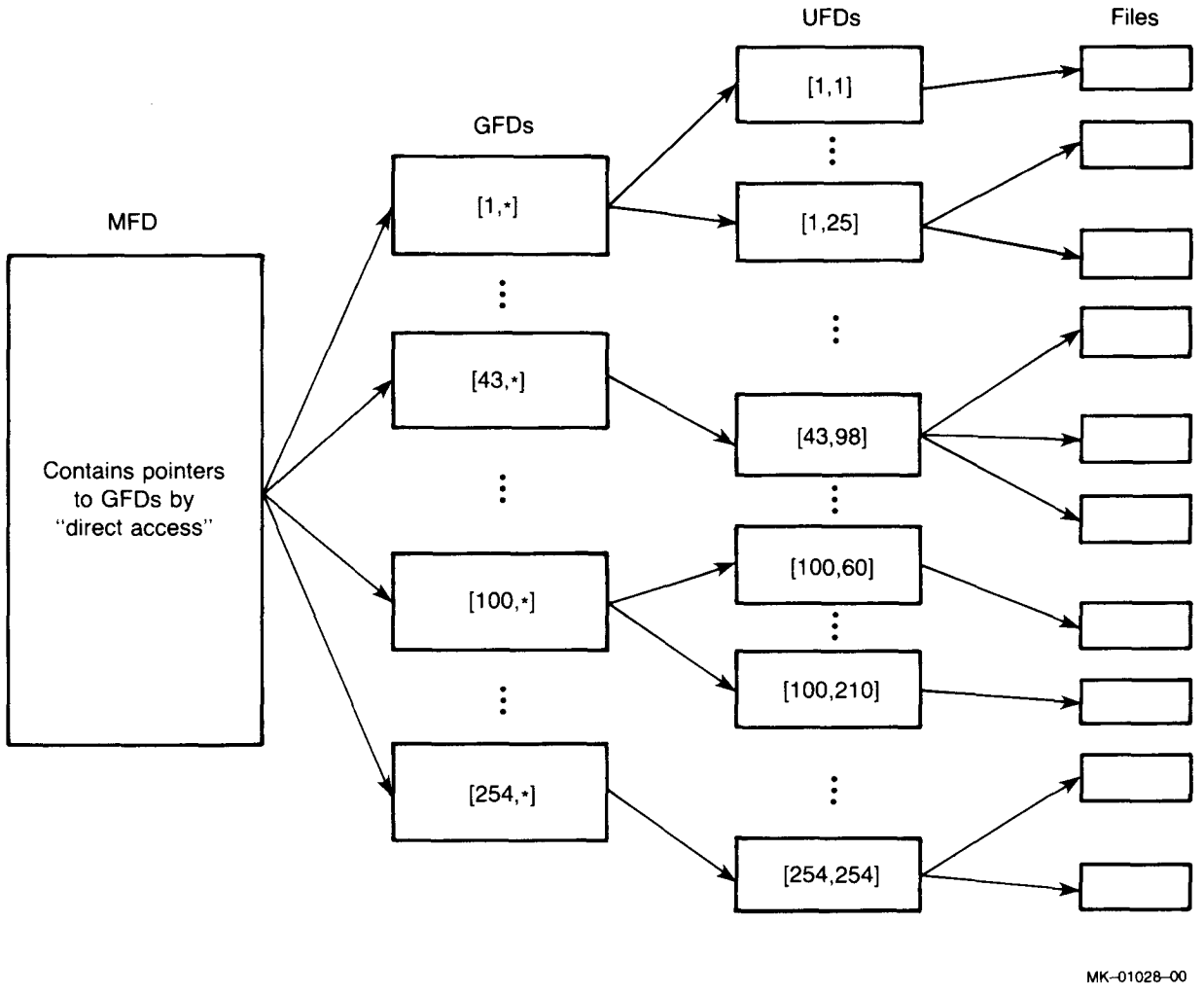
1.3.4 RDS1 Disk Structure for Version 8.0

As of Version 8.0, users access files on a RSTS/E system through the use of three structures:

1. A Master File Directory (MFD)
2. A Group File Directory (GFD)
3. A User File Directory (UFD)

Figure 1–2 shows how these structures are related.

Figure 1–2: MFD, GFDs, and UFDs in the RDS1 File Structure



As shown in Figure 1–2, the MFD contains pointers to the group file directories, or GFDs. The GFDs point to UFDs within each group. For example, a pointer in the MFD could access the GFD for files in [43,*] directories, from which it could access a specific UFD, such as [43,98].

1.3.5 Differences Between RDS0 and RDS1

The new disk structure (RDS1) differs from the previous disk structure (RDS0) in the following ways:

- The MFD is no longer associated with an account number.

  Formerly, the MFD was located in [1,1], and it had to be at Block #1. Now, Block #1 points to the location of the MFD, allowing the MFD to be placed anywhere on the disk.

- The MFD and GFDs in the new structure do not store files.

  Formerly, the MFD could store files as well as pointers to accounts. This use of the MFD was discouraged because it limited the number of accounts that could be created.

- GFDs allow more direct access to files

  Formerly, the MFD contained a linked list of pointers to all the UFDs. This meant that UFDs were listed in the order in which they were created with the REACT program, and that the MFD had to search the list of directories in sequence until it located a UFD. Now, the access method is more efficient. The MFD contains direct pointers to GFDs, which in turn can point directly to any UFD. (However, the structure of the UFDs remains the same as in RDS0.) Therefore, programs that display accounts (such as MONEY) list the accounts in ascending order, rather than in chronological order.

- You can create up to 64,771 accounts with the new structure.

  Formerly, only 1735 accounts could be created and referenced on a disk. Now, you can create accounts for 254 groups, each with 255 users, plus the account [0,1]. Only disk size restricts the number of accounts.

The users on your system should not notice the differences between the two structures. A major advantage to the new structure is that it offers better performance with less overhead. RDS1 takes up a little more disk space — there are more pointers, because each group is handled separately. However, access to files is faster and the system can have more accounts. The linked list structure of UFDs is replaced by direct access to GFDs.

1.3.6 Disk Optimization

When a user tries to access a file on the public structure, the system searches the directories on all disks that are part of the structure. The search either verifies that the file exists or verifies that it does not. You can avoid the overhead of searching more than one disk by placing frequently accessed files on a private disk. This technique is valid for systems that have a number of disks in the public structure, as well as for a system whose public structure consists of only one disk. The ability to place and preextend directories can also help you optimize disks; see Chapter 4 for more information.

It is sometimes advantageous to dedicate an entire private disk to a single large production file. This organization ensures an efficient directory structure and minimizes overhead to access file data. When more than one file is on the same private disk, it is best to dedicate a whole account to each production file. The system then spends less time searching the directory for file information.

In general, you should keep volatile files and stable files in separate accounts to avoid fragmentation. This condition, where file entries in the directory are spread across the face of the disk rather than being confined to a few sequential blocks, requires the disk head to move more times than is optimally efficient. Preextending directories, as described in Chapter 4, can ensure that the directory is in sequential blocks. You can also optimize disk usage by separating files that you access frequently from those that you use less often.

In an environment where distinct data files must be accessed by the same program, the optimal organization is to keep each file on a different private disk. If a program must access more than one file on the same disk, overhead is increased because of disk-head movement. A large percentage of time, therefore, is spent in moving the head back and forth. If, however, each file referenced by the program exists on a distinct private disk, head movement is not required whenever a program references another file. Head movement is restricted to locating the data itself. Positioning of directories, as described in Chapter 4, can also help reduce head movement.

1.4 System Operation Concepts

Immediately after logging in to the system, a user's terminal is under the control of the system's default keyboard monitor. The terminal is said to be at the system command level because you can type a system command and the keyboard monitor processes it accordingly. The terminal returns to the system command level when a command or program finishes executing, or when you type a CTRL/C at the terminal. When the terminal is at the system command level, the keyboard monitor examines each ASCII text line entered and determines whether that line is a system command. System commands are executed immediately after being entered as described in the *RSTS/E System User's Guide* and the *RSTS/E DCL User's Guide.*

The user job area is initialized at log-in time and set to a size of at least 1K words (K = 1024). The job area can grow in increments of 1K words to a maximum size set by the system manager at the start of time-sharing operations. The maximum size for any job image, including memory used by the job's current run-time system, is 32K words. This is the maximum size allowed by the PDP–11 architecture.

Under the RSTS/E system, jobs run one at a time. A job runs until it either enters an I/O wait state or exhausts the time quantum assigned by either the system or the system manager. When the currently running job stops running, the scheduler runs the next job that is ready. Meanwhile, the

interrupt-driven I/O device handlers are processing requested data transfers. After completing a transfer, the scheduler marks the job that requested the transfer as ready to run again and starts it from the point at which execution stopped.

RSTS/E tries to keep as many jobs in memory as possible. When more memory is required to run a job than is available, the system temporarily moves some jobs out of memory and stores them in one of four areas known as swap files. By convention, these files are named SWAP0.SYS, SWAP1.SYS, SWAP.SYS, and SWAP3.SYS. This operation is called swapping. When a job stored in a swap file is again eligible to run, it is swapped back into memory. Jobs waiting for keyboard input and jobs waiting for device I/O completion are most likely stored in the swap files, while jobs currently running or involved in disk or magnetic tape data transfers are necessarily in memory.

As the system processes each job, it maintains accounting information in memory concerning that job. When the job is logged off the system, this information is used to update the accounting information stored on the system disk for that account.

1.5 Privilege

Privilege is a special condition for a user job. With privilege, a job has capabilities that other, nonprivileged jobs on the system do not have. The sections that follow define privilege and explain how to acquire and relinquish it.

1.5.1 Privileged Capabilities

A privileged job on RSTS/E has the following special capabilities:

- Unlimited access to files on system

  This means no protection code can protect a file against read and write access. A privileged job can create and delete files under any account number and can access files on locked disks. Activity of this type by a privileged job does not generate the normal error (ERR=10): ?PROTECTION VIOLATION. Note, however, that privilege does not bypass intrinsic protection mechanisms such as locked blocks in files open in update mode (MODE 1).

- Ability to designate privileged programs

  A privileged job can set the privileged bit in the protection code. The privileged bit, with the compiled file protection bit, designates a program as privileged. Thus, any program with a protection code of <192> (the sum of the privileged protection <128> and the compiled file protection <64>) or greater denotes a privileged program. A privileged user must set both protection code values for a program to have privilege.

- Use of privileged aspects of system programs, privileged SYS system functions, and the PEEK function

Many system programs acquire privilege at system generation time. These programs execute privileged SYS system functions and the PEEK function and thus must be protected from unauthorized use. In general, information about privileged programs and operations will not appear in manuals the beginning user normally has access to. This manual describes privileged system programs and privileged aspects of system programs. The *RSTS/E Programming Manual* describes privileged SYS system functions and the PEEK function. The *RSTS/E System Directives Manual* describes the corresponding directives for assembly language programmers.

1.5.2 Privilege and System Operation

A job has privilege under one of the following conditions:

- It is a logged-out job (a job that is not associated with an account)
- It is running under a privileged account
- It is running a privileged program

The privilege remains or is dropped depending on how processing continues for the job.

A logged-out job has privilege because the system must minimally perform certain privileged operations to log a job in to the system. The privilege remains in effect as long as the job remains logged out, or as long as the program continues to run — even if it logs itself in. Therefore, the programmer would explicitly choose to make the program drop its privilege.

A job running under a privileged account has privilege. A privileged account is one whose project number is 1. The account number thus has the form [1,*], where the asterisk (*) represents a programmer number between 0 and 254. The system library account [1,2] is an example of a privileged account.

A job running under a privileged account has permanent privilege until the job is logged out or the job changes to a nonprivileged account (an account whose project number is not 1).

1.5.2.1 Establishing Program Privilege — You can use DCL or the PIP system program to set the privileged bit in the protection code. To designate a privileged program, assign a protection code of <192> or greater to the compiled form of the program. The following example shows the procedure:

```
$ SET PROTECTION=254 FILE.BAC (RET)
```

This protection code designates the privileged and compiled protection codes (<128> and <64>) with:

1. Read and write protection against the owner, the owner's project, and those outside the owner's project (<2> + <8> + <32>)
2. Execute protection against the owner's project and those outside the owner's project (<4> + <16>)

This protection code allows the owner to execute the file but allows read and write access to privileged users only.

When a disk file has the privilege <128> bit set, deleting the file causes the monitor to overwrite it with zeros under the following conditions:

1. The privileged <128> bit is set and you execute an explicit KILL or UNSAVE
2. You execute an OPEN FOR OUTPUT with the same file name, explicitly deleting the file
3. The file is tentative and you execute a reset CLOSE with a negative channel number
4. You perform a ZERO operation on the file's account
5. You use PIP or the DCL DELETE command to delete the file.

Because the monitor is in a locked state during the overwrite operation, no other programs on the system can perform file processing. Thus, permanent privilege is required to set the <128> bit. Also, because the monitor overwrites a file with the <128> bit set in the protection code when it is deleted, you should make sure to protect sensitive files, such as ACCT.SYS. That is, rename the file to its required protection code plus <128> but do not set the <64> compiled bit. Note that to process a privileged (<128>) file through either system spooling package, you must have permanent privilege. Furthermore, any attempt to delete such a file with the QUE program /DE switch, or with the DCL PRINT command's /DELETE qualifier, will fail.

1.5.2.2 Temporary Privilege – RSTS/E allows jobs running under both privileged and nonprivileged accounts to run privileged programs (that is, programs stored on disk with a protection code that includes both the privileged <128> bit and the compiled <64> bit). Jobs running under privileged accounts (privileged jobs) always have permanent privilege, but jobs running under nonprivileged accounts (nonprivileged jobs) run privileged programs with temporary privilege.

A job with temporary privilege has complete access to the system, but built-in system protection mechanisms again take effect when temporary privilege is dropped. The rest of this section describes how a nonprivileged job gains and loses temporary privilege.

A nonprivileged job gains temporary privilege when it starts executing a privileged program through a RUN command, a CCL command, or a BASIC-PLUS CHAIN statement that starts execution at the lowest numbered line. The temporary privilege remains in effect until either:

1. The program exits
2. The program drops its temporary privilege, either temporarily or permanently, by executing a system function call (see the *RSTS/E Programming Manual*)

In addition, the run-time system controls temporary privilege under certain conditions. For example, the BASIC-PLUS run-time system drops a job's temporary privilege when a job running under its control enters a keyboard monitor wait (^C) state. This action occurs if the program gets an untrapped error, executes a STOP statement, or is interrupted by a CTRL/C. BASIC-PLUS clears the program from memory and drops the job's temporary privilege. BASIC-PLUS may also drop a job's temporary privilege when a program is run by a CCL command at other than the lowest numbered line. The conditions under which this action occurs are described in the *RSTS/E Programming Manual*. Finally, when a program is run by a CHAIN statement at other than the lowest numbered line, BASIC-PLUS drops the job's temporary privilege unless:

1. The program executing the CHAIN has currently active temporary privilege, and
2. The program called by CHAIN has access to temporary privilege; that is, the program is stored on disk with a protection code that includes both the <128> bit and the <64> bit.

Privileged programs can drop and regain temporary privilege as needed by executing the system function call described in the *RSTS/E Programmming Manual*. However, a program can regain privilege only if it was temporarily dropped.

You can take advantage of the ability to drop and regain temporary privilege when you write programs that perform privileged operations, such as executing privileged system function calls. To prevent misuse of privilege, DIGITAL recommends that user-written privileged programs initially drop privilege and then regain privilege only as needed during processing.

1.5.3 Guidelines for Privileged Operation

The list that follows contains a few important guidelines the system manager and other privileged users should understand when using the full capabilities of their RSTS/E system:

- Use caution when using privileged operations

  Permanent privilege gives a user the full capabilities of the RSTS/E system. It also assigns to each privileged user a distinct responsibility. The capabilities can damage and even destroy the system. The system generates no error messages because privilege bypasses the normal protection mechanism. To avoid destroying data, you as system manager must carefully control and use privilege.

- Setting up privileged accounts

  Privilege capability begins with the assignment of privileged accounts. For example, when the system manager is setting up a system disk in system generation, the only privileged account that is required is the library account [1,2]. (Nonsystem disks are not required to have privileged accounts.) By having access to the system

library, you have privilege capability. One such capability lets you designate other users as privileged. You do this by assigning them accounts with a project number of 1. That is, those users on your system who have privilege are those that have accounts such as: [1,233], [1,196], or [1,243].

- Restrict access to privileged accounts

  The system manager should prevent unauthorized access to privileged accounts. Keep passwords to such accounts confidential and change them from time to time to a set of characters that are not easily guessed. Using your first or last name as a password is, for instance, not a good practice.

  Obviously, the responsibility for maintaining system integrity extends to other privileged users besides the system manager. It is therefore advisable to keep the number of privileged accounts as small as possible. Privileged users have all the capabilities of the system manager and thus must bear part of the responsibility of privileged use. Specifically, privileged users can design and implement routines that use privilege SYS system functions and the PEEK function.

- Privileged users can also designate programs as privileged

  If a program is designated as privileged and is not protected against execution, any nonprivileged user can run the program with temporary privilege. This type of activity extends privilege to nonprivileged users in the same way the standard system library programs allow access to its programs. For example, the program TTYSET allows users to change the characteristics of their terminals. Such operation is privileged. With temporary privilege, however, TTYSET executes the normally privileged operation for an owner of a nonprivileged account. TTYSET does not generate the expected protection violation error.

  Furthermore, the same TTYSET program lets a privileged user change the characteristics of another user's terminal. As a precaution, however, TTYSET checks to make sure a user attempting to change the characteristics of another's terminal is indeed a permanently privileged user. Thus, TTYSET provides a nonprivileged user certain privileged operations but restricts other privileged features to privileged users only.

- Privileged programs in nonprivileged accounts

  It is not a good practice to place a privileged program in a nonprivileged account. (A privileged program is stored in any file that has both the privileged <128> code and the compiled <64> protection code set.)

  If you do place a privileged program in a nonprivileged account, however, the nonprivileged user does not have write access to the file, regardless of its protection code. This restriction is intended to prevent the user from accidentally corrupting the file.

Chapter 2
System Start Up, Shutdown, and Automatic Restart

This chapter describes how to start and halt a RSTS/E system and how the system recovers automatically from catastrophic system errors.

2.1 Starting up RSTS/E

To start a RSTS/E system, you must first load a program called the initialization code into memory and then use one of the options this program provides to perform the start-up operation.

The initialization code is one large stand-alone program that, once you transfer it from the RSTS/E distribution kit, exists on the system disk in a file called INIT.SYS. You use the options the initialization code provides to perform a number of important system functions, such as:

- Creating the RSTS/E file structure
- Creating system files
- Installing a RSTS/E monitor
- Setting system defaults
- Creating start-up conditions for RSTS/E

This chapter describes how to create the start-up conditions for a RSTS/E system. Refer to the *RSTS/E System Generation Manual* for a description of all the other initialization code options and the functions they perform.

Before you can use the initialization code to start your RSTS/E system, you must move the code into memory. This is called a "bootstrapping" operation. The following sections describe how to transfer the intialization code into memory.

2.1.1 Bootstrapping RSTS/E with a Hardware Bootstrap Loader

The procedures for bootstrapping RSTS/E depend on the type of hardware bootstrap loader and the type of disk used as the system device. You should make sure:

1. The system disk is physically mounted on a disk unit.
2. The drive on which the system disk is mounted has the READY light on and is in the WRITE ENABLE condition.
3. All required units are running and READY, including any RK05F disks.
4. The console terminal is on line.

Read the *RSTS/E System Generation Manual* for the proper instructions on using the hardware bootstrap on your system

After you use the bootstrap to load the initialization code, the INIT.SYS program marks the successful completion of the bootstrap procedure by printing an identification line that includes the version number, the disk unit that was bootstrapped, and the installation name, followed by the Option: prompt:

```
Enabling only console, disks, and tapes

RSTS V8 (DB0) TIMESHARING

Option:
```

If the program does not print this information, make sure the console terminal is on line and try the operation again. If an error message is printed on the console terminal, refer to the *RSTS/E System Generation Manual* for recovery procedures.

If you want to enable the automatic restart facility, set the CPU switch register so that bit zero is on. The automatic facility remains enabled as long as the CPU switch register remains set in this way. If your CPU does not have a switch register, the automatic restart facility is always enabled.

2.1.2 Bootstrapping RSTS/E after a System Halt

When a RSTS/E system halts as a result of a catastrophic error, the halt address is displayed in the address lights. For central processors that do not have console lights (such as the 11/23–PLUS, the 11/44, and CPUs with RDCs – Remote Diagnostic Consoles), the system prints the address on the console terminal. It is essential that you record this address and the first ten locations in memory. The exact procedure for examining memory locations depends on the type of processor and front panel that your system has. Your DIGITAL Software Specialist can describe the exact procedure for your system. Once you record this information, bootstrap your system again using the procedures described in Section 2.1.1. (See Section 2.3.1 for a description of catastrophic errors, system crashes, and crash dumps.)

2.1.3 Starting Timesharing

After you bootstrap the system disk, the initialization code (INIT.SYS) is loaded into memory. INIT then prints on the console terminal an identification line followed by the Option: prompt. The program is thus ready to accept one of the options summarized in Table 2–1. (Refer to the *RSTS/E System Generation Manual* for a complete summary of these options.) At this point, you can use any of the INIT.SYS options to perform system operations. If you do not need to perform any of these tasks, type START and press the RETURN key to begin timesharing.

```
Option: START(RET)
```

After you press the RETURN key, the system prints a few messages and begins time-sharing operations. Table 2–1 summarizes all of the options the initialization code makes available to you.

Table 2–1: Initialization Option Summary

Option Name	Abbreviation	Meaning
DSKINT	DS	Initializes and optionally formats a disk. It also checks for bad blocks.
COPY	CO	Transfers required files from the distribution medium to a system disk and bootstraps the system disk.
PATCH	PA	Alters RSTS/E system code to correct program errors and to add new program features.
HARDWR	HA	Specifies device controller characteristics.
INSTALL	IN	Installs a RSTS/E monitor.
REFRESH	RE	Creates and positions system swap files, adds bad blocks to the bad block file, and positions optional system files on the system disk or any auxiliary disk.
DEFAULT	DE	Establishes or changes a monitor's start-up defaults such as the job and swap mamimum.
SET	SE	Sets device characteristics on a unit-by-unit basis.
START	ST	Starts the RSTS/E system for SYSGEN or normal timesharing. Press the LINE FEED key to start timesharing also.
BOOT	BO	Bootstraps a tape or disk.
SAVRES	SA	Backs up (SAVE) and reconstructs (RESTORE) a RSTS/E disk.
HELP	HE	Prints a help message.

2.2 Halting the RSTS/E System

A halt on a RSTS/E system is caused by a system crash or by setting the CPU console HALT/ENABLE switch to its HALT position. The PDP–11/70 and PDP–11/45 are considered in a halt state when both the RUN

and PAUSE indicators on the CPU console panel are not lit. For PDP–11/23–PLUS and PDP–11/44 processors, the RUN light is off when the system is in a halt state. In all other cases, the CPU is running.

The SHUTUP program shuts down the RSTS/E system. During this operation, SHUTUP makes sure all files are properly closed and that system accounting information is accurately updated. If you want to turn the power off, run SHUTUP, wait for the Option: prompt, and press the HALT/ENABLE switch. Refer to Section 3.2 for a complete description of the SHUTUP program.

It is dangerous to halt RSTS/E by moving the HALT/ENABLE switch on the CPU console to its HALT position, if you do so during system shutdown. SHUTUP may not have completed clean-up operations. File data can consequently become corrupted and accounting information may be lost. The only way to recover from such a disorderly halt and to salvage possibly vital file information is to raise the HALT/ENABLE switch back to its ENABLE position before any other action is taken and press the CONT switch, thereby returning the RSTS/E system to the state it was in before the halt occurred.

2.3 Automatic Recovery and Restart Facilities

This section describes how catastrophic errors occur and how your system can automatically recover from them. It also describes the need to include commands in the CRASH.SYS file to perform critical system operations when the system is initialized in automatic restart mode.

2.3.1 Catastrophic Errors and System Crashes

A catastrophic error or a system crash results from an unexpected error trap. (For information on error traps, see the related processor handbook.) Unexpected error traps can be caused when the processor:

- Refers to a nonexistent or nonresponding UNIBUS address (bus timeout trap)
- Refers to an odd address with an instruction that requires a word address
- Attempts to execute a reserved or nonexistent instruction

Catastrophic errors and system crashes, therefore, may occur as a result of three types of problems:

1. Privileged account programming errors

   The RSTS/E system software is designed to protect itself against programming errors that occur under nonprivileged accounts. The system itself, when it detects an error of this kind, aborts execution of the potential error and reports a corresponding error message.

The RSTS/E software is vulnerable, however, to certain types of errors caused under privileged accounts. By intent and design, those users who have [1,*] accounts have extensive powers that are not available under nonprivileged accounts. These powers allow privileged users to modify system library programs or to create utility programs so that they can access parts of memory. Even though most mistakes do not cause the system to crash, users should take special care when programming with privileged system function calls. (Refer to the *RSTS/E Programming Manual.*)

2. Hardware malfunctions

   Hardware malfunctions can cause the system to crash. If system crashes or catastrophic errors randomly occur that you cannot explain (particularly on systems that have traditionally been functioning well), it is likely that a hardware problem has arisen. You can diagnose hardware problems by examining the output logged by the programs in the RSTS/E System Error Package. (Refer to Chapter 6 for information on error logging.)

3. System software malfunctions

   Although every attempt has been made to detect and eliminate system software errors, the paths through the RSTS/E software are incalculably numerous. It is possible that, given certain conditions and certain sequences, the RSTS/E software can trap unexpectedly. If a problem of this type is discovered (it should be reproducible in a defined environment and under defined conditions), you should contact a DIGITAL Software Specialist. As problems of this type become known, DIGITAL reports them as described in the *RSTS/E Software Dispatch.*

2.3.2 Automatic Recovery from Catastrophic Errors

The system places a trap in one of two categories when an unexpected trap occurs:

1. A catastrophic error that affects only one user
2. A system crash for which some software or hardware problem is possibly responsible

For a catastrophic error, the system determines which user was responsible for the error trap. It flags the user's job with a special code that causes the system to completely reinitialize that user's job area when it is the user's turn to run. The system prints an error message on the user's terminal, followed by the text: ?PROGRAM LOST-SORRY.* The user is now in the same state as someone who has just logged in. Finally, the system resumes normal time-sharing operations.

* Run-time systems other than BASIC-PLUS may issue other error messages.

When the system detects a condition from which it cannot recover, it does the following:

1. If the system manager enabled the crash dump facility at start-up time, the system writes an image of read/write memory and tables to the CRASH.SYS file.
2. If the CPU switch register has bit 0 set and at least one minute has passed since start-up, the system bootstraps the system disk, loads the initialization code into memory, and executes an automatic restart.

If the CPU register does not have bit 0 set or one minute has not passed since start-up, an automatic restart will not be performed. The system instead reboots but does not restart. To clarify, the automatic restart operation restarts timesharing using commands in the CRASH.SYS file. A reboot, in contrast, merely loads the initialization code (INIT.SYS) into memory. INIT.SYS then prints its OPTION: prompt, to which you can type any of the options described in Table 2–1.

If the system takes the automatic restart path, the system bootstraps itself into memory from the system disk. After the system has been bootstrapped into memory, control is passed to the initialization routines. The system then recognizes that it was not started through a normal system start-up but rather through an automatic restart. It consequently initializes itself in automatic restart mode. If an error trap occurs within one minute of system start-up or if two error traps occur within the same minute, the system halts at address 54. Note that during automatic restart, the system protects itself against an infinite loop or error traps and automatic restarts caused by some repeating hardware malfunction.

2.3.3 Automatic Restart Mode Initialization

When the system is initialized in automatic restart mode, control bypasses all parts of the initialization code that call for operator intervention and initializes the system using information already stored in memory. The system logs job 1 in to the system on KB0: under account [1,2] and causes the job to run the system library program INIT beginning at line 100. Because INIT begins at line 100 in automatic restart mode (instead of at its lowest line number), it takes directions from the CRASH.CTL system library file (rather than from the operator-designated control file). The CRASH.CTL file must contain INIT commands that perform all operations the system manager considers necessary in the case of an automatic restart. Refer to Section 3.1 for information about controlling system start-up.

2.3.4 Power-Fail Hardware Used by RSTS/E

When power fails, a PDP–11 processor traps through the vector at location 24. When power is restored, the system may be configured to do one of three things:

1. Halt.
2. Boot a specific device. If the hardware is configured to boot the system disk, INIT.SYS will print its "OPTION:" prompt and wait for you to enter one of the options from Table 2–1.
3. Trap again through the vector at location 24. In this case, RSTS/E must have core memory or battery backup.

Chapter 3
Controlling Timesharing

You can control system start-up with the compiled version of the system program INIT.BAS. Start-up occurs when you use the START option of the initialization code or when the system restarts automatically. At start-up, the monitor transfers control to the primary run-time system at its start-up entry point. Then, the run-time system runs the INIT program which executes special commands from a control file. You can control start-up by selecting commands for the control file that suit the requirements of your installation.

3.1 What INIT.BAS Does

To control start-up efficiently, you must understand the conditions at start-up time:

- Login attempts are prohibited (the monitor disables the login capability)
- The monitor logically mounts only the system disk
- No output is made to any terminal
- The monitor logs the console terminal KB0: in to the system under the system library account [1,2]
- At the console terminal, the monitor executes code equivalent to the BASIC-PLUS statement CHAIN "INIT" (for manual start-up) or CHAIN "INIT" LINE 100 (for automatic restart)

As soon as the monitor executes the command for manual start-up or automatic restart, the INIT.BAS program begins to execute. The tasks INIT.BAS performs depend on the way it was initiated — manual start-up or automatic restart:

Manual Start-up

When INIT.BAS runs in response to the START option, it executes from the lowest line number and then prints a header line and the following question at the console terminal:

```
CONTROL FILE NAME?
```

Enter the name of the command file, which must exist on the system disk (SY0:). You can enter the name of your own command file, but if 30 seconds pass without a response or if you press the RETURN key, INIT.BAS, by default, uses START.CTL on the system disk (SY0:) in account [1,2].

Automatic Restart

The INIT.BAS program does not print a prompt when it runs as a result of an automatic restart caused by a system crash. Instead, it automatically reads the file CRASH.CTL on the system disk (SY0:) in account [1,2].

Error Conditions

If the INIT program encounters an error condition while reading the control file, it prints an error message in the form:

```
text   - ERROR IN READING  filename
```

The text of the error message helps you better understand the condition that caused the error to occur. INIT tells you the name of the file it tried to read but could not. The console terminal remains logged in to the system, but INIT stops executing. After you fix the condition that caused the error, start the initialization again by typing RUN $INIT.

To stop the execution of the INIT program before it comes to a normal end, type CTRL/C. How INIT reacts to the interruption depends on the state it was in at the time you tried to stop the program:

Detached

If INIT is running detached (that is, it is not currently associated with a keyboard), it returns the following information to the keyboard device to which INIT was last attached:

```
CTRL/C DETECTED ON TARGET KB: (KBn:)
<text>
INIT WILL 'KILL' ITSELF
```

The message that INIT prints identifies the keyboard from which you issued the CTRL/C and provides information about the aborted run. The program returns control to the keyboard monitor.

Attached

If INIT is running attached, it returns the following information to the keyboard device to which is attached:

```
CTRL/C DETECTED ON TARGET KB:(KBN:)
<text>

$
```

INIT identifies the keyboard on which the CTRL/C was typed and prints a message containing information about the aborted run. The program then returns control to the keyboard monitor, in this case DCL, at the terminal to which the program is attached.

Control files contain special commands for initializing the system for timesharing, recovering system crash information, and performing other required operations. The following sections describe the INIT commands and how to use them.

3.1.1 INIT Program Commands

The RSTS/E system is not fully initialized until INIT runs and executes commands in the start-up control file. For instance, a system generally uses more disks than just the system device. With the MOUNT command, you can make public and private disks immediately available to users before they can log in to the system. The start-up procedures must include instructions to make the specified devices physically ready on the proper disk drive. Table 3–1 contains the descriptions of the commands used to control how your system enters timesharing.

Table 3–1: Control File Commands

Command Name and Format	Use
LOGINS	Allows users at both local and remote terminals to log in to the system.
SEND xxx	Transmits the text xxx to all keyboards currently on line, except to the terminal on which INIT is running.
@filespec	Causes INIT to process the indirect command file specified by filespec and stored on a currently mounted disk. The specification can be any valid RSTS/E disk file specification. If you do not specify a file type, INIT uses the file type .CMD. INIT returns to the next command in the current control file when the indirect command file is completed. Indirect command files can contains at sign commands (@) to allow nesting to ten levels.
LOGIN KBn: [n,m]	Logs the terminal KBn: in to the system using account [n,m]. INIT automatically looks up the password. Note that unless INIT has been detached, you cannot use the LOGIN KBn: command on the terminal to which INIT is attached.
FORCE KBn: xxx	Places the text xxx in the input buffer of the terminal KBn: and executes the text if as if you had typed it at the terminal keyboard. The text can be any BASIC-PLUS or system command.

(continued on next page)

Table 3–1: Control File Commands (Cont.)

Command Name and Format	Use
FORCE KBn:xxx (cont.)	To force a control character combination (such as ^C) to a terminal, use the circumflex (^) in the first position of the text, place the control character (in this case, C) in the next postion, and then press the RETURN key. No other text should follow the control character combination. If the circumflex (^) is the only character in the text, INIT forces a CTRL/C to the terminal. If the circumflex is the first character of more than two characters of text, INIT forces a CTRL/C to the terminal ahead of the text. Note that if you attempt to force a CTRL/Z to the terminal to which INIT is attached, an error is generated and INIT ends. To properly terminate INIT, place either the END or BYE command as the last command in the control file. If an attempt is made to force a CTRL/C to the terminal to which INIT is attached, an error is generated and INIT stops. To properly terminate INIT, use the END or BYE command as the last command in the control file.
MOUNT dev:id MOUNT dev:id/RONLY	Causes RSTS/E to logically recognize the disk unit specified by the device designator and by the pack identification. As an option, you can include the /RONLY switch to mount the disk with read only access. Refer to Chapter 7 for information about mount, clean, and unlock operations.
ADD SWAPFILE n dev:[filename]	Adds swap file n with a file specification of filename to a disk device (dev:). Valid swap file numbers n are 0, 1, and 3. Swap files must have a file type of .SYS and be in the system account [0,1]. Make sure to mount the device before you use this command. If you do not include a device, INIT uses SY:, the public structure.
ADD ERROR dev:filename	Adds the file to device dev: from which the system can extract error message text. INIT stores the file in account [0,1] and gives it a file type of .SYS. Make sure to mount the device before you use this command. If you do not specify a device, INIT uses SY:, the public structure.
ADD OVERLAY dev:filename	Adds, as the system overlay file, the file with the name given and with file type .SYS stored in account [0,1] on the device and unit indicated. Mount the device before you execute the ADD OVERLAY command. If you do not include a device, INIT uses the public structure (SY:).

(continued on next page)

Table 3–1: Control File Commands (Cont.)

Command Name and Format	Use
DETACH	Causes INIT to detach from the terminal from which it is running and print the message DETACHING. With INIT running detach, you can log in to the system at that terminal and use it for other purposes. When you want to reattach to INIT, use the ATTACH command.
ATTACH	Attaches INIT to the terminal from which INIT was detached by the DETACH command. The terminal should not be in use by another job because INIT will wait until the terminal is free. The commands preceding ATTACH should log out the job or detach it from the terminal.
BYE or END	Causes INIT to stop executing. BYE causes the job running ERRINT under account [1,2] to be logged out. This frees the console keyboard for other use and prevents unauthorized use of account [1,2]. Use END instead of BYE when running ERRINT on the console terminal. END does not log out the job running ERRINT under account [1,2]. See Section 6.1.1 for a description of ERRINT. If you do not include an END or BYE command in the control file, INIT gets an END OF FILE ON DEVICE error and leaves the terminal logged in to the system.

To have the system execute other operations on the system, you can instruct INIT to execute system commands and programs. The LOGINS command to INIT enables further logins on the system. The LOGIN command automatically logs a specified job in to the system at a designated terminal to allow execution of commands and programs. The FORCE command can run the TTYSET system program at the terminal and can then execute commands to establish terminal characteristics of certain keyboards. The SEND command prints text on all on-line terminals.

The INIT program can also be run during timesharing to perform routine operations. Special control files can be created with the proper sequence of commands to execute. The system manager can simply run INIT and specify the proper control file from which INIT will extract the commands to perform the required routine operation.

3.1.2 Creation and Use of Control Files

The INIT program control files must contain the commands necessary to properly initialize your RSTS/E system. You must be sure the control files are on the system disk, which can be mounted on any drive. The files that come on the RSTS/E distribution kit are only samples and may not execute properly on your system without modification. You must make sure the files

include the necessary commands to initialize your installation. To replace the sample START.CTL file supplied with the RSTS/E kit, use a text editor on your system (such as EDT) to modify it:

```
$ EDT $START.CTL (RET)
    1               @TTY.CMD
*INSERT (RET)
                    @MOUNT.CMD
                    ^Z
    1               @TTY.CMD
*TYPE WHOLE (RET)
    0.1             @MOUNT.CMD
    1               @TTY.CMD
    2               ;@SPOOL.CMD
    3               @RTS.CMD
    4               @CCL.CMD
    5               FORCE KB0: RUN $ERRINT
    6               FORCE KB0: 100
    7               FORCE KB0: NO
    8               LOGINS
    9               SEND RSTS/E IS NOW ON THE AIR...
   10               END
[EOB]
*EXIT (RET)
START.CTL 17 lines

$
```

This procedure ensures the new file is created on the system disk (SY0:). The SY0: designator is critically important on systems with more than one disk on the public structure. Its use explicitly identifies the system disk (the disk from which you booted the system). The file must reside on the system disk because, when INIT initially runs, only the system disk is mounted.

It is important for the start-up control file to perform the following operations:

- Mount all public disks
- Add system files
- Use indirect command files to perform routine operations like:
  - Set terminal characteristics
  - Establish auxiliary run-time systems
  - Define CCL commands
  - Run spooling and operator services programs
- Run the ERRINT program to start error logging
- Enable logins
- Broadcast any messages
- End the control file

The order in which the operations are performed (and thus, the order of commands in the control file) is critical. For instance, mounting all public disks ensures the integrity of the public structure by making available all files on the system. You can mount any private disks in a similar manner. By adding system files, such as the swap, error message, and overlay files, you establish the monitor structures for a system with full system file capabilities. The routine operations further tailor the system hardware and software according to local installation requirements.

It is important for you to run the ERRINT program on your system. This program sets initial conditions for system error logging and runs the ERRCPY program, which transfers to a disk file all errors logged by the RSTS/E monitor. If ERRINT is the last program to be run and is run as recommended, it will occupy job number 1 on the system and thereby be easily monitored. ERRINT and ERRCPY are described in Chapter 6.

The command to enable logins should appear after the commands that perform all required and routine operations. This command activates all swap space added on the system and allows users to log in to the system. You can then broadcast any messages notifying users that timesharing has started, for example. To stop INIT without an error, the END command is necessary, especially in a control file that runs the ERRINT program. The END command leaves the terminal on which INIT is running logged in to the system. When ERRINT runs, it uses the job number (usually number 1) given up by the termination of INIT. ERRINT detaches itself and frees the terminal for other use. You can use the BYE command in other control files that do not run ERRINT and that must log out the terminal running INIT.

It is also important for the crash control file to perform operations similar to those that you ordinarily perform at system start-up. The following additional operations are recommended. You might want to have the crash control file submit these operations as a batch job, because completing them can take a long time.

1. Run ONLCLN to rebuild all disks.
2. Run the ANALYS program to extract system data and error logging data from the CRASH.SYS file.
3. Print the files that ANALYS creates.

After these important operations are performed, you can initialize the system for normal time-sharing operations.

The methods used by the start-up and crash control files to perform operations is also important. The control files should take advantage of INIT and RSTS/E features by performing operations in the following manner:

1. Do all forces of commands to the system console terminal KB0:.
2. Perform all routine operations in indirect command files.
3. Use INIT's ability to detach itself from the terminal that initiated it and to reattach itself to that terminal.

Because INIT can detach itself from the terminal that initiated it, all commands that are forced to that terminal are executed while INIT is running. In addition, on all RSTS/E systems, you can log in to the system on the system console terminal regardless of the number of logins currently allowed.* Assuming the control file adds sufficient swap space, subsequent commands in the control file can start as many jobs as required on the system console terminal without having logins enabled. When logins are not enabled (one of the start-up conditions), users cannot log in to the system while start-up and recovery operations are still in progress.

By placing routine operations in indirect command files, you can simplify the main start-up and crash control files. Changing the main procedure becomes a simple matter of changing one indirect command in the main control file. If you isolate routine operations to a separate file, this also makes the flow of the main control file easier to follow.

3.1.2.1 Detaching and Attaching INIT – The DETACH command of INIT frees the terminal on which INIT was run, and the ATTACH command reattaches INIT to that terminal. Because of the way RSTS/E and INIT handle text forced to a terminal, these commands allow more flexible use of the terminal on which INIT was initialized. In particular, the commands allow you to use the system console terminal to start system jobs and to perform other routine operations while INIT is still running and logins are not enabled.

The way INIT handles FORCE commands depends on whether INIT is attached or detached. If INIT is attached, it checks whether the FORCE command is being sent to the terminal on which it is running. If the destination is its own terminal, it makes sure that a CTRL/C is not being forced. A CTRL/C forced to INIT's own terminal prematurely and perhaps unintentionally terminates. Therefore, INIT discards all such FORCE commands, prints an error message at the terminal, and stops processing the control file immediately. You must correct the control file, and rerun INIT. When INIT is detached, it does not need to check the forced text because the detached job is unaffected by a CTRL/C response.

When INIT processes a FORCE command, it first checks the output buffers of the destination terminal. If the terminal is busy performing output, INIT waits until the buffers are cleared before actually executing the force operation. As a result, output and input text at a terminal are not mixed.

FORCE commands that INIT executes depend on the state of the destination terminal. You should design a sequence of FORCE commands to exactly simulate manual input. The commands INIT forces are executed as if they were typed at a terminal. Commands forced to a terminal that is not logged in must either be valid logged out commands (if you have applied the LOGIN feature patch to enable logged out commands on your system) or a

* As long as swap space is available on the system, a user can log in to the system on keyboard unit 0 regardless of the number of logins currently allowed. The system manager can install a patch that allows a different terminal to have this special privilege.

sequence of input that logs a terminal in to the system. (The LOGIN command of INIT simply forces the proper sequence of text to log a job in to the system.) If commands are forced to the terminal that initiated INIT and if INIT is attached to that terminal, the commands are not executed until the terminal is under the control of a keyboard monitor or is logged out.

Commands forced to the terminal that initiated INIT are executed according to whether INIT has freed the terminal (the DETACH command was executed), or INIT has terminated (by either the END or BYE command). Because of these conditions, it is recommended that:

1. The routine operations done by INIT be performed on the system console terminal while INIT is detached. This procedure ensures that the operations are performed while INIT is still running and, more importantly, before INIT has executed the LOGINS command to enable users to gain access to the system.

2. Three FORCE commands, the command to start ERRINT and the commands to answer its two dialogue questions, be executed while INIT is attached. This step causes the FORCE command to be executed after INIT terminates. Because this step comes near the end of the control file, there is little delay between the time users receive the message that timesharing has started and the time INIT terminates, allowing ERRINT to run. An additional benefit is that ERRINT occupies job number 1 on all systems and is therefore visible to persons responsible for maintaining the system.

3. Each indirect command file performs a DETACH and ATTACH operation. The use of the detach feature makes each indirect command file independent of the main control file. Because INIT must be attached to its terminal to properly terminate and to print error messages, this method also ensures the main control file is always processed while INIT is attached.

3.1.2.2 Indirect Command Files – An indirect command file consists of an ASCII file containing INIT program commands that it executes when the file is specified in either a main control file or another indirect command file. The at sign character (@), followed by a file specification, causes INIT to temporarily close the current file it is processing, open the indirect command file, and process its commands. If no account is specified in the command, INIT searches for the indirect command file in the account under which the program is currently running. If there is no file type included with the file specification, INIT attempts to open the file with a .CMD file type. After processing the commands in the indirect command file, INIT closes the file, reopens the previous file, and continues processing the remaining commands.

An indirect command file does not need to contain an END or BYE command, unless it is to be run as a main control file. If an END or BYE command is found in an indirect command file, INIT discards it and continues processing. This means that only an END or BYE command in the main control file can properly stop the execution of the INIT program.

Use indirect command files to perform the routine operations necessary at system start-up time. Each indirect command file to be run for system start-up or crash recovery should perform the following operations:

1. Detach itself from the terminal on which INIT was run (usually the system console terminal).
2. Use the proper sequence of LOGIN and FORCE commands to perform the desired operations at the terminal from which INIT is detached.
3. Reattach INIT to its terminal.

This sequence of steps is the most efficient way to run INIT during system start-up. If you use the system console terminal to issue FORCE commands while INIT is detached, then logins do not need to be enabled and all routine operations are performed while INIT is still running and logins are still disabled. INIT remains attached while commands in the main control file are being executed. Because an indirect command file contains only the LOGIN, FORCE, DETACH and ATTACH commands, it is less likely that INIT will encounter an unexpected error, attempt to issue a message to its keyboard (while detached), and enter the hibernate state. (INIT prints errors on its console terminal. If INIT is always attached while the main control file is executing, and the indirect command file contains only the recommended commands while INIT runs detached, the chance that the job will enter the hibernate state is greatly diminished.)

3.1.3 Setting Terminal Characteristics — TTY.CMD

At start-up time, the system sets the characteristics of all keyboard lines (except line number 0) to hard copy, 72 column output at the line speed of 110 baud. Thus, run the TTYSET program to set characteristics that match the local hardware. For example, if keyboard 1 is an LA120 terminal running at 2400 baud and keyboard 2 is a VT100 running at a speed of 9600, you can create an indirect command file to properly set the characteristics:

```
DETACH
LOGIN KB0: [1,2]
FORCE KB0: RUN $TTYSET
FORCE KB0: KB1:;LA120;SPEED 2400
FORCE KB0: KB2:;VT100;SPEED 9600
FORCE KB0: EXIT
FORCE KB0: BYE/F
ATTACH
```

The indirect command file detaches INIT, logs the system console terminal in to the system under a privileged account, forces the necessary sequence of TTYSET commands, logs the terminal off the system, and reattaches INIT.

3.1.4 Spooling and Operator Services — SPOOL.CMD

You can start auxiliary jobs on the system with an indirect command file. The following sample command file shows the procedure:

```
DETACH
LOGIN KB0: [1,2]
FORCE KB0: RUN $OPSER
FORCE KB0: OPE KB0:[2,2]
FORCE KB0: LOG OPSER.LOG;ALL
FORCE KB0: MES ALL
FORCE KB0: DET
LOGIN KB0: [1,2]
FORCE KB0: RUN $QUEMAN
FORCE KB0: DET
LOGIN KB0: [1,2]
FORCE KB0: RUN $SPOOL
FORCE KB0: LP0:/ASS/PRI:0/RUN:6
LOGIN KB0: [1,2]
FORCE KB0: RUN $BATCH
FORCE KB0: BA0:/PRI:0/RUN:6/QUE:LP0:/NODEL/ERROR:FATAL
ATTACH
```

The commands are in the proper sequence to run the spooling jobs. OPSER must be run first, followed by QUEMAN and any spooling programs to be started. For more information on the operator services and spooling programs, refer to Chapter 5.

If your system is running the small spooling package (new for Version 8.0), then SPOOL.CMD should instead contain commands such as these:

```
DETACH
LOGIN KB0: [1,2]
FORCE KB0: RUN $QUEUE
FORCE KB0: START/QUEUE/MANAGER
FORCE KB0: INITIALIZE/PRINTER LP0:/FORMS=NORMAL
FORCE KB0: EXIT
ATTACH
```

If your system supports both spooling packages, put the above commands for the small spooling package at the end of the other commands in SPOOL.CMD.

3.1.5 Establishing Auxiliary Run-Time Systems — RTS.CMD

You can use an indirect command file to add auxiliary run-time systems at the start of timesharing. The following sequence of commands shows the procedure:

```
DETACH
LOGIN KB0: [1,2]
FORCE KB0: RUN $UTILTY
FORCE KB0: ADD RT11
FORCE KB0: ADD RSX
FORCE KB0: ADD BASIC
FORCE KB0: ADD DCL
FORCE KB0: DEFAULT KBM DCL
FORCE KB0: EXIT
FORCE KB0: BYE/F
ATTACH
```

The ADD command of the UTILTY program creates the monitor structures necessary for auxiliary run-time systems. For more information on run-time system control, refer to Section 7.1.5.

The RTS.CMD command file on your distribution medium automatically adds DCL each time your system comes up under timesharing. It does not make DCL the default keyboard monitor. If you want DCL, include the following line in your RTS.CMD file:

```
ADD DCL
```

Otherwise, delete it. To make DCL the default keyboard monitor, include in your RTS.CMD file:

```
DEFAULT KBM DCL
```

Place this command on the line immediately following the ADD command.

3.1.6 Defining CCL Commands — CCL.CMD

The following example shows the commands that the system programs handle:

```
DETACH
LOGIN KB0: [1,2]
FORCE KB0: RUN [1,2]UTILTY
FORCE KB0: CCL ATT-ACH=[1,2]LOGIN.*;PRIV 30000
FORCE KB0: CCL BCK-=[1,2]RMSBCK.TSK;0
FORCE KB0: CCL BYE-=[1,2]LOGOUT.*;PRIV 0
FORCE KB0: CCL CCL-=[0,1]DCL.DCL;PRIV 0
FORCE KB0: CCL CNV-=[1,2]RMSCNV.TSK;0
FORCE KB0: CCL DCL-=[0,1]DCL.DCL;PRIV 0
FORCE KB0: CCL DEF-=[1,2]RMSDEF.TSK;0
FORCE KB0: CCL DES-=[1,2]RMSDES.TSK;0
FORCE KB0: CCL DIS-MOUNT=[1,2]UMOUNT.*;PRIV 30000
FORCE KB0: CCL DI-RECTORY=[1,2]DIRECT.*;PRIV 30000
FORCE KB0: CCL DSP-=[1,2]RMSDSP.TSK;0
FORCE KB0: CCL EDT-=[1,2]EDT.TSK;0
FORCE KB0: CCL HELLO-=[1,2]LOGIN.*;PRIV 0
FORCE KB0: CCL HELP-=[1,2]HELP.*;PRIV 30000
FORCE KB0: CCL IFL-=[1,2]RMSIFL.TSK;0
FORCE KB0: CCL LIB-R=[1,2]LIBR.SAV;8208
FORCE KB0: CCL LIN-K=[1,2]LINK.SAV;8208
FORCE KB0: CCL LOG-IN=[1,2]LOGIN.*;PRIV 0
FORCE KB0: CCL MACR-O=[1,2]MACRO.SAV;8216
FORCE KB0: CCL MAC-=[1,2]MAC.TSK;0
FORCE KB0: CCL MOU-NT=[1,2]UMOUNT.*;PRIV 30000
FORCE KB0: CCL PAT-=[1,2]PAT.TSK;0
FORCE KB0: CCL PIP-=[1,2]PIP.SAV;8208
FORCE KB0: CCL PL-EASE=[1,2]PLEASE.*;PRIV 30000
FORCE KB0: CCL QU-EUE=[1,2]QUE.*;PRIV 30000
FORCE KB0: CCL RST-=[1,2]RMSRST.TSK;PRIV 0
FORCE KB0: CCL SE-T=[1,2]TTYSET.*;PRIV 30000
FORCE KB0: CCL SRT-=[1,2]SORT.TSK;0
FORCE KB0: CCL SU-BMIT=[1,2]QUE.*;PRIV 30000
FORCE KB0: CCL SW-ITCH=[1,2]SWITCH.*;PRIV 30000
FORCE KB0: CCL SY-STAT=[1,2]SYSTAT.*;PRIV 30000
FORCE KB0: CCL TKB-=[1,2]TKB.TSK;0
FORCE KB0: CCL UT-ILTY=[1,2]UTILTY.*;30000
FORCE KB0: EXIT
FORCE KB0: BYE/F
ATTACH
```

It is most convenient to install Concise Command Language (CCL) commands on the system at the start of timesharing.

The abbreviations, shown in the definitions of the CCL commands, reflect the way the RSTS/E documentation describes the commands. You as the system manager can redefine the abbreviations but must not alter the fully defined command because each program is coded to recognize the fully expanded command.

NOTE

CCL commands are optional on all systems, especially if the DIGITAL Command Language (DCL) keyboard monitor is present. One system program, UMOUNT, can be run only by the MOUNT and DISMOUNT CCL commands. To prevent users from running UMOUNT, do not include MOUNT and DISMOUNT as Concise Command Language commands. (The system manager can always run UTILTY to mount and dismount disks.) To prevent nonprivileged DCL users from mounting and dismounting disks, change the protection code of ONLCLN.SAV to 124.

The CCL.CMD file on your distribution medium contains the lines:

```
CCL DCL-=[0,1]DCL.DCL;PRIV 0
CCL CCL-=[0,1]DCL.DCL;PRIV 0
```

Do not include these lines in the command file if you do not want DCL on your system. You should, however, leave them in the command file to allow users to:

1. Issue a DCL command without switching into DCL (by typing "DCL" and the command, followed by pressing the RETURN key).
2. Switch into DCL from another keyboard monitor (by typing "DCL" and then pressing the RETURN key).
3. Issue a CCL command from any keyboard monitor (by typing "CCL" and the command, followed by pressing the RETURN key).

   Note that typing "CCL" before the name of a CCL command is only required in DCL when a DCL and a CCL command have the same name.

By leaving the commands in the command file, you allow users this flexibility, whether they are in DCL or another keyboard monitor. For example, if you install MOUNT as a CCL on your system, users can issue a command by typing "CCL MOUNT".

You can type selected DCL commands from other keyboard monitors without the DCL prefix. For example, you can define COPY as a command that can be issued from any keyboard monitor, without the need for a prefix. To do this, define each of the selected commands as a CCL command, which

invokes [0,1]DCL.DCL. For example, the following UTILTY command defines COPY as a CCL command:

```
CCL CO-PY=[0,1]DCL.DCL;PRIV 0
```

Note that the abbreviation point should be the same as the one used by DCL itself. For most DCL commands, the abbreviation point follows the second character, but for some the abbreviation point follows the third or fourth character, to assure a unique abbreviation for each command.

Use caution when you define DCL commands in this way. You may make features of other keyboard monitors unavailable. This may happen if, for example, you define PRINT as a CCL command. BASIC-PLUS users will then not be able to use the immediate mode PRINT statement.

Two CCL commands duplicate commands the BASIC-PLUS keyboard monitor recognizes: HELLO and BYE. The commands are shown here to point out what BASIC-PLUS does by default. Other keyboard monitors may not recognize the commands HELLO and BYE if they do not exist as CCL commands.

Each CCL definition requires one buffer. The buffer will be allocated from the FIP small buffer pool. If this buffer pool is full, however, the monitor allocates the buffer for the CCL definition from the general buffer pool.

3.1.7 System Start-Up Control File — START.CTL

A complete system start-up control file contains the required INIT commands to perform initialization for timesharing and the proper indirect command file specifications to perform routine operations. The following sample shows a complete start-up control file:

```
@RTS.CMD
@TTY.CMD
@SPOOL.CMD
@CCL.CMD
FORCE KB0: RUN $ERRINT
FORCE KB0: 100
FORCE KB0: NO
LOGINS
SEND RSTS/E IS NOW ON THE AIR...
END
```

NOTE

The start-up control files must be on the system disk (SY0:, the disk that was bootstrapped).

The sample control file specifies indirect command files that handle routine operations. When INIT finishes processing all routine operations, it forces commands to run ERRINT on keyboard unit 0 and enables logins. The ON THE AIR message notifies users that timesharing has begun. The END command properly terminates INIT, and leaves keyboard unit 0 logged in to the

system so that the text forced to its input buffers will properly run ERRINT. ERRINT detaches from keyboard unit 0, thus freeing it for other uses. ERRINT then chains to ERRCPY which takes job number 1, previously used by INIT.

3.1.8 System Crash Control File — CRASH.CTL

In performing a start-up during crash recovery, the system runs INIT, which automatically accesses the file CRASH.CTL in account [1,2] on the system disk (SY0:). An example of a crash control file follows:

```
@RTS.CMD
@ANALYS.CMD
@CLEAN.CMD
@TTY.CMD
@SPOOL.CMD
@CCL.CMD
FORCE KB0: QUE ANALYS.DMP/DE
FORCE KB0: RUN $ERRINT
FORCE KB0: 100
FORCE KB0: YES
LOGINS
SEND RSTS/E IS NOW ON THE AIR...
END
```

Run the ANALYS program immediately after a system crash to recover valuable diagnostic information. An example of a ANALYS.CMD file follows:

```
DETACH
LOGIN KB0: [1,2]
FORCE KB0: RUN $ANALYS
FORCE KB0: [0,1]CRASH.SYS
FORCE KB0: ANALYS.DMP
FORCE KB0: $ERRCRS.FIL
FORCE KB0: BYE/F
ATTACH
```

Refer to Chapter 6 for a description of the ANALYS program.

Run the ONLCLN program to clean other disks on the system. An example of a CLEAN.CMD command file follows:

```
DETACH
LOGIN KB0: [1,2]
FORCE KB0: RUN $UTILTY
FORCE KB0: ENABLE CACHE/ALL
FORCE KB0: ADD RT11
FORCE KB0: EXIT
FORCE KB0: RUN $ONLCLN
FORCE KB0: DR1:
FORCE KB0: ^C
FORCE KB0: RUN $ONLCLN
FORCE KB0: DM0:
FORCE KB0: ^C
FORCE KB0: RUN $ONLCLN
FORCE KB0: DM1:
FORCE KB0: ^C
```

(continued on next page)

```
FORCE KB0: RUN $UTILTY
FORCE KB0: REMOVE RT11
FORCE KB0: ENABLE CACHE/FILE
FORCE KB0: EXIT
FORCE KB0: BYE/F
ATTACH
```

Refer to Chapter 7 for a description of the ONLCLN program.

3.2 Performing System Shutdown — SHUTUP

The shutdown procedures for the RSTS/E system are critically important. If you are not careful in performing system shutdown, valuable user data can be irretrievably lost. To fully understand shutdown procedures, you must understand other RSTS/E system procedures. You should study some of the concepts described in Chapter 7.

3.2.1 SHUTUP Tasks

To verify that conditions are suitable for shutting down the RSTS/E system, you should use any of the available system utility programs, such as:

- SYSTAT, described in the *RSTS/E System User's Guide*
- VT50PY, the system status display program described in Chapter 7

By running either of these programs, you can determine what jobs are active on the system and what disks and other peripheral devices are in use.

You can run the SHUTUP program only from KB0:, the system console terminal. If you try to run SHUTUP on any other keyboard, the program displays the following message:

```
? 'SHUTUP' can only be run from console terminal (KB0:)
Aborting shutdown run  --  Please try again later
```

SHUTUP must be stored in its compiled form in the system library account [1,2] with a protection code of <124>. The SHUTUP program can run only if it is compiled (not in .BAS form) and if it is running under the primary run-time system. If your RSTS/E system has two or more run-time systems, the operator should make sure the SHUTUP program has been compiled under the primary run-time system.

SHUTUP operates in either of two modes:

1. Without OPSER: In this mode, regardless of whether OPSER is running, the SHUTUP program treats all jobs in the system alike during the shutdown procedure (except itself, the DECnet/E EVTLOG program, the ACTLOG program, and ERRCPY).
2. With OPSER: In this mode SHUTUP allows OPSER to shut down its controlled jobs before SHUTUP continues with the regular shutdown procedures.

You should be familiar with the OPSER program and how it controls various utility programs (such as BACKUP and the spooling package programs).

As SHUTUP runs, it proceeds through thirteen phases:

1. Set-up dialogue
2. Warning message
3. DECnet/E shutdown (if necessary)
4. Initial job killing
5. OPSER shutdown (if necessary)
6. EVTLOG shutdown (if necessary)
7. ERRCPY shutdown (if necessary)
8. Final job killing (if necessary)
9. EMT logging shutdown (if necessary)
10. Unload and remove run-time systems and resident libraries
11. Swap file removal
12. Disk dismount
13. Final shutdown

The following sections describe these phases, the operator interactions required, and both the expected and unexpected results of the shutdown operation. Several sample runs of SHUTUP show the results of shutting down the system with and without OPSER.

3.2.2 Set-Up Dialogue Phase

The set-up dialogue phase processes all questions relating to the selection of options in running SHUTUP. It also checks for the presence of OPSER.

In response to questions that appear during this phase, you can press the ESCAPE (or ALTMODE) key to return to the previous question. Otherwise, you should terminate your responses by pressing RETURN.

If OPSER is present and running, SHUTUP prints the following question:

```
Use 'OPSER' for utilities shutdown (YES/NO) <YES>?
```

Press the RETURN key or type YES, to have SHUTUP communicate with OPSER during the shutdown procedures. If you type N, SHUTUP prints another question to confirm your response:

```
Are you sure you don't want to use 'OPSER' (YES/NO) <NO>?
```

Pressing the RETURN key or typing NO causes the program to repeat the previous question so that you can again decide to use or not to use OPSER during shut down. Otherwise, the program ignores OPSER during the shutdown procedures.

If you choose to use the OPSER program, SHUTUP prints the following additional question:

```
Allow utilities to reach logical end point (YES/NO) <YES>?
```

If you press RETURN or type YES, the program directs OPSER to tell its on-line jobs to shut down their operations at the next logical breakpoint in their job streams. For a line printer spooler, for example, this breakpoint occurs between queued jobs.

If your response is NO, the program directs OPSER to tell its on-line jobs to immediately abort and cease operations. You should abort in this way only in cases of emergency; jobs being processed at the time of shutdown may be lost (unless you have previously halted queuing and spooling operations and requeued these jobs).

SHUTUP then asks you to specify the number of minutes you want to elapse before shutdown occurs:

```
Minutes until system shutdown (0-99) <5>?
```

This waiting period is called the warning message phase (described below) and may be as short as 0 minutes or as long as 99 minutes. Press the RETURN key to select the default value of 5 minutes. If you decide to bring the system down immediately by typing 0, SHUTUP does not ask the two questions about disabling logins and the DECnet/E network.

When you have specified a shutdown period greater than 0 minutes, SHUTUP allows you to select the amount of time that will elapse before further logins are disabled:

```
Minutes until logins are disabled (0-99) <0>?
```

The question gives you the opportunity to select the amount of time during which users can continue to log in to the system as shutdown occurs. For the number of minutes you specify, users can log in to and log out of the system, as if the system were not shutting down at all. The ability to allow users this flexibility is particularly useful when a long shutdown period has been specified. There will be times when you do not want any more users on the system. In this case, accept the default response by pressing the RETURN key. Users can no longer log in to the system but those already logged in remain unaffected.

The last message in the set-up dialogue phase allows you to choose the number of minutes that will elapse before new network activity is disabled:

```
Minutes until new network activity is disabled (0-99) <nn>?
```

The number of minutes SHUTUP selects for the default value <nn> depends on your response to the system shutdown question. As in the previous question, being able to specify when the network will shut down allows users time to complete network tasks. If you choose to allow no further logical links to be created, type 0, and then press the RETURN key. Those still using the network can continue to work but no new users are allowed access to the network.

3.2.3 Warning Message Phase

At the beginning of the warning message phase, as well as in the beginning of all other phases that SHUTUP may execute, a message appears on the system console terminal (KB0:) in the following format:

```
hh:mm AM dd-mmm-yy
      PM        ********  <phase title>  ********
```

SHUTUP prints:

- The time and date with each phase title message
- The hour (hh) and the minute (mm) that it issues the message in the format hh:mm
- Whether the message appeared before or after noon, AM or PM
- The day, month, and year in the date format, dd-mmm-yy

SHUTUP prints the following message on the system console terminal when the time specified for disabling logins has elapsed:

```
Further LOGINs are now disabled
```

The RSTS/E shutdown procedure requires that the system limits on logins be set to 1 before the monitor can perform the final shutdown operation. SHUTUP lowers the login limit at the start of the warning message phase to prevent any new users from logging on to the system.

During the following phases, SHUTUP continuously checks that the logins limit remains 1. If, for any reason, the limit changes, SHUTUP immediately aborts all operations with the following message:

```
LOGINS ARE NOT EQUAL TO 1
ABORTING SHUTDOWN RUN  --  PLEASE TRY AGAIN LATER
```

You should determine the cause of the change, correct the problem, and run SHUTUP again.

If your system has DECnet/E support, the SHUTUP program prints the following message when the time specified for disabling network activity has expired:

```
Further network activity is now disabled
```

The message tells you that no new logical links can be formed. Those links that are already active, however, can remain active until the time to network shutdown has elapsed. Before the network shuts down, the DECnet/E program EVTLOG may print messages indicating the status of the network. The initial EVTLOG message marks the beginning of the network shutdown phase and the second, issued later in the shutdown process, indicates that the network has been shut off completely. (Refer to Section 3.2.14.2 for an example of these messages.)

During the warning message phase, the program issues warning messages indicating the system is shutting down. These messages appear on most keyboards in the system, including pseudo keyboards. The following messages are the exceptions:

```
The OPSER operator services console (OSC)
The system console terminal (KB0:)
```

The program issues warnings after an appropriate waiting period, defined by the following formula:

waiting time = (total time left)5 + 1 minutes

For an initial waiting period of 60 minutes, warning messages appear at 60, 47, 37, 28, 22, 17, 13, 10, 7, 5, 3, 2, and 1 minute before shutdown begins. SHUTUP prints the following message on all system terminals at each warning time:

hh:mm AM dd-mmm-yy
PM System going down in <waiting time> minutes, please finish up

At the same time, the following message appears on the system console terminal (KB0:):

<waiting time> minute message sent

When the time to shutdown has expired or if the initial waiting period is specified as 0, the final warning message appears:

hh:mm AM dd-mmm-yy
PM **** FINAL WARNING!!!!! System shutting down ****

At this point, the program enters the next phase. The time and date are printed for both the SYSTEM GOING DOWN and the FINAL WARNING messages. The hh and mm in the time format, hh:mm, represent the hour and minute the message was printed. Whether the message appeared before or after noon is indicated by the AM and PM abbreviations. The day, month, and year are included in the date format, dd-mmm-yy.

3.2.4 DECnet/E Shutdown Phase (If Necessary)

The time to DECnet/E shutdown begins as soon as you enter the number of minutes that should elapse before new network activity is disabled:

```
Minutes until new network activity is disabled (0-99) <nn>?
```

The number of minutes <nn>, shown in angle brackets, is the default response. It assumes the same value you typed in response to the SHUTUP question:

```
Minutes until system shutdown (0-99) <nn>?
```

Rather than accept the default response (that is, to the network activity question) and have new network activity disabled at the same time the system shuts down, it is better to disable new network activity before shutdown occurs. Entering a value less than the number of minutes to shutdown establishes a period before shutdown in which no new links can be created and allows enough time for those already active to complete network tasks. If you want new network activity disabled immediately, type 0 and press the RETURN key. When the time to disable network activity has elapsed (0 to 99 minutes), SHUTUP prints a message to indicate no further network activity is allowed:

```
Further network activity disabled
```

SHUTUP disables new network activity by executing the equivalent of the Network Control Program (NCP) command SET EXECUTOR STATE SHUT. This prevents the creation of new logical links but does not affect existing logical links. In essence, new network links can be created until the network activity is disabled. After that period ends, no new links can be created, but users already on the net can continue working until the system shuts down. The network remains up until there are no links active or until the system shuts down.

As soon as all logical links have been disconnected by their users, the network state changes automatically to OFF. The state is immediately set to OFF if no links existed at the time SHUTUP disabled new network activity. Both the transition to SHUT and the one to OFF are logged by the DECnet/E event logging program EVTLOG, if it is on and has event 2.0 enabled. (See the *DECnet/E System Manager's Guide* for a description of EVTLOG.)

SHUTUP never sends any messages indicating the approach of a DECnet/E shutdown. Only from SHUTUP broadcasts to all terminals and from the fact that the network is in the SHUT instead of the ON state can users tell that SHUTUP is shutting down the system.

If at the end of the warning message phase the network has still not completely shut down, because active links remain, SHUTUP explicitly sets the network to OFF. This disconnects all remaining logical links. SHUTUP then waits 15 seconds to allow affected jobs to complete network tasks and exit, before it enters the job killing phase.

Should you decide not to shut the system down and thus not shut off the network after you have started SHUTUP, pressing a CTRL/C to stop the execution of SHUTUP may reactivate the network. A CTRL/C causes SHUTUP to print a message telling you the state in which the network was left. If the

network is not in the state you want it, you must use NCP to place the network and the event logger (EVTLOG) in the desired state. Refer to the *DECnet/E System Manager's Guide* for a description of NCP.

3.2.5 Initial Job Killing Phase

In the initial job killing phase, SHUTUP begins to clear the system of currently active jobs. If SHUTUP finds there is only one job (itself) running on the system, it skips this phase and then continues the shutdown procedure at the unload and remove run-time systems and resident libraries phase (Section 3.2.10).

If SHUTUP is using OPSER in the next phase (OPSER shutdown phase, Section 3.2.6), the program ignores, during this phase, all jobs currently associated with OPSER (as indicated by entries in OPSER's on-line job table). It also ignores any job whose primary keyboard is a pseudo keyboard because it is probably being controlled by one of the BATCH programs.

SHUTUP further divides all other jobs active in the system into two classes: attached and detached. This classification depends on whether they have an attached primary keyboard (KB:). For all attached jobs, SHUTUP now forces the following text string into the keyboard input buffer:

```
^C  ^C  BYE/Y
```

For all detached jobs, SHUTUP issues the kill job SYS call to remove the job.

SHUTUP makes two passes through the current active job table during this phase. In the first pass, the program terminates in the proper way all active jobs not being ignored. At the end of the first pass, it establishes the number of attached jobs that were found and forced to logout. SHUTUP then waits for a specified period to allow the LOGOUT program to complete operations on each of the attached jobs.

After the waiting period (if any) expires, the program makes a second pass through the active job table. During this pass, all jobs not being ignored are removed with the kill job SYS call. At the end of the pass, the program checks the table. If any jobs that should have been killed were not, SHUTUP aborts operations with the following error message:

```
'SHUTUP' Failed in initial Job Killing Phase
Aborting shutdown run  --  Please try again later
```

You should determine why SHUTUP could not kill all jobs, correct the problem, and run SHUTUP again.

If SHUTUP successfully stops all jobs during the phase, or if at any time during either pass the number of active jobs becomes 1 (SHUTUP only), the next phase begins.

3.2.6 OPSER Shutdown Phase (If Necessary)

The program skips the OPSER shutdown phase if: (1) OPSER is not running on the system or (2) you elect not to use OPSER during the shutdown procedures.

If you choose to use OPSER, SHUTUP immediately detaches from the system console terminal (KB0:) to enable OPSER to reattach to the terminal for use during OPSER's shutdown procedures. As the detach occurs, the following messages appear:

```
DETACHING...
'OPSER' ATTACHING
#
```

SHUTUP sends a message to OPSER directing it to begin its shutdown procedures by selecting the appropriate mode of OPSER shutdown. SHUTUP then waits for 60 seconds to see if OPSER receives the message and/or is properly functioning. If OPSER does not respond properly within that time, SHUTUP attempts to reattach to the system console terminal (KB0:). If it is successful, the following message appears:

```
'OPSER' not active
Aborting shutdown run  --  Please try again later
```

If the terminal is attached or assigned, SHUTUP waits for 1 second and tries again. You should take whatever steps are necessary to free the system console terminal (KB0:) for use so SHUTUP can complete its activities.

If OPSER is shutting down its on-line programs in the immediate mode, it sends each program the appropriate message to cease operations. OPSER waits a certain amount of time to make sure all programs have completed their assignments. OPSER then closes its files, prints the following message on the system console (KB0:), and stops execution:

```
'OPSER' TERMINATING
```

SHUTUP allows OPSER 120 seconds to complete its shutdown in the immediate shutdown mode. If unsuccessful, SHUTUP aborts its operations, signaling the failure as follows:

```
'OPSER' Shutdown taking too long
'SHUTUP' Aborting shutdown operation
```

If OPSER is shutting down in the logical end mode, it proceeds to shutdown successive levels in its on-line job table. Appropriate system console messages (see Section 3.2.14) signal the end of each job. When all on-line OPSER jobs are gone, OPSER closes its files and kills itself, freeing the system console for SHUTUP use again. OPSER tells you when it finishes processing:

```
'OPSER' TERMINATING
```

SHUTUP allows 60 minutes in the logical end mode for OPSER to complete operations. If the procedure is not successful, SHUTUP aborts operations with the error message described previously. Otherwise, the shutdown of OPSER-related jobs and activities is complete and the program enters the next phase.

3.2.7 EVTLOG Shutdown Phase (If Necessary)

The SHUTUP program enters the EVTLOG shutdown phase if it finds "EVTLOG" in the message/receiver table. (DECnet/E does not need to be present for SHUTUP to perform this phase.) SHUTUP sends EVTLOG a special error message that causes EVTLOG to log the shutdown occurrence on the system console, perform various other shutdown operations, and kill itself.

When SHUTUP detects that EVTLOG has completed its shutdown activities and is no longer present, SHUTUP proceeds to the next phase. SHUTUP allows 60 seconds for EVTLOG to complete its tasks, and if failure occurs, SHUTUP aborts operations with the following error message:

```
'EVTLOG' Failed to shutdown
Aborting shutdown run  --  Please try again later
```

Refer to the *DECnet/E System Manager's Guide* for information about the EVTLOG program.

3.2.8 ERRCPY Shutdown Phase (If Necessary)

The program enters the ERRCPY shutdown phase if the system error logging utility ERRCPY is present. SHUTUP sends ERRCPY a special error message (SH error) which causes ERRCPY to log the shutdown occurrence in the system error log, close the file, and kill itself.

When SHUTUP detects that ERRCPY has completed its shutdown and is no longer present, SHUTUP proceeds to the next phase. SHUTUP allows 60 seconds for ERRCPY to complete its tasks, and if failure occurs, SHUTUP aborts operations with the following error message:

```
'ERRCPY' Failed to shutdown
Aborting shutdown run  --  Please try again later
```

3.2.9 Final Job Killing Phase (If Necessary)

If SHUTUP reaches this point, and is not the only job left running in the system, SHUTUP makes one last attempt at killing all other remaining jobs. If SHUTUP is successful, it enters the next phase. If not successful, SHUTUP aborts operations with the following error message:

```
'SHUTUP' Failed in final Job Killing Phase
Aborting shutdown run  --  Please try again later
```

3.2.10 EMT Logging Shutdown Phase (If Necessary)

The program enters the EMT logging shutdown phase if the EMT logging utility is present. (See Chapter 7 for a description of the EMT logger.) SHUTUP sends a shutdown message to the EMT logger, informing the EMT logger to stop. If the EMT logger does not shut down in 60 seconds, SHUTUP aborts operations with the following error message:

```
EMT logging failed to shutdown.  Killing EMT logger job = n
```

In this message, "n" is the job number of the EMT logger.

3.2.11 Unload and Remove Run-Time Systems and Resident Libraries Phase

During the unload and remove run-time system and resident libraries phase, the program unloads from memory all run-time systems and resident libraries. It then removes them from the system tables. SHUTUP does not attempt to remove the system default run-time system.

3.2.12 Swap File Removal Phase

During the swap file removal phase, SHUTUP makes sure all open swap files are closed and released from the system.

3.2.13 Disk Dismount Phase

During the dismount phase, the program dismounts all mounted disks, both private and public, from the system. The system disk always remains mounted.

If you run SHUTUP in its source form (.BAS) by mistake, SHUTUP may encounter an unexpected error. If the temporary file (TEMPnn.TMP), created by the BASIC-PLUS run-time system and associated with SHUTUP, resides on a public disk that is not the system disk, an unexpected error occurs when SHUTUP tries to dismount the affected disk. SHUTUP aborts and you must compile the program before retrying the shutdown procedure from the beginning.

3.2.14 Final Shutdown Phase

SHUTUP enters the final shutdown phase to output the last status message, clears the system console terminal (KB0:) buffers, and executes the special system shutdown SYS call. The monitor transfers control to the system initialization code (INIT.SYS) and bootstraps the system disk. The initialization code is loaded and INIT prints the OPTION prompt.

3.2.15 SHUTUP Operation Examples

Each of the following sections contains an example of a SHUTUP operation. With these examples and the description of the SHUTUP program, you should be able to shut down your system in a correct and orderly fashion.

3.2.15.1 SHUTUP Example — OPSER Not Present — The following example shows a system shutdown when OPSER is not present. Except for the absence of the first OPSER-related question, the example applies when OPSER is present but not used during shutdown.

```
RUN $SHUTUP
SHUTUP V8 RSTS V8 TIMESHARING

12:24 PM 22-Dec-82  ********  Set-up   Dialogue   Phase  ********

Type 'ESC' ('ALT') to any query to backup one (1) step

'OPSER' not running

Minutes until system shutdown (0-99) <5>?  0

12:24 PM 22-Dec-82  ********  Warning   Message    Phase  ********
Further LOGINs are now disabled

12:24 PM 22-Dec-82  ********  Initial Job Killing Phase  ********

12:24 PM 22-Dec-82  ********  EMT Logging Shutdown Phase ********

12:24 PM 22-Dec-82  ********  Remove RTS/RES LIB  Phase  ********

12:24 PM 22-Dec-82  ********  SWAP File  Removal  Phase  ********

12:24 PM 22-Dec-82  ********  Disk     DISMOUNT   Phase  ********

12:24 PM 22-Dec-82  ********  Final    Shutdown   Phase  ********

Please wait for system to re-boot itself

RSTS V8 TIMESHARING

Option:
```

3.2.15.2 SHUTUP Example — DECnet/E Shutdown — This example shows SHUTUP when OPSER is used to shut down a spooling system of five spoolers. It also illustrates how the network shuts down when a system includes DECnet/E support and event logging by the DECnet/E EVTLOG program has been enabled.

```
RUN $SHUTUP
SHUTUP V8 RSTS V8 TIMESHARING

12:39 PM 22-Dec-82  ********  Set-up   Dialogue   Phase  ********

Type 'ESC'('ALT') to any query to backup one (1) step

Use 'OPSER' for utilities shutdown (YES/NO) <YES>?
Allow utilities to reach logical end point (YES/NO) <YES>?
Minutes until system shutdown (0-99) <5>?
Minutes until logins are disabled (0-99) <0>?
Minutes until new network activity is disabled (0-99) <5>? 0
```

(continued on next page)

```
12:39 PM 22-Dec-82  ********  Warning   Message   Phase  ********
Further LOGINs are now disabled
Further network activity is now disabled


Event type 2.0, local node state change
Occurred 22-Dec-82 12:39:05.0 on node 135 (OTHG)
Reason for state change: Operator command
Old node state = On
New Node state = Shut

5 minute warning message sent
3 minute warning message sent


Event type 2.0, local node state change
Occurred 22-Dec-82 12:41:55.0 on node 135 (OTHG)
Reason for state change: Normal command
Old node state = Shut
New Node state = Off

2 minute warning message sent
1 minute warning message sent

12:45 PM 22-Dec-82  ********  DECNET   Shutdown   Phase  ********

12:45 PM 22-Dec-82  ********  Initial Job Killing Phase  ********

12:45 PM 22-Dec-82  ********  ' OPSER ' Shutdown  Phase  ********
Detaching...


'OPSER' ATTACHING
#
JOB #6 'BA1SPL' TAKEN OFFLINE
#
JOB #5 'BA0SPL' TAKEN OFFLINE
#
MESSAGE             28 : 22-Dec-82 12:45 PM  JOB:3  DET    QUMRUN[1,10]
        BA1SPL (6) REQUESTED OFF-LINE -- TAKEN OFF-LINE
#
MESSAGE             29 : 22-Dec-82 12:45 PM  JOB:3  DET    QUMRUN[1,10]
        BA0SPL (5) REQUESTED OFF-LINE -- TAKEN OFF-LINE
#
JOB #4 'LP0SPL' TAKEN OFFLINE
#
MESSAGE             30 : 22-Dec-82 12:45 PM  JOB:3  DET    QUMRUN[1,10]
        LP0SPL (4) REQUESTED OFF-LINE -- TAKEN OFF-LINE
#
JOB #3 'QUEMAN' TAKEN OFFLINE
#'OPSER' TERMINATING


Re-attaching...


12:45 PM 22-Dec-82  ********  ' EVTLOG ' Shutdown Phase  ********

Shutting down EVTLOG by operator request.

12:45 PM 22-Dec-82  ********  ' ERRCPY ' Shutdown Phase  ********

12:45 PM 22-Dec-82  ********  Remove RTS/RES LIB  Phase  ********

12:45 PM 22-Dec-82  ********  SWAP File  Removal  Phase  ********
```

(continued on next page)

```
12:45 PM 22-Dec-82  ********  Disk     DISMOUNT     Phase  ********

12:45 PM 22-Dec-82  ********  Final    Shutdown     Phase  ********

Please wait for system to re-boot itself

RSTS V8 TIMESHARING

Option:
```

3.2.15.3 SHUTUP Example — Spooling Shutdown — In this example, OPSER is used to shut down the spooling system in immediate mode.

```
RUN $SHUTUP
SHUTUP V8 RSTS V8 TIMESHARING

12:11 PM 22-Dec-82 ********  Set-up   Dialogue   Phase  ********

Type 'ESC'('ALT') to any query to backup one (1) step

Use 'OPSER' for utilities shutdown (YES/NO) <YES>?
Allow utilities to reach logical end point (YES/NO) <YES>? NO
Minutes until system shutdown (0-99) <5>?
Minutes until logins are disabled (0-99) <5>? 0

12:11 PM 22-Dec-82  ********  Warning   Message    Phase  ********
Further LOGINs are now disabled
5 minute warning message sent
3 minute warning message sent
2 minute warning message sent
1 minute warning message sent

12:12 PM 22-Dec-82  ********  Initial Job Killing Phase  ********

12:12 PM 22-Dec-82  ********  ' OPSER ' Shutdown  Phase  ********
Detaching...


'OPSER' ATTACHING
#
'OPSER' TERMINATING


Re-attaching...

12:12 PM 22-Dec-82  ********  ' ERRCPY ' Shutdown Phase  ********

12:12 PM 22-Dec-82  ********  Final Job  Killing  Phase  ********

12:12 PM 22-Dec-82  ********  EMT Logging Shutdown Phase ********

12:12 PM 22-Dec-82  ********  Remove RTS/RES LIB  Phase  ********

12:12 PM 22-Dec-82  ********  SWAP File  Removal  Phase  ********

12:12 PM 22-Dec-82  ********  Disk     DISMOUNT     Phase  ********

12:12 PM 22-Dec-82  ********  Final    Shutdown     Phase  ********

Please wait for system to re-boot itself

RSTS V8 TIMESHARING

Option:
```

3.2.15.4 SHUTUP Example — Normal Shutdown — The system in the following example is being shut down in a leisurely fashion. This gives system and network users enough time to complete their tasks. SHUTUP allows users to log in to the system during the entire shutdown phase; logins are disabled at system shutdown, after 15 minutes. Those using the network can log in until the ten minute mark, after which network access by users is not allowed. Those on the network before new network activity is disabled, however, can continue to work until the system shuts down completely.

```
RUN $SHUTUP
SHUTUP V8 RSTS V8 TIMESHARING

10:11 AM 28-Oct-82  ********  Set-up   Dialogue   Phase  ********

Type 'ESC'('ALT') to any query to backup one (1) step

Use 'OPSER' for utilities shutdown (YES/NO) <YES>? YES
Allow utilities to reach logical end point (YES/NO) <YES>? YES
Minutes until system shutdown (0-99) <5>? 15
Minutes until logins are disabled (0-99) <0>? 15
Minutes until new network activity is disabled (0-99) <5>? 10

10:11 AM 28-Oct-82  ********  Warning   Message   Phase  ********
15 minute warning message sent
11 minute warning message sent
8 minute warning message sent
6 minute warning message sent
4 minute warning message sent
3 minute warning message sent
2 minute warning message sent
Further LOGINs are now disabled
1 minute warning message sent

10:18 AM 28-Oct-82  ********  DECNET   Shutdown   Phase  ********

10:18 AM 28-Oct-82  ********  Initial Job Killing Phase  ********

10:19 AM 28-Oct-82  ********  ' OPSER ' Shutdown  Phase  ********
Detaching...


'OPSER' ATTACHING
#
JOB #5 'BA0SPL' TAKEN OFFLINE
#
JOB #6 'BA1SPL' TAKEN OFFLINE
#
MESSAGE            293 : 28-Oct-82 10:19 AM  JOB:3  DET     QUMRUN[1,9]
        BA0SPL (5) REQUESTED OFF-LINE -- TAKEN OFF-LINE
#
MESSAGE            294 : 28-Oct-82 10:19 AM  JOB:3  DET     QUMRUN[1,9]
        BA0SPL (6) REQUESTED OFF-LINE -- TAKEN OFF-LINE
#
JOB #4 'LP0SPL' TAKEN OFFLINE
#
MESSAGE            295 : 28-Oct-82 10:19 AM  JOB:3  KB??:  QUMRUN[1,9]
        LP0SPL (4) REQUESTED OFF-LINE -- TAKEN OFF-LINE
#
JOB #3 'QUEMAN' TAKEN OFFLINE
#'OPSER' TERMINATING
```

(continued on next page)

```
Re-attaching...

10:19 AM 28-Oct-82  ********  ' EVTLOG ' Shutdown Phase  ********

Shutting down EVTLOG by operator request.

10:19 AM 28-Oct-82  ********  ' ERRCPY ' Shutdown Phase  ********

10:19 AM 28-Oct-82  ********  Remove RTS/RES LIB  Phase  ********

10:19 AM 28-Oct-82  ********  SWAP File  Removal  Phase  ********

10:19 AM 28-Oct-82  ********  Disk    DISMOUNT    Phase  ********

10:19 AM 28-Oct-82  ********  Final   Shutdown    Phase  ********

Please wait for system to re-boot itself

RSTS V8 TIMESHARING

Option:
```

3.2.16 Notes on SHUTUP Operation

The following notes can help you manage the operation of the SHUTUP program on your system.

- The SHUTUP program shuts down a RSTS/E system in an orderly manner if a compiled version (not in .BAS form) of SHUTUP is running under the primary run-time system. Be sure, if your system has two or more run-time systems, to compile the SHUTUP program under the primary run-time system.
- It is helpful to the users on your system if you establish administrative procedures governing the hours the system is in operation. If you have a fixed shutdown schedule, users can plan their work load and properly complete tasks in the allotted hours of timesharing. You can keep them informed by placing messages in the NOTICE.TXT file. The system prints the contents of NOTICE.TXT each time users logs in. It is, therefore, a convenient way to make sure all users know about important system operations.

Chapter 4
Account Creation and Account Statistics

This chapter describes how to create and delete accounts with the REACT program. The chapter also includes a description of the MONEY system program that lists the account information your system retains.

4.1 Creating and Deleting User Accounts with REACT

The REACT program allows you or other privileged users to create and delete accounts from the disk devices on your system. You run REACT with the RUN command by typing:

```
$ RUN $REACT(RET)
```

After you press the RETURN key, the REACT program prints a line of information that includes the name of the program, the RSTS/E version number, and the name given the system at system generation time. The program then prompts you for one of three functions described in Table 4–1: ENTER, DELETE, or STANDARD. The output from the REACT program is:

```
REACT V8 RSTS V8 TIMESHARING
System Account Manager
Function?
```

Enter one of the single letters from Table 4–1, press the RETURN key, and wait until the REACT program presents you with a set of dialogue questions. The dialogue you receive depends on the function you choose.

Table 4–1: REACT System Program Functions

Function	Abbreviation and Purpose
ENTER	E Creates an account on a disk.
DELETE	D Deletes an account from a disk.
STANDARD	S Creates standard user accounts on the system disk from the ACCT.SYS file at system generation time.

The next three sections describe these REACT functions.

4.1.1 Adding Individual Accounts with the ENTER Function

The ENTER function allows you to create user accounts on either the system disk or private disks. Users can then create files on the disks that contain their accounts. For a user to gain access to a private disk, the pack must be logically mounted and unlocked by means of the UTILTY or UMOUNT system program. Refer to Chapter 7 for a description of the UTILTY program and to the *RSTS/E System User's Guide* for information on UMOUNT. These procedures are unnecessary if you are adding accounts to the system disk. It is mounted and on line, ready for immediate use.

To add an account, type the letter E in response to the FUNCTION prompt. REACT recognizes your response and prints a series of questions at your terminal:

```
$ RUN $REACT (RET)
REACT V8 RSTS V8 TIMESHARING
System Account Manager
Function? E (RET)
Proj,Prog? 125,125 (RET)
Disk:Password? TIME (RET)
Quota <0>? 1000 (RET)
Cluster Size <0>? 4 (RET)
Number of Clusters (0-7) <1>? 5 (RET)
Position <0>? 1200 (RET)
Account Name? GEORGE POLK (RET)
Account _SY0:[125,125] created at DCNs 32000 32001 32002 32003 32004
Proj,Prog? (CTRL/Z)
```

Table 4–2 describes the responses you can make during the ENTER dialogue.

Table 4–2: Responses to ENTER Function Questions

Question	Response Format and Meaning
Proj,Prog?	*proj,prog* Enter the new user account number. The project number *proj* can be any number from 1 to 254. The programmer number *prog* can be any number from 0 to 254. After you run the dialogue to enter an account, REACT returns to this question for the next account. Type CTRL/Z in answer to this question to stop entering accounts; that is, to terminate REACT.
Disk:Password?	*dev:* or *password* (or both) Enter a device name and unit number if are creating an account on a private disk. If you do not specify a device, the system disk is assumed. Enter a password if you are creating an account on the public disk structure. A password can consist of from one to six alphanumeric characters. If you do not specify a password, then a password of "??????" is assigned. (You cannot log in to any account whose password on the system disk contains one or more question marks.)

(continued on next page)

Table 4–2: Responses to ENTER Function Questions (Cont.)

Question	Response Format and Meaning
Quota <0>?	*n* The number *n* indicates the number of 512-byte blocks of disk storage that the user can have allocated for the account at logout. The quota *n* can range from 0 to 65535. A range of 0 means that no limit is imposed on disk storage for the account.
Cluster Size <0>?	*n* The number *n* defines the cluster size of the User File Directory (UFD) for the account. This number must be greater than or equal to the disk pack cluster size (defined when the disk was initialized). Possible values are 0, 1, 2, 4, 8, or 16. A value of 0 causes REACT to use the pack cluster size. If you are not sure of the current pack cluster size, run the SYSTAT program (by typing "SY/D" from the keyboard monitor) to check. It is good practice to choose a small UFD cluster size for an account that will contain a small number of files, and a large UFD cluster size for an account that will contain a large number of files. To approximate the maximum number of files a user can create, multiply the UFD cluster size by 72.
Number of Clusters (1–7) <1> ?	*n* The number *n*, ranging from 1 through 7, defines the number of UFD clusters to be preallocated for the UFD. (The UFD cluster size is defined by REACT's "Cluster Size ?" question.) Press the RETURN key or type 0 to preallocate no space for the UFD. It is useful to preallocate space for the UFD for accounts that you know will have a large number of files. Otherwise, the monitor will allocate space for the UFD as files are added to the account, and the UFD may not be stored in contiguous disk space. This can make disk access to files slower for these accounts. Refer to Section 1.4.2 for more information on preextending directories.
Position <0> ?	*device-cluster-number* REACT places the UFD beginning at the device cluster number (DCN) specified in the response. Type –1 to place the UFD near the middle of the disk. See Section 1.4.2 for more information on positioning directories. Also, refer to Appendix E, "Disk Device Sizes," for information on device cluster numbers for various types of disks.
Account Name?	*name* REACT places the account name you enter in the ACCT.SYS file, which the GRIPE program uses to identify the owner of the account. (Do not include any commas, apostrophes, or quotation marks in the account name.)

When you answer the last question in the dialogue, REACT displays the message confirming the device and account number, followed by DCNs for the number of clusters you specify. Note that five clusters are specified in the example:

```
Number of Clusters (0-7) <1>? 5 RET
```

Therefore, five DCNs are displayed:

```
Account _SY0:[125,125] created at DCNs 32000 32001 32002 32003 32004
```

Be aware that because ACCT.SYS records the passwords of all accounts you add with REACT, you must make sure nonprivileged users do not gain access to the ACCT.SYS file. (Privileged users are always able to read the file.) To help you keep the contents of this file secret, it is created with a protection code of <188>.

4.1.2 Positioning and Preextending Directories

Positioning and preextending directories are ways to help you optimize disks. The DSKINT program (described in Chapter 7) lets you specify the same information to position and preextend directories as the REACT program for accounts [0,1], [1,1], and [1,2]. Information about all other accounts must be provided when the accounts are created with the REACT program.

Positioning directories lets you place files in locations that require a minimum of extra disk head movement. You can position a directory by specifying a location in the following prompt in the ENTER function of the REACT command:

```
Position <0> ? 3000 RET
```

In this example, a DCN of 3000 is specified. The system considers 3000 as a starting place and first looks in larger block numbers for available space. If the system does not locate free space in this search, it then looks for space in the lower block numbers. If it does not find free space in the lower block numbers, the error message "?No room for user on device" is returned. This error will not be returned if at least one cluster can be allocated to the directory.

If you specify a position that will cause the directory to exceed the limits of the disk, then you will get the error message:

```
?Illegal SYS() usage
```

Refer to Appendix E for information about disk sizes.

The following list describes the choice of values you can specify after the prompt:

1 = Place the directory as close as possible to the beginning of the disk. Block #1 is the lowest block number on the disk.

–1 = Place the directory as close as possible to the middle of the disk. (The system calculates the exact location.)

0 = Place the UFD as close to the GFD (Group File Directory) as possible. If this is the first UFD you are creating for the group, then:

1. The system will first create the GFD as close to the MFD as possible, and
2. The UFD being created will then be placed as close to the new GFD as possible.

DCN = The device cluster number where you want to position the directory.

Preextending directories can help you ensure that the directory entries for a group of files are confined to a few sequential blocks, therefore reducing the amount of disk head movement required to access files. You can preextend a directory by your response to the following prompt in the ENTER function of the REACT command:

```
Number of Clusters (0-7) <1> ? 6 RET
```

In the example, 6 clusters are specified. When you preextend a directory this way, you specify that the system should create 6 clusters first. If your response to this question is 0, the system creates the space dynamically.

There are two disadvantages to creating an account with a preextension that is too large:

1. Any unused space is essentially wasted, because the clusters are reserved for one directory only and cannot be used by other directories.
2. It takes longer to rebuild the disk if the directory is preextended. The system has to verify the entire directory, whether or not all the clusters allocated to it are used.

4.1.3 Checking to See if a Directory is Contiguous

Having a contiguous directory can help optimize the disk. Although you cannot specify a directory to be contiguous, you can check to see whether or not it was created that way.

Consider the following REACT dialogue:

```
$ RUN $REACT (RET)
REACT  V8  RSTS  V8 Timesharing
System Account Manager
Function? E (RET)
Proj,Prog? 48,48 (RET)
Disk:Password? MACHO (RET)
Quota <0> ? 1000 (RET)
Cluster Size <0> ? 8 (RET)
Number of Clusters (0-7) <1> ? 7 (RET)
Position <0> ? 5000 (RET)
Account Name? FREDDY (RET)
Account _SY0:[48,48] created at DCNs 5779 5781 5783 5785 5787 5789 5791

Proj,Prog? (CTRL/Z)
```

Notice that the system created the directory as close as possible to the specified position:

```
Position <0> ? 5000 (RET)
```

In this case, the first DCN in this example is 5779.

The following steps explain how to see whether or not the directory [48,48] was created contiguous.

1. Note the response to the prompt:

   ```
   Cluster Size <0> ? 8 (RET)
   ```

   A directory cluster size of 8 is specified in the example. (You can also check the directory cluster size with output from the MONEY command, in the column labeled UFD. See Section 4.2.2 for more information.)

2. Record the numbers displayed in the message that confirms creation of the account. These numbers are the DCNs:

```
Account SY0:[125,125] created at DCNs 5779 5781 5783 5785 5787 5789 5791
```

   Be aware that you can get these numbers only when you create an account with REACT.

3. Check the pack cluster size by running the SYSTAT program and noting the column labeled "Clu" for the device where the account was created. If you do not specify a device, the system disk (SYSDSK in the example) is assumed.

```
$ SY/D (RET)

Disk Structure:
Dsk Open     Size        Free    Clu  Err  Name     Level  Comments
DL1    0    20460     6765  33%    1    0  V72       0.0   Pri, DLW
DM1    0    53768     5342   9%    1    1  OLDDSK    0.0   Pri, DLW
DR1   49   131648    15220  11%    4    0  SYSDSK    1.1   Pub, DLW
DR2    0   242576    33040  13%    8    0  NEWDSK    1.1   Pri, R-O, DLW
DR3   22   500352    52456  10%    4    0  KNIGHT    1.1   Pri, DLW
DR4   10   242572    12832   5%    4    0  GILLUM    1.1   Pri, DLW
DR5    0   500352    76152  15%    8    0  HISTRY    0.0   Pri, R-O, DLW
```

4. The ratio of the directory cluster size to pack cluster size tells you the incremental difference in DCNs if the directory was created contiguous. In the example, the directory cluster size is 8 and the pack cluster size is 4. 8 divided by 4 is 2, which is the increment between DCNs if the directory was created contiguous. The directory was created contiguous, because the device cluster numbers differ by 2:

   ```
   5779 5781 5783 5785 5787 5789 5791
   ```

You may not always be able to place the directory where you specify, nor can you request that the directory will be contiguous. However, the message displayed by REACT lets you know where the directory was created, and whether or not it was created contiguous. If you are creating accounts on a new system disk that has a lot of free space, it is likely that the accounts will be positioned where you specify, and that the UFDs will be contiguous.

The following example shows a directory that was not created contiguous:

```
Proj,Prog? 40,40 (RET)
Disk:Password? KNIGHT:YOUHOO (RET)
Quota <0> ? 1200 (RET)
Cluster Size <0> ? 16 (RET)
Number of Clusters (0-7) <1> ? 3 (RET)
Position <0> ? 10000 (RET)
Account Name? LASTLY (RET)
Account _DR3:[40,40] created at DCNs 18051 23993 23997
```

The SYSTAT information corresponding to the disk on which the account was created shows a pack cluster size of 4:

```
$ SY/D (RET)

Disk Structure:
Dsk Open     Size        Free    Clu  Err  Name     Level  Comments
DL1    0    20460     6765  33%    1    0  V72       0.0   Pri, DLW
DM1    0    53768     5342   9%    1    1  OLDDSK    0.0   Pri, DLW
DR1   49   131648    15220  11%    4    0  SYSDSK    1.1   Pub, DLW
DR2    0   242576    33040  13%    8    0  NEWDSK    1.1   Pri, R-O, DLW
DR3   22   500352    52456  10%    4    0  KNIGHT    1.1   Pri, DLW
DR4   10   242572    12832   5%    4    0  GILLUM    1.1   Pri, DLW
DR5    0   500352    76152  15%    8    0  HISTRY    0.0   Pri, R-O, DLW
```

The directory cluster size is 16, and 16 divided by 4 is 4. Note that the device cluster numbers of the last two DCNs differ by 4, but the first two DCNs differ by a much greater number (23993 minus 18051 equals 5942). Therefore, the directory was not created contiguous.

4.1.4 Deleting Accounts with the DELETE Function

The DELETE function removes individual accounts from the system disk or from private disks. As with the ENTER function, you must logically mount a private disk before you can use REACT to delete any accounts.

Before deleting an account from a disk, you must remove all the files from the UFD with either the PIP or UTILTY program. The /ZE switch of PIP and the ZERO command of UTILTY initialize the UFD directory and thus effectively remove all the files stored in the user account. See the *RSTS/E System User's Guide* for more information on PIP and Chapter 7 for a description of the ZERO command of UTILTY.

You now are ready to run the REACT program. Type the letter D (DELETE) to indicate that you want to remove an account, and respond to the series of questions the DELETE function initiates. For example:

```
$ RUN $REACT(RET)
REACT V8 RSTS V8 TIMESHARING
System Account Manager
Function? D(RET)
Proj, Prog? 100,100(RET)
Disk? DM1:(RET)
Proj,Prog? (CTRL/Z)
```

This example shows how to remove from DM1: account [100,100]. REACT prints the PROJ,PROG and DISK questions and returns to the PROJ,PROG question to let you delete as many accounts as you want. When you are through, type a CTRL/Z to the PROJ,PROG question to exit the program and return to the keyboard monitor prompt.

Table 4–3: Responses to DELETE Function Questions

Question	Response Format and Meaning
PROJ,PROG?	*p,pn* Enter the account number that you want to delete. The project number *p* can be any number from 1 to 254; the programmer number *pn* can be any number from 0 to 254. (CTRL/Z) Type CTRL/Z to the PROJ,PROG question after you finish entering accounts. (You type CTRL/Z only after you answer both the PROJ,PROG and DISK questions. REACT returns to the PROJ,PROG question again. It is then that you can add other accounts or exit the program with CTRL/Z.)
DISK?	*dev:* You must specify the disk device on which the account you are deleting resides. (If you do not, you will get the error message, "?Please Specify Disk".) The device is a combination of a two-character device designator and a unit number; for example, DM1: or SY:.

4.1.5 Automatic Account Creation with the STANDARD Function

The STANDARD function of REACT makes it easier for you to create all user accounts in the Master File Directory (MFD) on the system disk. Generally, you use this function when you are creating a new system disk during system generation. At that time, the system disk does not contain any user accounts. Your previous system disk does, however, contain a file named ACCT.SYS that has account information for the users on your system. You need to copy that file to the new system disk where the STANDARD function can use it to recreate all accounts that existed on the previous system disk. Without this feature, you would have to enter each user account individually – a tedious task.

To use the STANDARD function, run the REACT program, type the letter S (for STANDARD) in response to the FUNCTION question, and wait until REACT notifies you that the accounts have been created. The procedure is:

```
$ RUN $REACT(RET)
REACT V8 RSTS V8 TIMESHARING
System Account Manager
Function? S(RET)

All Accounts in Account File are now Entered
Function? (CTRL/Z)
```

The program prints the FUNCTION prompt again to allow further input. If you have no more accounts to create or delete, type CTRL/Z to return to the keyboard monitor prompt.

The information the REACT program uses to create all accounts on the system disk comes from the account file ACCT.SYS. The RSTS/E distribution kit contains an ACCT.SYS file with some account information already created. The file expands each time you add accounts to the system disk with the REACT program. The same information you type in response to the ENTER function dialogue is added by REACT to the ACCT.SYS file. Thus, ACCT.SYS includes, for each account on the system disk, the following information:

- User account number [nnn,mmm]
- Disk designator and account password
- Disk storage quota
- Account UFD cluster size
- Account name (optional)
- Number of clusters to preallocate
- Cluster to place the UFD

If you look at a listing of ACCT.SYS, you will notice that REACT places the information for each account on a single line, as in the following listing of three user accounts:

```
1 , 2 ,DR1:SWAP0, 0 , 16 ,SYSTEM LIBRARY , 7 , 0
100 , 100 ,DM0:PATCH?, 0 , 16 ,PATCH , 2 , 1000
161 , 51 ,DM0:*, 0 , 0 ,, 1 , 0
```

Thus, when you use the STANDARD function, REACT simply reads the ACCT.SYS file and creates an account for each account represented in the file. Table 4–1 contains the description of the items REACT needs to add an account to a disk device.

You do not need to include an account name when you use the ENTER function to add an account. An account name is, however, an effective way to identify the purpose of the account. It also serves to identify the account from which a user places a message on the system with the GRIPE program. Thus, GRIPE, and not REACT, is the only program on the RSTS/E system that uses the optional name information. So while it is an optional field, you may find it helpful to choose an account name that identifies who owns the account as well as indicates the type of work for which it is to be used. Be sure not to include any commas, single quotation marks, or double quotation marks in the name you choose.

The REACT program updates ACCT.SYS when you create a new account (with ENTER) on the system disk. It does not, however, remove information about accounts that are deleted (with DELETE). This means the ACCT.SYS file on your system may not be up-to-date at all times. You must therefore edit your ACCT.SYS file to make sure it contains the accounts you want created before using the STANDARD function. (You can edit the file using one of the text editors provided with your system.)

4.2 Performing System Accounting Operations — MONEY

The MONEY program prints a set of questions that enable you to extract information about the accounts on your system. You can be as selective as you want, choosing either to extract information about one account or to select information for all accounts on the system. Nonprivileged users can run the MONEY program to get their own account information (except password) but are not able to access information from any other accounts. The *RSTS/E System User's Guide* describes the nonprivileged aspects of the MONEY program, while privileged features are described in the following sections.

4.2.1 Running the MONEY Program

Run the MONEY program by typing the following command at a keyboard monitor prompt:

```
$ RUN $MONEY (RET)
MONEY V8 RSTS V8 TIMESHARING
System Accounting Program
```

MONEY then prints two lines of identification information followed by a sequence of questions. The following is an example of the MONEY program. This sample run shows how to extract information for account [1,210]:

```
$ RUN $MONEY (RET)
MONEY V8 RSTS V8 Timesharing
System Accounting Program
Output report to <_KB:.LST> ? (RET)
Print passwords <NO> ? YES (RET)
Reset <NO> ? YES (RET)
Account <_SY:[*,*]> ? [1,210]

Accounting dump of _SY:[1,210] on 12-Dec-82 at 10:23 AM
with statistics being reset

 Acct    Password    CPU-Time     KCT's Connect Device    Disk  Quota  UFD
  1,210  OHBOY     1:49:50.4    581137  121:38   2:29     3048   5000   16
Account <_SY:[*,*]>? (CTRL/Z)
```

Table 4–4 contains a description of all the questions in the program dialogue.

Table 4–4: MONEY Program Options

Option Question	Reply and Explanation
Output report to <_ KB:.LST>? *dev:filenam.typ*	Enter a file specification if you want MONEY to place the account information in a file. You can specify either a file-structured or a non-file-structured device. The default is your terminal. If you specify a file name but no device or file type, the defaults are SY: and .LST.
Print passwords <NO>?	Type YES if you want MONEY to provide a list of the passwords for the accounts. Type NO or press RETURN to accept the default of no passwords. If you do type YES, and output to a file, the file is written with protection code 128, which means that it is zeroed before it is deleted.
Reset <NO>?	Type YES to reset the accounting information. Selecting reset causes MONEY to reset CPU time, KCT time, connect time, and device time to zero (after producing its report). The report contains the line "with data being reset" to indicate a reset occurred. Type NO or press RETURN to accept the default of no reset.
Account <_SY:[*,*]>?	*dev:[proj,prog]* Type the disk and account that you want information for. The default is all accounts on the public disk structure. If you specify a device but no account, MONEY lists all accounts on that disk. If you type an account but no disk, the public disk structure is assumed. (If you type an account but no disk, you can omit the brackets.)

(continued on next page)

Table 4–4: MONEY Program Options (Cont.)

Option Question	Reply and Explanation
Account <_SY:[*,*]>?	Meaningful data for a nonsystem disk includes account number, number of blocks used, disk quota, and UFD cluster size. Data for disks in the public structure include CPU time, KCT time, connect time, and device time. Note that you can use the wildcard indicator (*) for either the project number, the programmer number, or both. A wildcard means that all data for all project and/or programmer numbers on the device will be given. MONEY repeats this question until you exit the program by typing CTRL/Z.

You can run MONEY during normal timesharing. No conflicts occur if the system attempts to update a user's accounting information while the MONEY program is accessing it. When you want to zero account information by typing YES to the RESET question, the program sets the account information to zero before the system is allowed to update the information again. Thus, no accounting information is lost.

You now know how to get information about the accounts on your system. The following section tells you what this information means.

4.2.2 Output from the MONEY Program

The MONEY program displays information for each account on a single line:

```
 Acct    Password     CPU-Time     KCT'S Connect Device     Disk  Quota  UFD
101,202  GUMMY      1:03:12.9    538370   85:59   1:11      748      0   16
```

The example represents a single entry generated by running MONEY and selecting a listing of all account information on a typical system disk. You see only a partial list here; to gain a more complete understanding of how MONEY works, run the following example on your system:

```
$ RUN $MONEY (RET)
MONEY  V8  RSTS  V8  Timesharing
System Accounting Program
Output report to <_KB:.LST>? (RET)
Print passwords <NO>? YES (RET)
Reset <NO>? (RET)
Account <_SY:[*,*]>? (RET)

Accounting dump of SY:[*,*] on 10-Jan-83 at 11:48 AM

 Acct    Password     CPU-Time     KCT's Connect Device     Disk  Quota   UFD
  0,1     ??????         0.0         0       0      0      6516      0    16
  1,0     ??????         0.0         0       0      0       480      0    16
  1,1     ??????       1:23.8      8882    1:31      0         0      0    16
  1,2     GERAM      7:43:20.8   5538005   35:59  39:49      8984      0    16
  1,50    AARDVK       8:18.6    120342    1:59   2:37      1304      0    16
  1,113   ROBOT1     2:32:06.4   2008382   26:58  19:50         0      0    16
```

(continued on next page)

```
  2,9     ?927??          51.5      8223         0       12       160         0       8
  2,112   ??????           0.0         0         0        0         0         0       4
  2,142   JUNQUE       1:58.4      22795      5:44        1       232         0       4
101,202   GUMMY     1:03:12.9     538370     85:59     1:11       748         0      16
```

The example is only a partial list of the accounts on a typical system disk. The list, as you will see for your system disk, may actually be much longer. When MONEY finishes displaying the accounts, it returns you to your keyboard monitor prompt. Now, refer to Table 4–5 for a description of each of the items in a MONEY account listing.

Table 4–5: MONEY Program Output

Header	Example and Description
ACCT	101,202 Number of the account for which information is kept. It represents the project-programmer number of the account.
PASSWORD	GUMMY The password you created with REACT (or changed with UTILTY). It is the password the user with account 101,202 enters when logging onto the RSTS/E system.
CPU-TIME	1:03:12.9 Number of hours, minutes, seconds, and tenths of seconds (hh:mm:ss:ts) of processor time the the account has used since you last reset either all the accounts on the system or reset this account.
KCTs	538370 Memory usage value that represents the use of 1K word of memory for one tenth of a second of CPU time. The acronym stands for Kilo-Core-Ticks.
CONNECT	85:59 Number of hours and minutes (hh:mm) a user has been logged in to the account, or, more simply stated, connected to the computer.
DEVICE	1:11 Number of hours and minutes (hh:mm) the account has made use of devices other than disk devices.
DISK	748 Number of 256-word blocks of disk storage the account currently uses.
QUOTA	0 Maximum number of 256-word blocks the account is allowed to retain on the specified disk. If the account has more than the allocated number of blocks when the owner logs out, the LOGOUT program asks the owner to delete enough files to fall below the quota limit. Until that happens, the owner is not allowed to log out. The value 0 means an infinite quota. An asterisk appears next to the quota if the account is over quota.
UFD	16 Cluster size of the User File Directory.

A few comments about the values MONEY prints may help you understand how to interpret and thus make better use of this information:

- You can use KCTs and CPU–TIME information to determine billing costs or evaluate account usage. It is important to know, however, that KCTs reflect system usage more accurately than CPU–TIME. For example, two users may exhaust one minute of CPU–TIME in an accounting period. But one user may tie up 2K words of memory while the other may occupy 26K. The KCTs for the first user are incremented by 2 for each tenth of a second the 2K job is running. With the 26K user, each tenth of a second causes the KCTs value to increment by 26; the process of incrementing KCTs is not using up system resources. This extra usage is reflected in the higher KCTs value. Thus, you can calculate a user's average job size by dividing the number of KCTs by the number of tenths of seconds derived from the value of CPU-TIME.
- The DISK value reflects the actual number of blocks that are allocated to the account on disk. It is not necessarily the same value reported by the BASIC–PLUS CATALOG or the DIRECTORY DCL command. A file may occupy one block on a disk as reflected by the CATALOG command but may tie up three additional blocks of disk storage if the file cluster size is four blocks. This means the system allocates four contiguous blocks although the file is currently occupying only one.
- The UFD value represents the cluster size of the User File Directory, which the system manager selects when the account is created by the REACT program. It has no accounting value but rather is provided for your information.
- The system manager should periodically type YES to the RESET question to reset account values. This must be done to prevent account values from overflowing. Table 4–6 shows the maximum amount of time that MONEY can store for each statistic without an overflow.

Table 4–6: Maximum Times Allowed Before Overflow

Time	On Disk	In Memory
DEVICE	1092 hours	1092 hours
CONNECT	1092 hours	68 hours
CPU	116 hours	29 hours
KCT'S	116 hours at 16K words	29 hours at 16K words

At log-out time, the system updates the values on the disk with the accumulated values from memory. Thus, to prevent the loss of accounting information, the user must log off the system before any of the values in memory overflow. In the same manner, you as system manager must reset the account values on disk before they overflow. The sizes of the accounting data fields on disk allow approximately one week of continuous system operation

without overflow. With this in mind, run MONEY to reset these values at least once a week. When it is about to reset account values, MONEY adds to the header line the following text:

```
With data being reset
```

For disks created in Version 8.0, MONEY lists the accounts in ascending order. For a disk created before Version 8.0, MONEY lists the accounts in their order of creation.

4.2.3 Using the Single Line MONEY Command Format

You can create a CCL command to allow a single line format for the MONEY command. (An alternative to creating a CCL is to write a program that puts the command line into core common and then chains to MONEY at line 30000.)

To install MONEY as a CCL command, type:

```
$ RUN $UTILTY (RET)
#CCL MON-EY=[1,2]MONEY.*; 30000 (RET)
#EXIT (RET)
```

You can add these commands to the CCL.CMD startup control file.

The single line MONEY command has the following format:

MONEY [outfile=][account][/qualifiers]

The available qualifiers are:

```
/PASSWORDS
/RESET
/CHAIN
```

For example:

```
MONEY JUN12.LST=[2,*]/RESET (RET)
```

Defaults are the same as in the dialogue. Therefore, if you do not specify an output file, the report is displayed at your terminal. If you do not specify an account, MONEY lists all accounts on the public disk structure. To specify the account, use the same format as in the dialogue.

If you simply type MONEY and press the RETURN key, the result is the same as if you type RUN $MONEY.

The qualifiers for the single line format are described in Table 4–7.

Table 4–7: Qualifiers in the Single Line MONEY Command Format

Qualifier	Description
/PASSWORDS	Causes MONEY to display a list of passwords for the accounts. By default, passwords do not appear.
/RESET	Causes MONEY to reset the accounting information (CPU time, KCT time, connect time, and device time) to zero. The report contains the line "with data being reset" to indicate that a reset occurred.
/CHAIN	Causes MONEY to chain to (or run) a program you specify when MONEY is ready to exit. The format is: `/CH[AIN]:"prog-name[;line-number[;core-common-string]]"` Instead of quotation marks, you can type any character that does not appear in the program name, line number, or core common string. "Line-number" is the line number at which to chain to the program. (For MACRO programs, this is the "parameter word.") The value 31000 is used by convention. If you do not specify a line number, MONEY assumes 0 and chains to the program at line 0. "Core-common-string" is the command string that MONEY will place in core common before chaining to the specified program. MONEY places the string starting at byte 2. If you do not specify a core common string, MONEY places the null string in core common, starting at byte 2. MONEY places a binary status code in byte 1 of core common. The status code is zero if MONEY succeeds. The status code is nonzero if MONEY detected an error while processing the command. Consider the following example of the /CHAIN qualifier. If your installation had a program to convert MONEY's output into billing information for its users, a sample command line would be: `MONEY ACCT,LST=[3,*]/RESET/CHAIN:"[1,2]STATMT;31000;BILLING PROJ=3"` RET In this example, ACCT.LST will contain accounting information for all [3,*] accounts. Accounting information in these accounts will be reset. MONEY then chains to the program named STATMT in account [1,2] at line 31000. MONEY places its exit status followed by the string "BILLING PROJ = 3" in core common before chaining to the STATMT program.

Chapter 5
Operator Services and Spooling

This chapter describes operator services programs and operations involving interaction with operator services. Except for BACKUP which is described in Chapter 8, the operator services programs are completely described in this chapter. Most of the programs require permanent privilege to run. General aspects of related user programs, such as QUE and BATCH, are mentioned only briefly in this chapter but are described fully in the *RSTS/E System User's Guide*.

5.1 Comparison of the Two Available Spooling Packages

RSTS/E supports two spooling packages in Version 8.0. Commands for using the standard package (referred to as the "large" spooler) are described in this chapter. Chapter 11 includes information on the DCL commands used for the micro-RSTS spooling package (referred to as the "small" spooler).

You can install either the large spooler, the small spooler, or both on your system. Differences between the two spoolers are described below. The *RSTS/E System Generation Manual* includes information on installing each of the two spoolers.

The primary differences between the small and large spoolers are:

- The small spooler reduces overall system load, and requires only one job slot on the system. By contrast, the large spooler requires a minimum of three job slots for comparable services.
- The small spooler executes considerably faster than the large spooler, due to improvements in the code for better performance.

- The large spooler provides both batch and printing services, while the small spooler provides only print spooling.
- The small spooler interface is considerably easier to use than the interface for the large spooler, because of improvements in the communication between various parts of the package.
- The large spooler provides greater operator control of the package than the small spooler.
- The small spooler can print files with any RMS format, while the large spooler is restricted to the more standard formats.

The rest of this chapter pertains to the large spooler.

5.2 Overview of Operator Services

Operator services on RSTS/E involve the OPSER program and controlled programs in OPSER tables. Controlled (on-line) programs are:

- QUEMAN
- SPOOL
- BATCH
- BACKUP

SPOOL and BATCH are spooling programs executing queued requests for either a line printer device or batch processor. QUEMAN is the queue manager program that passes queued requests to spooling programs and adds, updates, and deletes requests in the various queues. BACKUP is the system data saving and restoring package that OPSER can optionally control.

5.2.1 OPSER Program Overview

The OPSER program establishes interjob communication on which all on-line programs depend. OPSER declares itself a message receiver with a system-wide identification that the controlled programs can recognize. When a controlled program starts, the program declares itself a message receiver and supplies OPSER with certain data by means of the system message send/receive SYS call. (Refer to the *RSTS/E Programming Manual* for information about SYS calls.) OPSER places the program in its table of on-line jobs.

The identification with which OPSER communicates with on-line programs is called the message receiver identification, or simply, the receiver identification. Each identification exists in the system message receiver table and must be unique. Because of this uniqueness, only one copy of OPSER can be running on your system. Additionally, because all control and interjob communication of spooling jobs depends on OPSER, the operator services program must be running before any controlled jobs can be run.

After it establishes initial interjob communication, OPSER makes it possible for an operator to interact with the controlled jobs. It becomes the main interface between the operator and a system controlled program. Figure 5–1 illustrates the interaction among system controlled programs and the operator.

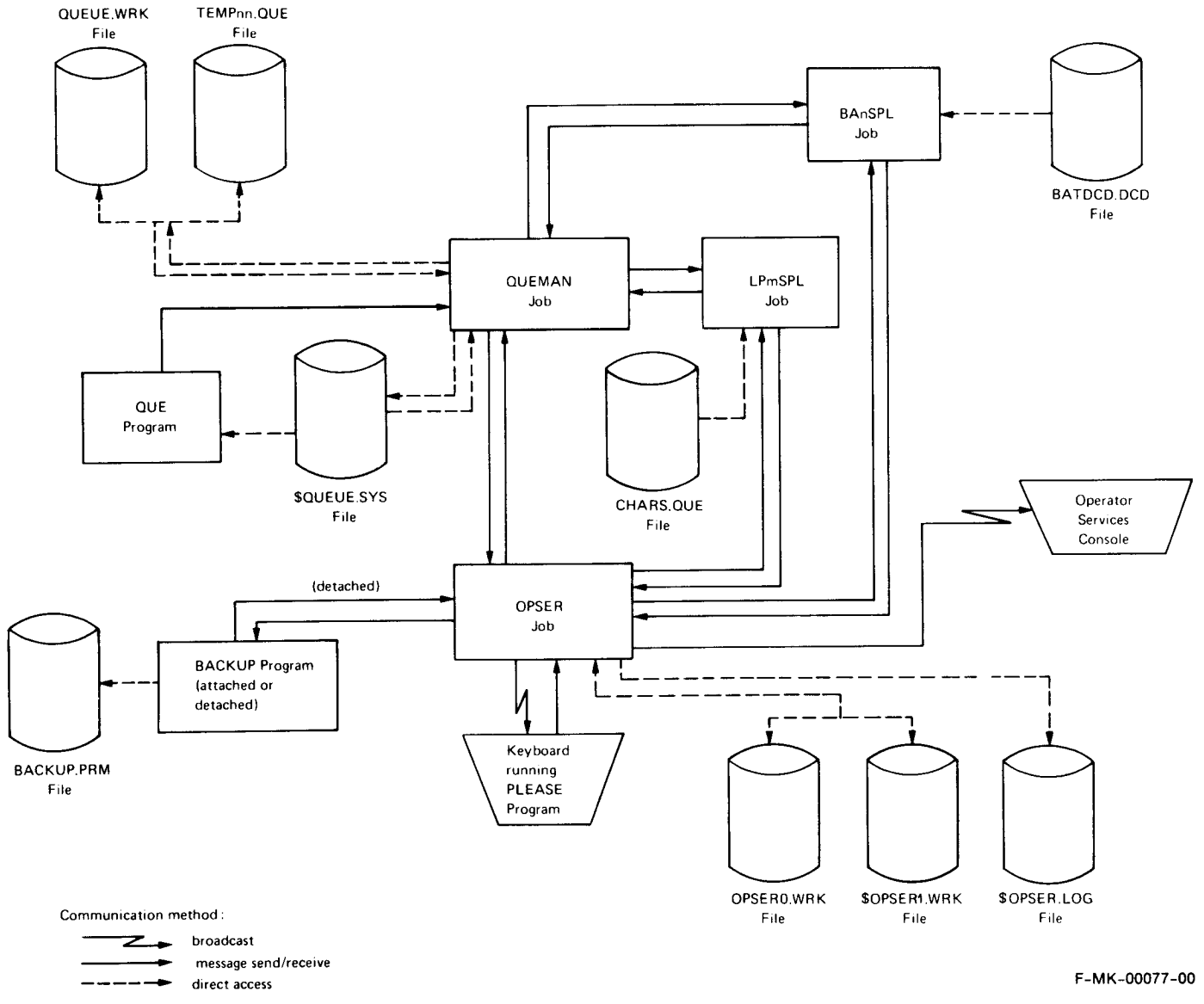
OPSER broadcasts information to the operator through a terminal designated as the Operator Services Console (OSC). For flexibility, the operator can make any keyboard line on the system the OSC. To isolate the keyboard control from unwarranted tampering, OPSER itself need not be connected to the OSC but may broadcast data on the physical keyboard line. Given that intermittent output is generated on the OSC, any other user, including the operator, could be logged into the system on that terminal.

OPSER itself establishes a data base by which it controls on-line jobs. The data base resides in the system library account [1,2] and consists of the work files OPSER0.WRK and OPSER1.WRK and the log file OPSER.LOG. OPSER0.WRK contains the current activities and message control directory tables. OPSER1.WRK has the table of jobs on line to OPSER, the valid operator table, and tables of messages and legal commands. The file OPSER.LOG, which can be optionally renamed, provides a history of operating activity.

To provide a nonvolatile data base during a time-sharing session and to provide continuity of operations between time-sharing sessions, OPSER stores the work file information on disk. Thus, if the system crashes or if OPSER alone unexpectedly terminates, the work files retain the most current processing information. Similarly, these files maintain data continuity from one time-sharing session to another. Whenever OPSER starts, it examines its files to determine whether entries in the on-line job and valid operator tables are still valid. Generally, OPSER retains those active and valid entries and purges those entries that do not satisfy validation requirements. Upon restarting, therefore, OPSER attempts to preserve data from its previous operational state.

The operator most conveniently interacts with OPSER through the PLEASE program. PLEASE transmits commands to the OPSER program. OPSER checks that the sender is a valid operator and, if so, takes the requested action. Because of PLEASE, the operator need never directly run or attach to OPSER to perform operator functions on controlled jobs. Section 5.7 contains a description of the PLEASE program.

Figure 5–1: System Controlled Programs and Operator Interaction



OPSER provides processing checks for on-line jobs. OPSER ensures that all on-line jobs are still active, are still valid receivers, and are not hibernating. If any job is found in the hibernate state, OPSER notifies the operator, who can attach the job to a terminal, remedy the cause of the hibernation, and/or restart the job.

OPSER interaction between a controlled job and the operator is recorded in one of three ways:

- As a message. A message is specifically formatted data to which OPSER assigns a sequence number for operator reference.
- As an action request. An action request is a special message that requires operator action and a response before a controlled job can resume processing. A request usually involves both performing an action (such as mounting a volume) and typing an answer (such as a device designator or program interrupt command) in direct response to the request.
- As an information line. An information line is a free form report of some internal operation performed. Information lines are typically not important to the operator function but merely provide a history of on-line events.

5.2.2 QUEMAN Program Overview

The QUEMAN program manages the system queue file QUEUE.SYS in the system library account [1,2]. The file retains all pending user requests and can store requests between time-sharing sessions. Users make requests of spooling programs through the QUE system program. To form a request for a user, QUE builds a message and sends it to QUEMAN, which updates the QUEUE.SYS file. QUEMAN maintains a table of on-line spooling programs in a work file, QUEUE.WRK, which is in the account that contains the Spooling Package Library. (Refer to Section 5.2.7.) The program passes a queued request to a destination spooling job not currently busy with a request. As a spooling job completes a request, it notifies QUEMAN, which updates the queue and work files and checks for another request to pass to the spooling job.

A spooling program typically handles one request at a time from QUEMAN. If no request is pending for a spooling program, the job enters an indefinite sleep state. The job is not awakened until the system queues a message for it. The message is the result of QUEMAN or OPSER transmitting data to the spooling job by means of the system message send/receive mechanism.

To retain as much data as possible between time-sharing sessions, QUEMAN, like OPSER, examines its files upon restarting. After restarting, QUEMAN should resume processing with no loss of data from its previous run. QUEMAN examines the queue file structure and contents and reestablishes synchronization with any spooling jobs already on-line and active.

5.2.3 SPOOL Program Overview

The SPOOL program handles requests made for line printer output. SPOOL maintains dual communication paths: one with QUEMAN and one with OPSER. QUEMAN transmits queued requests to SPOOL. SPOOL interacts with QUEMAN only to notify it that a transmitted request is completed. Under normal circumstances, communication between SPOOL and OPSER is necessary only when an operator requires some special action (such as aligning forms) of the spooling job.

To identify output of a user's queued request, SPOOL prints heading burst pages. The program accesses the character generation file CHARS.QUE in the Spooling Package Library to form the large, easily readable letters necessary to distinguish discrete requests and files within a request. A program generates the CHARS.QUE file during the system library build procedure of system generation. You can rebuild the CHARS.QUE file following the procedures described in Appendix A.

5.2.4 BATCH Program Overview

The BATCH program executes user requests by running a job for a user on a pseudo keyboard. By using a pseudo keyboard, BATCH eliminates the requirement for a physical terminal and relieves the user from typing by processing commands from a disk file.

BATCH, like SPOOL, maintains dual communication paths: one with the QUEMAN program and the other with OPSER. QUEMAN sends a message to BATCH to process a job request. BATCH interacts with QUEMAN only to notify it that a transmitted request is completed. BATCH typically depends on OPSER to form requests for operator action. To mount a user volume, for example, BATCH notifies OPSER, which, in turn, generates the request for the operator and processes the response from the operator.

To process commands from a user-queued disk file, BATCH requires the command decoding file BATCH.DCD in the Spooling Package Library. A program generates this file during the system library build procedure of system generation. You can rebuild it by following the procedures described in Appendix A. The BATCH.DCD file contains all the codes and parameters BATCH needs to decode user-specified control statements, perform syntax checking, and form a special intermediate file (BAnJmm.COM) of encoded BATCH commands.

Both the SPOOL and BATCH programs charge processing (execution) time to the account under which the user queues the request. Overhead time, used to set up a request, is charged to the account under which the program is running.

5.2.5 Controlling BACKUP with OPSER

An operator can control the BACKUP program through OPSER. If BACKUP runs detached, control through OPSER is the only means by which the operator can communicate with a BACKUP job. BACKUP, when detached, transmits all operator requests through OPSER and receives responses only from OPSER.

5.2.6 Overview of OPSER Shutdown

The entire operator services package is designed to terminate in a controlled manner using what is called shutdown level. The defined shutdown level for each job determines the order in which OPSER terminates controlled jobs during system shutdown. The SHUTUP system program can be run so that OPSER terminates controlled jobs with the lowest shutdown level first. OPSER informs each job at a certain level to complete processing at a logical end point. Only after OPSER shuts down all jobs at a given level does it proceed to shut down all jobs at the next highest level. SHUTUP itself does not proceed with final system shutdown until OPSER terminates all its controlled jobs and kills itself. By using the controlled shutdown of OPSER, the system manager can ensure continuity of processing user requests from one time-sharing session to another.

5.2.7 A Note on Running OPSER and the Controlled Programs

OPSER and the controlled programs QUEMAN, SPOOL, and BATCH must reside in the Spooling Package Library (SPL) in executable form. If you try to run one of these programs from the SPL account and the program is not in executable form, OPSER prints:

```
?<program> must be compiled
```

The message tells you which program is not executable. Generally, programs that are in executable form have a .BAC or .TSK file type. After the program prints the message, you must load a compiled version of the program into the Spooling Package Library account and try again. This message can also appear if you run the program successfully, terminate the program with a CTRL/C, and then try to run the program from memory by typing only RUN.

You must include the account number of the Spooling Package Library in the RUN command. For example, RUN $OPSER and RUN [1,10] OPSER are valid commands but RUN is not. These restrictions are necessary to allow you the option of placing the Spooling Package in any account on the system instead of requiring you to run these programs from the system library account [1,2] only.

5.3 Operator Services Program — OPSER

The operator services program OPSER consists of two modules: OPSER and OPSRUN. Each module is compiled and stored in the Spooling Package Library, has a protection code of <124>, and requires permanent privilege to run. The first module establishes the initial conditions on start-up, checks initial conditions upon restart, and chains to the second module. The second module actually executes commands and periodically checks status. For simplicity, the documentation refers only to one program, OPSER, which functionally includes the two modules.

You must run the OPSER program before the queue manager and spooling programs. To start OPSER, type the following command at a terminal logged into the system under a privileged account:

```
RUN $OPSER
OPSER V8  RSTS V8  TIMESHARING
#
```

At this point, the OPSER program performs a number of checks before it prints the pound sign (#) prompt. The remaining portion of this section describes these initial program checks. Once you are familiar with this procedure, you can then learn in Section 5.3.1 about OPSER operator commands.

If you are not using a privileged account, the system displays an error message to indicate that you do not have access to OPSER.

If the account is privileged, the program prints an identification line at the terminal to indicate it is running. (Refer to the *BASIC–PLUS Language Manual* for a description of the ?PROGRAM LOST-SORRY message.)

OPSER tests for the presence of its two work files (OPSER0.WRK and OPSER1.WRK) in account [1,2]. If the work files are not present, OPSER prints:

```
'OPSER' FILES NOT FOUND - WILL INITIALIZE
```

OPSER then initializes all its tables. At this point, the valid operator table has an entry that allows all privileged users from any terminal to communicate through OPSER. If ERRCPY is active, the on-line job table has ERRCPY as its only entry.

When OPSER finds the work files, it ensures that it has write access to the files. Should another job have write access to the files, OPSER prints the following message and terminates:

```
'OPSER' DOES NOT HAVE WRITE PRIVILEGES TO ITS FILES
```

To allow OPSER to gain write access, the operator must terminate the job currently having write access and run OPSER again.

The program performs further special processing for currently existing work files. The processing ensures data integrity in case OPSER is being restarted after a system crash. A check is made of the entries in the on-line job and valid operator tables.

For each entry in the on-line job table, OPSER checks the following conditions:

1. The job number indicated in the entry must be active on the system.
2. The receiver identification in the entry must exist in the system message receiver table for that job number.
3. The account in the entry must be a valid account number.

An entry for an inactive or otherwise illegitimate job is removed from the on-line job table. For each active and legitimate job, OPSER requests, when processing begins, retransmission of the job's last message. OPSER generates a command to display (when processing begins) the on-line job table.

OPSER checks the entries in the valid operator table. What further actions take place depend on whether jobs found in the on-line job table are active. If the on-line job table has no active jobs and a meaningless entry is in the valid operator table, OPSER clears the table and sets initial conditions. The resulting operator table has one entry that allows, as valid operators, all privileged users from any keyboard. If the on-line job table had an active job, the program merely removes from the valid operator table those entries that are not meaningful. To signal that a change in the current table has occurred, an internal command is created to print the valid operator list when processing begins.

After processing of the work files is complete, the program attempts to declare OPSER as a receiving job. If the declaration fails, the program prints:

```
'OPSER' CANNOT DECLARE ITSELF A RECEIVER
```

This message indicates that another copy of OPSER has been run and must be properly terminated to allow the current copy to run successfully.

OPSER establishes the current keyboard as the Operator Services Console (OSC) and a file named OPSER.LOG as the default log file. OPSER.LOG is optional, but OPSER uses it until an explicit command closes it or assigns another log file. When OPSER begins normal operation, it executes any internally generated commands to list on-line jobs and valid operators. After executing the commands, OPSER is ready to accept data sent to it by any program. OPSER prints the pound sign (#) character to show that it is ready to accept commands.

5.3.1 OPSER Operator Commands

You can abbreviate a valid OPSER command to three or more characters and separate the command from any operands by a space character. No embedded spaces are allowed within a command. Delimiters within the text of the operands are the semicolon (;) and colon (:) characters and must be present to delimit elements of text. Table 5–1 summarizes the commands and formats. Throughout the following discussion, note that square brackets enclose optional items.

Table 5–1: OPSER Commands

Command Name	Syntax and Meaning
ANSWER	ANS[WER]^msgnumber:text Conveys the text following the colon as a response to an action request denoted by message number. Also deletes the action request.
CHANGECONSOLE	CHA[NGECONSOLE]^KBn: Changes the operator services console to the keyboard unit designated by n. The unit should be on line to the system but need not be free. It must also not be gagged (refer to Section 7.4).
DELETE	DEL[ETE]^msgnumber Deletes from an OPSER internal table an unanswered action request denoted by its message number. DEL[ETE]^#m[:n] Deletes the n oldest (from 1 to 32) action requests for a given job. The command requires the pound sign (#), the number (m) of the sending job, a colon, and the number (n) of requests to delete.
DETACH	DET[ACH] Detaches the OPSER program from its keyboard, after which the operator can run PLEASE to communicate with OPSER.
EXIT	EXI[T] Closes the log file and work files, removes OPSER from the system message receiver table, and terminates the program. If the job is attached, the program returns you to your keyboard monitor prompt. If detached, the program broadcasts a message to the system console terminal (KB0:) and kills itself.
INTERRUPT	INT[ERRUPT]^rcvrid:text INT[ERRUPT]^#n:text Sends the unsolicited text to an on-line job specified either by its message receiver identification or by #n (where n is the number of the job under which the on-line program is running). The LIS JO command listing gives job numbers and receiver identifications.
LIST	LIS[T] JO[BS] LIS[T] OP[ERATORS] Prints a listing of all jobs on line to OPSER or a listing of valid operator accounts and keyboards.

(continued on next page)

Table 5–1: OPSER Commands (Cont.)

Command Name	Syntax and Meaning
LOGFILE	LOG[FILE]∧file;AL[L] RE[QUESTS] NO[NE] Establishes, as the OPSER log file, the specified file or device and sets the message level to the value specified. Message levels are ALL to record all messages and action requests, REQUESTS to record only action requests, and NONE to record no messages, action requests, or OPSER information lines. LOG[FILE]∧;msglevel If a file or device is not specified, a semicolon followed by a level simply changes the message level of the log file currently open. LOG[FILE] If neither a file nor message level is specified, OPSER closes the current log file and records nothing thereafter.
MESSAGE	MES[SAGE]∧level Sets the level of messages for the Operator Services Console. The level controls what types of messages are printed at the OSC. If no level is included in the command, an error is generated. MES[SAGE]∧AL[L] ALL includes all messages, action requests, and OPSER information lines. MES[SAGE]∧RE[QUESTS] REQUESTS includes only action requests and information lines and excludes messages. MES[SAGE]∧NO[NE] NONE means that nothing is printed on the Operator Services Console (OSC). (See Section 5.3.2 for a discussion of message level.)
OPERATOR	OPE[RATOR]∧KBn:[p,pn] For use by privileged operators only. Updates the valid operator list with the designated keyboard unit and account combination. The asterisk (*) character can replace the keyboard unit, project number(p), and programmer number (pn) to include all of that element. OPE[RATOR]∧-KBn:[p,pn] If minus (–) sign precedes the specification, that specific element is deleted from the list.
RETYPE	RET[YPE]∧msgnumber Reprints, at the requesting terminal, an unanswered action request denoted by its associated message number. RET[YPE]∧[#n[:n]] Several unanswered action requests can be printed by giving the sending job's job number. If :n is given with the job number, the n oldest (from 1 to 32) action requests are printed for that job.
The symbol ∧ marks the location of a required space.	

NOTE

Placing a nonprivileged account in the OPSER operator table with the OPERATOR command allows anyone logged in to the nonprivileged account access to all privileged accounts. You must therefore be careful which accounts you place in the OPSER operator table.

5.3.2 Message Types

OPSER displays two types of information on the Operator Services Console (OSC): messages and action requests. Messages are simply informative. For example, a message might tell you that a control file has been queued for batch processing. (Requests require an operator to perform some action before the job continues, such as mounting a tape.)

The MESSAGE command lets you define the level of information displayed at the OSC:

1. MESSAGE ALL – you want both messages and action requests displayed at the OSC.
2. MESSAGE REQUESTS – you want only action requests displayed at the OSC. Messages are held in a table until you specify MESSAGE ALL.
3. MESSAGE NONE – you don't want any displays at the OSC. Messages and requests are held in a table until you specify MESSAGE ALL or MESSAGE REQUESTS and then the RETYPE command.

When OPSER starts, the default is MESSAGE ALL. If you suspend the display of messages, or messages and requests, by typing MESSAGE REQUESTS or MESSAGE NONE, the suspended items are held in a table. The table can hold a maximum of 32 messages and/or requests. (The actual number held depends on the length of the messages and requests.) If OPSER receives a message and/or request after the table is full, the new message or request is added to the table, and the oldest message or request is deleted.

Thus, you should not suspend message printing for long, or you may lose messages and requests. When you want to see messages and requests again, type MESSAGE ALL. Then use RETYPE to print the messages and requests currently held in the table.

If you see "Message Table Full" displayed at the OSC, it means that a message was received, and the table was full. Some unknown number of old messages and requests have been lost while MESSAGE NONE or MESSAGE REQUESTS was in effect. (You probably forgot to use RETYPE; this clears the table and prevents the "Message Table Full" error from occurring.)

To help you distinguish between messages and action requests on the Operator Services Console, output appears in distinct formats. Message requests have the form:

```
MESSAGE tab   tab nnnnn : date time JOB:NN KB:MM Jobname [P,PN]
        tab   text
```

If the first character of the message text is a question mark (?) character, the words FATAL MESSAGE replace MESSAGE in the identification line.

Action requests have the form:

```
REQUEST tab   tab nnnnn : date time JOB:NN Jobname [p,pn] rcvrid
        tab   tab text
```

In an action request, the first character of text is a CTRL/G (BEL) that sounds a bell in the terminal to alert the operator. Table 5–2 summarizes the contents of messages and action requests.

An action request is the result of a spooling job's requesting interaction with the operator. The spooling job sends a message to OPSER, which forms an action request. The spooling job performs no further processing until operator action is taken or until the condition that generated the action request is satisfied. OPSER broadcasts action requests on the OSC unless the message level is set to NONE.

To respond to an action request, the operator uses the ANSWER command. Because OPSER usually runs detached, an operator can run PLEASE at valid operator terminals to send the response to OPSER. The ANSWER command requires a message number to identify the action request to which the operator is responding.

OPSER sends the text in the ANSWER command to the spooling program and deletes the action request. If the response satisfies the request, the program resumes processing. If the response does not satisfy the request, the spooling program still expects an appropriate response. The operator must then send such a response by means of the INTERRUPT command described in Section 5.3.4.

Table 5–2: OPSER Message and Action Request Contents

Item	Meaning
nnnnn	Sequence number of message starting at 1 and incremented by 1 during processing; it must be used in the ANSWER command to respond to a specific action request. After restarting with currently existing work files, OPSER rounds the last assigned sequence number up to the next multiple of 10. This number is the next assigned sequence number.
date	Current system date.
time	Current system time of day.
JOB:nn	Job number under which sending program is running; use it, preceded by the pound sign (#), to refer to the job in an INTERRUPT, DELETE, or RETYPE command.
KB:mm	The keyboard number of the job that sent the message. The KB:mm field is displayed as "DET" if the job was detached and as KB:?? if OPSER is unable to determine the keyboard number. Typically, the KB:?? message appears when the job has terminated.

(continued on next page)

Table 5–2: OPSER Message and Action Request Contents (Cont.)

Item	Meaning
jobname	Name of the program that an on-line sending job is running; this name is printed for information purposes. OPSER prints six question marks, ??????, in place of the jobname if it is unable to determine the jobname. Typically, the ?????? message appears when the job has terminated.
[p,pn]	Project-programmer number under which the sending on-line job is running.
rcvrid	For action requests only. The message receiver identification given to the job when it is started. The identification can be standard (LPnSPL for the SPOOL program or BAnSPL for the BATCH program) or can be optional (as specified in the NAME start-up switch for SPOOL or BATCH). Use the receiver identification in the INTERRUPT command to refer to the spooling program.
tab	Horizontal tab character(s) to indent text. A single HT character denotes message text; two HT characters denote action request text.

5.3.3 Valid Operator and On-Line Job Lists

The LIST command displays data concerning the valid operator list or the on-line job table maintained by OPSER:

1. LIST OPERATORS displays the keyboard number and project-programmer number combinations currently defined as valid operators. The OPERATOR command updates the valid operator list. However, only a privileged operator can update the list.
2. LIST JOBS prints the header ON-LINE JOB and then prints data concerning on-line jobs in the following format:

   #n [p,pn] 'rcvrid' SL = n

Table 5–3 describes the items in the on-line job list.

Table 5–3: OPSER On-Line Job List

Item	Meaning
#n	Number of job under which the on-line program is running.
[p,pn]	Project-programmer number under which the on-line job is running.
'rcvrid'	Message receiver logical identification used to identify the on-line program in INTERRUPT commands.
SL = n	OPSER shutdown level used when SHUTUP notifies OPSER to terminate spooling operations in an orderly fashion.

The job data listed is valuable to the operator. For example, certain OPSER commands require either a receiver identification or job number to communicate with a spooling program.

5.3.4 Operator INTERRUPT Command

The INTERRUPT command sends unsolicited messages and special commands to spooling programs. OPSER accepts any text in the INTERRUPT command and sends it to the destination job. The destination job alone interprets the text. If a response to an INTERRUPT command is generated, it is displayed on the Operator Services Console. Sections 5.3, 5.4, and 5.5 describe special text interpreted by spooling programs as commands.

The INTERRUPT command allows an operator to control and to monitor spooling programs. Each spooling program recognizes commands that can be divided into three sets:

1. Commands that have no direct effect on how the program itself processes a request queued by a user. Included in this set of commands are PAUSE, CONTINUE, NOTICE, LAST, and STATUS. They allow the operator to control the program in general and to gain information about the program.
2. Commands that do have a direct effect on program operation but are uniform for all spooling programs. END, ABORT, and OFFLINE represent this type of command. They directly influence the job.
3. Commands that are unique to the individual spooling program, such as those for forms control on a line printer.

5.3.5 OPSER Start-Up Procedure

Use commands in the system start-up control file to start the OPSER program. (Refer to Section 3.1 for information about creating a start-up control file.) This section describes the commands you should include in that start-up file.

The following control file causes INIT to start OPSER:

```
OPE KB*:(2,2)
LOG $OPSER.LOG;ALL
MES ALL
DET
```

The OPERATOR, LOGFILE, MESSAGE, and DETACH commands establish initial operating conditions for OPSER. The LOGFILE and MESSAGE commands set conditions normally established as defaults and are shown for clarity. An explanation of these commands follows:

- The OPE command updates the valid operator list with a nonprivileged account on any keyboard. It is important to define the operator with a nonprivileged account. This limits the operator to those privileges allowed by OPSER. Allowing the operator to communicate from any terminal on the system gives flexibility to those who are performing operator functions.
- The LOG command keeps the file $OPSER.LOG with a message level of ALL. As a result, all messages, action requests, OPSER information lines, and operator responses are written to the file to provide a complete historical reference.

- The MES command specifies that all messages, action requests, and OPSER information lines are printed at the OSC. Normally, the operator is concerned only with action requests. If the operator is concerned with any unanswered action request, the RETYPE command can be issued. In addition, the log file can be closed with the LOG command and printed at any time to recover recorded information.

- The DET command causes OPSER to detach itself from the terminal on which it is running. At the start of timesharing, this terminal is usually the system console terminal (KB0:). While OPSER runs detached, it is immune from tampering by unauthorized users. The operator can communicate with OPSER through the PLEASE program.

The terminal on which OPSER starts is automatically defined as the Operator Services Console (OSC). You can change this definition by placing the CHANGECONSOLE command in the start-up procedure. Any valid operator may also alter the assignment of the OSC. It is most convenient to keep keyboard unit 0 as the OSC because, regardless of the number of logins currently allowed on the system, the operator can always use the system console terminal to communicate with OPSER.

OPSER changes the OSC to the console terminal KB0: if it is:

- Disabled
- Controlled by a modem
- Gagged (Refer to Section 7.4)

5.3.6 OPSER Action Under Various Start-Up Conditions

OPSER start-up actions depend on the way OPSER last terminated:

- When OPSER runs after you have shut down operator services in an orderly fashion through SHUTUP, a normal start-up procedure occurs. When shutting down in an orderly fashion, OPSER removes all entries from its on-line job table and closes all files properly. On restarting, OPSER finds all data valid. The on-line job table is empty, and entries in the valid operator table are legal and therefore are retained.

- When OPSER starts after a system crash, it may find entries in its on-line job table. Because OPSER starts before other on-line jobs, none of the jobs in the table are active. The program therefore clears the entries from the table. If any entries in the valid operator table are illegal, OPSER clears that table and sets one valid operator entry – all privileged users from any keyboard.

- When OPSER starts after it has terminated unexpectedly, it may find entries in its on-line job table for jobs still active on the system. The program, in this instance, sends a command to each active job having an entry in the on-line job table. The command requests the job to retransmit the last message it sent to OPSER. OPSER does not wait for a response. If any job in the on-line job table does not meet OPSER's

integrity checks, OPSER removes its entry from the table (takes the job off line) and generates an internal LIST JOBS command. OPSER also scans the entries in the valid operator table. If any entry is illegal, the program removes it. If OPSER removes any entry, it generates an internal LIST OPERATORS command.

5.4 Queue Manager Program — QUEMAN

The queue manager program QUEMAN maintains the file of queued requests (QUEUE.SYS in the Spooling Package Library account, [1,2] by default) and communicates with spooling programs to execute queued requests. Queue management consists of two modules: QUEMAN and QUMRUN. Each module is compiled and stored in the Spooling Package Library, has a protection code of <124>, and requires permanent privilege to run. The first module sets initial conditions on starting and checks initial conditions after restarting. To begin processing, the first module chains to the second one, which actually executes commands and manages the queues. For simplicity, the documentation refers only to one program, QUEMAN, which functionally includes the two modules.

The QUEMAN program runs only if the job has permanent privilege and the OPSER program is running (that is, only if the name OPSER is in the system table of message receivers). Starting QUEMAN is a prerequisite to starting the spooling programs. To start QUEMAN, type the following command while logged into the system under a privileged account:

```
RUN $QUEMAN
QUEMAN V8  RSTS V8  TIMESHARING
STARTED AT: 01:52 PM ON 03-OCT-82
```

If you are not using a privileged account, the system displays an error message to indicate that you do not have access to QUEMAN.

If the job has permanent privilege, the program prints its identification line and start-up lines and begins preliminary error checking.

QUEMAN tests for the presence of OPSER. If OPSER is not running, QUEMAN prints an error message and terminates:

```
QUEMAN CANNOT RUN WITHOUT 'OPSER' ACTIVE
```

The operator services program OPSER must be started before QUEMAN can start.

QUEMAN opens the files QUEUE.WRK, OPSER1.WRK, and QUEUE.SYS in the system library account. If QUEUE.WRK does not exist, the program prints the following message and creates the file:

```
'QUEUE.WRK' NOT FOUND - WILL INITIALIZE
```

If OPSER1.WRK does not exist, the program prints the following message and terminates:

```
$OPSER1.WRK NOT FOUND --- CAN'T RUN
```

To recover, the operator must start the OPSER program before starting QUEMAN. If QUEUE.SYS does not exist, the program creates and initializes it. QUEMAN signals this action with the message:

```
NO QUEUE FILE FOUND -- WILL INITIALIZE
```

If QUEUE.SYS exists, QUEMAN ensures that it has write access to the file. If another program has write access to the QUEUE.SYS file, QUEMAN generates the following message:

```
QUEUE FILE OPENED BY ANOTHER PROGRAM
TRY AGAIN (Y/N) <N>?
```

The operator must determine which job has the file QUEUE.SYS open and must terminate that job. If you type Y, QUEMAN retries the open operation. Typing N terminates QUEMAN.

When the open operation successfully gains write access to QUEUE.SYS, QUEMAN declares itself a message receiver on the system. If QUEMAN is already declared by a different job, the program prints the message:

```
QUEMAN CANNOT DECLARE ITSELF AS A RECEIVER ... CAN'T RUN
```

The operator must terminate the other job so that it is removed from the message receiver table. Typing the CONT command causes QUEMAN to try again to declare itself a receiver.

To ensure that the previous QUEMAN job properly closed the currently existing QUEUE.SYS file, the program checks a flag value in the file. If the flag is not properly set, the program prints the following message:

```
QUEUE FILE NOT CLOSED PROPERLY --  NOW CHECKING DATA FOR CONSISTENCY
```

A later message signals that all integrity checks were successful:

```
QUEUE FILE DATA CHECKED FOR CONSISTENCY
```

If the flag is properly set, QUEMAN performs consistency checking but does not print any messages unless it finds problems. If any problems are found, QUEMAN generates the following messages:

```
QUEUE FILE DATA INCONSISTENT - WILL INITIALIZE
INITIALIZED
```

The INITIALIZED message indicates that QUEMAN has set all entries in the QUEUE.SYS file to their initial conditions and has removed all queued requests from the queues. Section 5.4.5 describes the queue file consistency checks QUEMAN performs.

After completing all initial checks, QUEMAN prints the pound sign (#) character indicating its readiness to accept a start-up command or switch.

5.4.1 QUEMAN Start-Up Commands and Switches

QUEMAN recognizes the start-up commands shown in Table 5–4.

Table 5–4: QUEMAN Start-Up Commands

Command	Syntax and Meaning
DETACH	DET[ACH] Detaches QUEMAN from the terminal. You should run QUEMAN detached.
INITIALIZE	INI[TIALIZE] Sets all the entries in the QUEUE.SYS file to their initial conditions and sets initial conditions for the spooling queues. Any currently queued requests are lost.

Before QUEMAN executes the INITIALIZE command, it checks its on-line spooler table. If the table has an entry for a spooling job, QUEMAN prints the following message:

```
SPOOLERS ON-LINE -- CAN'T INIT
```

The program prints the prompt again.

QUEMAN also recognizes start-up switches. One or more switches may appear following a command or may be alone on a line in response to the prompt. Table 5–5 lists the start-up switches QUEMAN allows.

Table 5–5: QUEMAN Start-Up Switches

Option	Syntax and Meaning
PRIORITY	/PRI[ORITY]:nnn Sets the job priority to nnn, where nnn can be from –120 to +120. Without the switch, the program automatically sets the priority to 0.
RUNBURST	/RUN[BURST]:nnn Sets the job run burst to nnn, where nnn can be from 1 to 127. Without the switch, the run burst value currently assigned is used.

If any data entered in response to the pound sign prompt are not valid commands or options, QUEMAN prints the following message:

```
INVALID RESPONSE -- text
```

The text is the data entered. The program prints the pound sign prompt again.

To detach QUEMAN, the operator types the DET command in response to the prompt. For example:

```
#DET
DETACHING...
```

The program then prints the DETACHING message and detaches itself from the keyboard. (You should always run QUEMAN detached.) While detached, the job receives commands from the operator through the operator services program as described in Section 5.4.2. At no time should the operator type CTRL/C to the QUEMAN program. This may corrupt the QUEUE.SYS file.

5.4.2 QUEMAN Interrupt Commands

QUEMAN recognizes interrupt commands that allow an operator, through the INTERRUPT command of OPSER, to get information on queue manager processing and to control program operation. Table 5-6 summarizes these interrupt commands.

Table 5–6: QUEMAN Interrupt Commands

Command	Option and Meaning
END	Terminates QUEMAN in an orderly manner and disables queuing and spooling. If any spooling programs are on line to QUEMAN, END sends to OPSER the notification message SPOOLERS STILL ONLINE – WILL CLEAR TABLE and clears the on-line spooler table. END then sends a command to OPSER that takes the QUEMAN entry out of OPSER tables. QUEMAN closes its work files and kills itself.
LAS[T]	Sends to OPSER the most recent message generated by QUEMAN.
NEX[T]	*quenam = reqnam* Places the pending user request identified by *reqnam* at the head of the queue specified by *quenam*. A request name can contain a job name, project-programmer number, and sequence number. Request names are found in the listing generated by the QUE program command L dev:, where dev: is the appropriate queue name. For example, LP:, BA:, or RJ: are valid queue names.
OFF[LINE]	Terminates QUEMAN as the END command does but does not generate a notification message.
STA[TUS]	Prints a brief report of spooling jobs on-line to QUEMAN.
DIS[ABLE]	QUE[UING]<*text*> SPO[OLING] ALL Disables QUEUING or SPOOLING or both operations with the ALL option. When you specify DISABLE QUEUING, the program QUE stops sending messages to QUEMAN and stops listing the queue. Any jobs already sent to QUEMAN are processed. If you specify DISABLE SPOOLING, QUEMAN prevents any further jobs from being sent to any spooler. Jobs sent to the spooler prior to the DISABLE command are processed to completion. Specifying DIS[ABLE] ALL tells QUEMAN to terminate both QUEUING and SPOOLING. You can include a message to the user while specifying the DISABLE command. Type the command, the option, and then a space followed by the message. After QUEMAN processes the command, it prints the <text> message in one of the following formats: 1. Further QUEUING DISABLED <text> 2. Further SPOOLING DISABLED <text> 3. Further QUEUING AND SPOOLING DISABLED <text> If you ignore the <text> field, QUEMAN prints the default text, "by operator". The message appears when you try to queue a file.

(continued on next page)

Table 5–6: QUEMAN Interrupt Commands (Cont.)

Command	Option and Meaning
ENA[BLE]	QUE[UING]<text> SPO[OLING] ALL Enables QUEUING or SPOOLING or both operations with the ALL option. When you specify ENABLE QUEUING, the program QUEUE begins to send messages to QUEMAN and starts listing the queue. If you specify ENABLE SPOOLING, QUEMAN begins to send jobs to the spoolers. Specifying ENABLE ALL tells QUEMAN to start the QUEUING and SPOOLING functions. Use the <text> field to send messages to the user. Type the command, the option, and then a space followed by the text of the message. After QUEMAN processes the command, it prints the message in the following formats: 1. QUEUING ENABLED – <text> 2. SPOOLING ENABLED – <text> 3. QUEUING AND SPOOLING ENABLED – <text> The default <text> message is "by operator". The message appears when you try to queue a file.

When a command is received from the operator through OPSER, QUEMAN performs the action requested and either generates a message or an information line. OPSER formats the message or information line and displays it on the OSC. An example of this interaction between an operator, QUEMAN and the OPSER program is shown using the STATUS command. The following example assumes the operator is running the PLEASE program and the OSC is the keyboard on which PLEASE is running:

```
#/INT #6:STATUS
COMMAND SENT TO 'OPSER'
#
MESSAGE 29 : 10-OCT-82 10:48 AM  JOB:6 DET QUMRUN[1,2]
        1 SPOOLER(S) ON-LINE -
(25) LP1SPL LP1: FORMS=NORMAL;
```

PLEASE passes the full command line to OPSER and notifies the operator before it reprints the prompt. OPSER recognizes the text INT as a request to send unsolicited text to a spooling job. The characters #6 in the command line are recognized as the number of the destination job to which the text must be sent. QUEMAN receives the text STATUS, generates a response, and sends the response back to OPSER. OPSER formats a message and displays it on the OSC for the operator.

The STATUS command itself is the operator's way of getting information about jobs that are on-line to QUEMAN. QUEMAN's status printout tells the number of spooling jobs currently on-line and supplies data on each job (job number within parentheses, receiver identification, physical unit being spooled, and program default conditions). If the particular spooling job is

processing a request, the data in the printout includes the following information about the request: its name, project-programmer number under which it is queued, and its sequence number. The STATUS command also tells you if SPOOLING or QUEUING is disabled.

The NEXT command allows the operator to place a job request at the head of a queue. Job requests are identified by the name, account, and sequence number under which the request is queued. This format is described in the QUE program discussion in the *RSTS/E System User's Guide*. A request moved to the head of a queue is the next one QUEMAN sends to a spooling program servicing that queue.

If QUEMAN detects an error in the NEXT command, it generates a message in the format:

```
'NEXT' CMD: string - text
```

The string is the command typed and text is one of the following phrases:

Text	Meaning
ILLEGAL PARAMETERS	Only a queue name and request name, separated by an equal sign, are allowed in NEXT command.
ILLEGAL SYNTAX	A space or equal sign is missing.
JOB IN PROCESS	The specified job request has already been sent to spooling program.
MULTIPLE JOB SPECIFIED	The specified request name matches two or more jobs in the queue.
NO MATCH FOR JOB	Request name does not match any job in queue.

5.4.3 QUEMAN Start-Up Procedure

You should place commands in the system start-up control file to start the QUEMAN program. (Refer to Section 3.1 for information about creating a start-up control file.) The following description contains some suggestions about the QUEMAN commands you should include in the control file.

Use the following commands to start QUEMAN:

```
/PRIORITY:0
DETACH
```

The priority is set to 0 because most jobs on RSTS/E run at priority –8 and QUEMAN, running at priority 0, can more readily process, because it has a higher priority, the queue file.* Include the DET command to make QUEMAN run detached on your RSTS/E system.

*Without the PRIORITY switch, QUEMAN automatically sets its priority to 0. You can explicitly specify the switch to provide a record of what the program does automatically.

Because QUEMAN examines the QUEUE.SYS file when it starts, there is no need to explicitly initialize the file. If the file contains bad data, the QUEMAN program resets all entries to initial states. Therefore, you should specify the INI command only in situations in which you want to explicitly initialize the queue file.

5.4.4 QUEMAN Action Under Various Start-Up Conditions

QUEMAN start-up actions depend on the way QUEMAN last terminated:

- When QUEMAN runs after you shut down operator services in an orderly fashion through SHUTUP, a normal start-up procedure occurs. QUEMAN finds that the QUEUE.SYS file was closed properly. The program performs consistency checking but does not print any messages unless it finds problems. If the data is consistent, QUEMAN retains all job requests in the queue file and examines each request for completeness and status. Because QUEMAN normally starts before any spooling programs, there are no entries in the online spooler table. QUEMAN checks to see that this table is clear of entries.
- When QUEMAN starts either after the system crashes or after QUEMAN itself terminates unexpectedly, it finds the queue file improperly closed and generates a message to that effect. The program performs consistency checking on the queue file structure and on the data in the QUEUE.SYS file itself, generates messages telling the results, and reestablishes communication with all spooling jobs.

5.4.5 QUEMAN Consistency Checking

The QUEUE.SYS file has room for 250 queued requests. QUEMAN creates an entry for a queued request from an entry in a free list. A request for a line printer or batch queue requires one entry from the free list. A request queued with an AFTER date and time, however, requires two entries from the free list – one for the proper queue and one for the AFTER queue.

In performing consistency checking on the QUEUE.SYS file, QUEMAN initializes all entries if it finds one of the following conditions:

1. An entry in the free list is also in a queue or AFTER list
2. The root of the free list is outside the legal range
3. A request queued with an AFTER date and time that has not expired does not have an entry in the AFTER queue

For other inconsistencies, QUEMAN may either remove a single entry or perform some related action. If an entry in the AFTER queue has no corresponding entry in a spooling queue, QUEMAN removes the AFTER entry. If an entry in either an AFTER or spooling queue is not completely set up, the program removes the entry. (QUEMAN may have terminated while it was processing the request.) To signal this event, QUEMAN generates a message in the format:

```
%LPn: 'reqnam' [p,pn];QUEUED JOB INCOMPLETE - REMOVED FROM QUEUE
```

If an entry to be killed still remains, QUEMAN removes it and generates a message in the format:

%LPn: 'reqnam' [p,pn];QUEUED JOB IN 'KILL' STATUS - REMOVED FROM QUEUE

If QUEMAN finds that an entry has been processed (sent to a spooling program) but not yet completed, it places the request in a hold status and generates text in the format:

%LPn: 'reqnam' [p,pn];JOB PREVIOUSLY SENT TO SPOOLER;WILL BE PUT INTO HOLD STATUS

The requester can remove the job request from hold status.

QUEMAN checks the on-line spooler table in the QUEUE.WRK file for consistency. If the program finds the table empty, it generates the message:

```
# SPOOLERS ONLINE = 0; WILL CLEAR TABLE
```

If the program finds that the count of on-line spooling jobs is larger than the limit (16), it clears the table and generates the message:

```
ON-LINE SPOOLER TABLE CORRUPT - WILL CLEAR TABLE
```

If QUEMAN clears the on-line spooler table but spooling jobs are still running, the operator must terminate those jobs through OPSER and restart each spooling program. This action allows QUEMAN to put each spooling job properly on-line again.

For each spooling program found in the on-line job table, QUEMAN removes the entry and generates the message:

'rcvrid' (nn) FOUND ON-LINE-TAKEN OFFLINE

QUEMAN sends the job a message requesting that it declare itself on line to QUEMAN again. The spooling job does not respond to the message while it is processing a queued request. The answering message indicating that the job is again on line to QUEMAN does not appear until the job completes the current request. When QUEMAN receives the on-line declaration from a spooling job, it generates text in the format:

'rcvrid' (nn) PUT ONLINE

5.5 Line Printer Spooling Program — SPOOL

The line printer spooling program SPOOL runs without operator intervention and executes queued requests to transfer disk files to a line printer or terminal. The program consists of three modules:

1. SPOOL establishes initial conditions on start-up and checks initial conditions upon restart.
2. SPLIDL executes when no job is currently being printed.
3. SPLRUN prints any spooled file.

Each module is compiled and stored in the Spooling Package Library, has a protection code of <124>, and requires permanent privilege to run. For simplicity, the documentation refers only to one program, SPOOL, which functionally includes the three modules.

The SPOOL program runs only if the job has permanent privilege and the OPSER and QUEMAN programs are running. To start SPOOL, type the following command while logged in to the system under a privileged account:

```
RUN $SPOOL
SPOOL V8  RSTS V8  TIMESHARING
#
```

At this point, the SPOOL program performs a number of preliminary checks before it prints the pound sign (#) prompt. The remaining portion of this section describes these initial program checks. Once you are familiar with this procedure, you can then learn in Section 5.4.1 about the options SPOOL accepts in response at its program prompt. A description of these SPOOL program checks follows.

If you are not using a privileged account, the system displays an error message to indicate that you do not have access to SPOOL.

If the job has permanent privilege, SPOOL runs and prints its identification line and the pound sign (#) prompt. In response to the prompt, give a specification in the following form:

logical device:/start-up switch(es)

The logical device is the name of the queue and the unit number within that queue from which this copy of SPOOL takes job requests to execute. This name ordinarily corresponds to the physical device on which SPOOL prints the requests; the physical device, however, may be changed by the PHYSICAL start-up option described in Section 5.5.1.

The logical device name must have the form LPn: or LP:, where n is a unit number from 0 to 7. If the name is LPn:, the spooling job prints only requests queued either to that explicit unit or to the general spooling queue LP:. If LP: is specified, this copy of SPOOL prints requests in the printer queue regardless of the unit to which they were originally queued.* The logical device specified as the queue name must be a line printer; the device name given must not have a logical assignment to some other device.

You can use any combination of start-up switches described in Section 5.5.1 to condition the operation of the SPOOL program. A switch is formed by a slash (/) character and an option that may take an operand or other switches.

*The general spooling queue LP: is useful only on systems having line printers with similar characteristics.

5.5.1 SPOOL Start-Up Options

Start-up options alter default processing conditions. You use these options to:

- Spool output to a different physical device
- Change form information
- Control job environment

Table 5–7 summarizes these options. Note that the symbol (_) is an underscore character and not a space.

Table 5–7: SPOOL Start-Up Options

Option	Syntax and Meaning
ASSIGN	/ASS[IGN] Reserves the physical device for this job. Without this option, SPOOL retries continually whenever it cannot gain access to the unit and is ready to print a job.
FORM	/FOR[M]:name switch Defines the current form according to a name (NORMAL is the default name) and switches as follows:
	/ALI[GN] — Causes SPOOL to execute a forms alignment procedure before processing any further queued requests.
	/DFL[ENGTH]:nnn — Declares the device form length as nnn lines, where nnn is a number from 1 to 127. The length is that understood by the hardware of the output device. For example, on a line printer in the United States, the standard length is usually 66 lines per page.
	/HEA[DINGS]:n — Causes SPOOL to print n (from 0 to 3) heading burst pages preceding each job request. Unless NH accompanies a file specification in the QUE command, SPOOL also prints n burst pages preceding each file in a job request. The default value is 1.
	/LEN[GTH]:n — Defines the length of software form as n lines per page, where n is a number from 1 to 127. If this switch is not given, 66 lines per page is assumed.
	/PAG[E_EJECT]:yes — Indicates that the device to which this copy of SPOOL directs output has a hardware top of form capability and that the device interprets the CHR$(12) CHARACTER as a top of form command. This condition is the default case for line printer devices.

(continued on next page)

Table 5–7: SPOOL Start-Up Options (Cont.)

Option	Syntax and Meaning
	/PAG[E_EJECT]:no — Conditions the software to translate a form feed character CHR$(12%) into the proper number of line feed characters because the hardware does not recognize the character as a top of form command. This condition is the default case for SPOOL output directed to terminal devices by the PHYSICAL option. /WID[TH]:n — Defines the width of the heading burst page as n characters per line. If this switch is not given, the default value 132 (which is the maximum) is used.
LPFORM	/LPF[ORM]:YES Indicates that LPFORM characters are interpreted by the system software for the output device. During normal transfer, the program passes LPFORM characters unmodified to the device. This condition is the default for line printer devices. /LPF[ORM]:NO Indicates that the system software does not interpret LPFORM characters. During normal transfer, the program converts each LPFORM character to an appropriate number of line feed characters. This condition is the default for terminal devices. Note that this option affects only jobs queued with the /LPFORM switch. The SPOOL program does not automatically invoke /LPFORM processing.
NAME	/NAM[E]:rcvrid Places the specified identification in the system message receiver table and in OPSER's on-line job table. This identification rather than the default identification is used by the operator in OPSER commands to access this spooling job. This identification must not be in use by some other job.
PHYSICAL	/PHY[SICAL]:dev: Uses this device as the physical device for this spooling program. Requests in the queue specified by the logical device are printed on this device. The default value is the logical device specified as the queue name.
PRIORITY	/PRI[ORITY]:nnn Sets the job priority to nnn, where nnn is a number from –120 to 120. Without this option, the priority remains as set by LOGIN. Do not use /PRIORITY after a /FORM, /NAME, or /PHYSICAL option, or after an LPn: designator.
RUNBURST	/RUN[BURST]:nnn Sets the job run burst to nnn, where nnn is a number from 1 to 127. Without this option, the run burst remains as set by LOGIN.

The PHYSICAL option alters the physical device on which queued requests are printed. For example, requests queued for line printer unit 1 could be processed on another line printer unit or on a keyboard unit. The actual device, in any event, differs from the one for which the requests were queued.

The ASSIGN option reserves the actual spooled device to the job. Without this option, SPOOL periodically tries to gain access to the spooled device whenever SPOOL is ready to print and another job has ownership of the device. When SPOOL terminates, it deassigns the device.

The NAME option allows the receiver identification to be different from that normally assigned by SPOOL. The identification should be from one to six alphanumeric characters and must not currently exist in the system message receiver table. The identification must not begin with a number because the operator uses either this identification or a job number to identify a spooling job.

The FORM option allows the default form information to be altered. Individual characteristics of the NORMAL form can be altered, a new form with a different name can be defined, or a forms alignment can be requested. A forms alignment can be requested by itself or in conjunction with a form alteration. When a form name other than NORMAL is in effect, user's queuing requests must explicitly specify the form name to have files printed with its characteristics.

To alter the default characteristics of the NORMAL form, the operator can specify the appropriate auxiliary switches without the /FORM: switch. For example, to change the number and width of heading burst pages for line printer unit 0, type:

```
LP0:/HEADINGS:2/WIDTH:80
```

Two heading burst pages with a width of 80 characters per line are printed for each job request and each file within a request. The width used applies only to the burst pages, not to the data being printed.

Specifying a form with a name other than NORMAL usually means that the operator must load special paper in the output device. The /ALIGN switch with the FORM option requests a forms alignment. The operator then must align the form at system start-up time. A SPOOL interrupt command with the /ALIGN switch can request a form alignment during timesharing. Section 5.5.10 describes the forms alignment procedure.

The following message can occur when the line printer queues are empty, but jobs with nondefault form names exist:

```
NO JOBS WAITING WITH FORMNAME 'XXXXXX' FOR SPOOLER 'LPn:'
OTHER JOB(S) WAITING.
***PLEASE INSPECT QUEUE AND TAKE APPROPRIATE ACTION***
```

The operator should then decide whether or not to use an interruption command to change the form name so that the other jobs can be queued.

The /PAGE_EJECT switch determines whether the SPOOL program counts lines to effect forms control. (The /DFLENGTH switch can modify the effect of the /PAGE_EJECT switch.) A value of NO with the /PAGE_EJECT switch indicates that the output device (driver) does not understand a form feed character; that is, a form feed character does not cause the device to position itself at top of (paper) form. With NO in effect, SPOOL counts lines to effect forms control. A value of YES with the /PAGE_EJECT switch indicates that the device (driver) translates the form feed character to an appropriate number of line skips* to place the device at top of form. With YES in effect, SPOOL need not count lines but may be able to pass all forms control data unaltered to the device driver. (The /DFLENGTH switch can modify the effect of YES.)

If NO is in effect for the /PAGE_EJECT switch, SPOOL ignores any value specified in a /DFLENGTH switch. The operator should specify /PAGE_EJECT:YES for any terminal device that has a top of form capability.

SPOOL uses the value specified in the /DFLENGTH switch only if /PAGE_EJECT:YES is in effect. The /PAGE_EJECT:YES switch can be in effect either through default (on a line printer) or by explicit specification (for a terminal device having the form feed capability). The operator may specify the /DFLENGTH switch either at start-up time in a SPOOL start-up switch or during processing in a FORM interrupt command. For proper formatting, the DFLENGTH value must be the maximum number of lines that the output device skips when it receives a form feed command. This number of lines is called the device form length. This number may differ from the length of the paper form being used.

Different devices have different form lengths. Three examples can show how the device form lengths vary:

1. The LA180 device has a switch labeled Length of Form, which has settings for 4, 8.5, and 11 inches. These settings correspond to 24, 51, and 66 lines per form. (The number of lines per form given for each setting assumes the standard 6 lines per inch on the LA180.) If, when starting the spooling job, the operator sets the switch on the LA180 to 8.5 inches, the operator should also specify the /DFLENGTH switch

*SPOOL issues three forms-control characters: 1) form feed (the CHR$(12) character, whose ASCII value is 12), 2) carriage return (the CHR$(13) character, which is used for overprinting), and 3) line feed (the CHR$(10) character). No other forms-control characters are used. SPOOL has no provisions for changing or extending the set of control characters for any special device(s).

The terms line feed and line skip are not synonymous. Line feed refers only to the CHR$(10) character, which either SPOOL or the device driver sends to a device. Line skip refers to the movement of paper that the device itself effects.

with a value of 51. If, at some later time, the operator changes the setting on the LA180 to 4 inches, the operator should also specify a new /DFLENGTH switch with a value of 24 in a FORM interrupt command to the spooling job.

2. Many line printers have only one form length. If the hardware length is 66, the value in the /DFLENGTH switch should be 66. (The default SPOOL uses in absence of the option is 66.) If the hardware form length is 51, the operator should specify the value 51 with the switch when starting the spooling job.

3. Some printing devices have an adjustable top of form. On these devices, the operator can adjust the device to execute an arbitrary number of line skips when it receives a form feed character. This adjustment is most often on terminal-type devices with high-quality print characteristics. When resetting the top of form for this type of device, the operator should also specify the DFLENGTH and LENGTH options with the new form length information. The operator should also request a forms alignment for each new form.

The effect the /DFLENGTH switch has on the SPOOL program differs for line printers and terminal devices:*

1. For a line printer, the DFLENGTH value affects only the procedure SPOOL invokes when it receives an ABORT command while it is processing a job request. If the DFLENGTH value equals the current length of paper form, the program issues a form feed character to position the device at the top of the next form. If the values are not equal, the program automatically invokes the forms alignment procedure to reestablish the top of form position.

   The automatic forms alignment procedure requires operator action to align the form and to respond to requests SPOOL makes. This is necessary only when the program has no way to determine where the top of form lies after an ABORT operation.

2. For a terminal, the DFLENGTH value controls the way SPOOL keeps its place on a form and the way the operator handles recovery procedures. If the value of the /PAGE_EJECT switch is YES, the equality of the device form length and the paper form length is important. If the device form length and paper form length values are not equal, the SPOOL program counts lines to maintain a record of its position on a form and issues an appropriate number of line feed characters to simulate form feed. If these values are equal, SPOOL does not count lines and simply sends a form feed character to position the paper at top of form.

*A line printer is a device whose interface to the computer is through a line printer controller and whose designation is in the form LPn:. A terminal is a device whose interface to the computer is through a keyboard line and whose designation is in the form KBn:. This distinction is critical because some devices such as the LA180 are called printers but may be connected to the computer through a keyboard line.

If the value in effect for DFLENGTH does not correctly match the setting of the hardware device, two undesirable conditions exist:

1. SPOOL must incur the overhead of counting lines.
2. Forms alignment may be lost during processing.

For these reasons, DIGITAL highly recommends that the operator supply a new value of DFLENGTH when changing the hardware setting on the device.

5.5.2 Line Printer Spooling

If the physical device specified at start-up time is a line printer unit, SPOOL applies the following default values:

Switch	Result
/DFLENGTH:66	The device form length is 66 unless otherwise specified. If you have patched the monitor to apply a different default form length, you must always specify the correct value in a /DFLENGTH switch.
/FORM:NORMAL /HEADINGS:1 /LENGTH:66 /WIDTH:132	The values shown are used as the form definition.
/LPFORM:YES	For any line printer, SPOOL assumes that the device driver properly processes the special LPFORM characters (refer to the *RSTS/E Programming Manual*).
/PAGE_EJECT:YES	For any line printer, SPOOL assumes that the device driver properly processes top of form characters.

5.5.3 Keyboard Spooling

If the physical device specified at start-up is a keyboard device, SPOOL applies the following default values:

Switch	Result
/DFLENGTH:66	DFLENGTH has no effect unless PAGE_EJECT is explicitly specified as YES.
/FORM:NORMAL /HEADINGS:1 /LENGTH:66 /WIDTH:132	The values shown are used as the form definition.
/LPFORM:NO	The terminal driver does not process LPFORM characters. For any file that contains LPFORM characters, SPOOL simulates the LPFORM effect by issuing line feed characters.
/PAGE_EJECT:NO	Most terminals do not have a hardware top of form capability. SPOOL simulates top of form by issuing the correct number of line feed characters.

For terminals that do have a hardware top of form capability, the value of PAGE_EJECT should be YES. The value for DFLENGTH should reflect the actual device form length, as discussed in Section 5.5.1.

For terminals with the hardware top of form capability: 1

1. Set the value of PAGE_EJECT to YES.
2. Set the value for DFLENGTH to the actual device form length (see Section 5.5.1).
3. Set the proper characteristic of the device that accepts and processes a form feed. Use the TTYSET FORM command to perform this operation.

5.5.4 Start-Up Error Processing

If you make an error when responding to the pound sign (#) prompt, SPOOL prints a message and the unparsed command line in the format:

```
?ERROR MESSAGE
unparsed command line
```

The unparsed command line is the righthand part of the response beginning at the element causing the error. Table 5–8 summarizes the messages that are possible.

If SPOOL encounters no syntax errors in the response, it begins setting initial conditions. Errors in this phase are reported by printing a message in the format:

ERROR IN SOME OPERATION – RESTARTING
error message

The program reprints its identification line and the prompt.

An error message ?DUPLICATE RECEIVER ID means that the receiver identification, either the default one or the one specified in a NAME option, is already defined for another job. The operator should type the command line again and specify a unique receiver identification in the NAME option.

The message ?NO ROOM IN RECEIVER TABLE means that a general small buffer is not available to allow SPOOL to declare itself a receiving job. A later retry with the same command line should succeed.

The following message means that SPOOL attempted to send a message to that program and failed:

```
*****OPSER HUNG***** or *****QUEMAN HUNG*****
```

Either the named program does not have an entry in the system message receiver table or the program is not processing its messages and its message limit has been reached. The operator must determine the cause of the problem and restart the named program.

Table 5–8: SPOOL Syntax Error Messages

Message and Meaning
?CAN'T PARSE REMAINING STRING Something illegal or undefined was found in the command line. An undefined switch or missing slash (/) character causes this error.
?DUPLICATE SWITCH Two occurrences of the same switch were found in the command line.
?ILLEGAL LOGICAL DEVICE The logical device specified was not in the form LP: or LPn: (where n is a number from 0 to 7).
?ILLEGAL OPERAND An illegal operand was found in an option or switch. For example, specifying nonnumeric characters in a switch that requires a number (/WIDTH:n) generates this error.
?ILLEGAL PHYSICAL DEVICE The device specified in the PHYSICAL option is not a line printer or keyboard device.
?MISSING OPERAND A switch requiring an operand was specified without one. For example, if a receiver identification is missing from the NAME option, this error is generated.

5.5.5 SPOOL Interrupt Commands

The operator communicates with a SPOOL job by means of INTERRUPT commands sent through the operator services program OPSER. Table 5–9 summarizes these commands. Note that any responses to an INTERRUPT command are displayed on the Operator Services Console.

Table 5–9: SPOOL Interrupt Commands

Command and Syntax	Meaning
ABO[RT]	Immediately terminates the current process and removes the request from the queue.
CON[TINUE]	Wakes up the spooling job to continue processing after a PAUSE command.
END	Closes out processing after completing the current request. The operator must then run the spooling program again to process further requests.
FOR[M]^name/switch	Changes the current output form to one identified by name and defined by switches. These switches change current form characteristics; those characteristics not changed by a switch remain at their current definitions. See the start-up switch FORM description in Table 5–7 for the allowable switches. Alignment is done only if the /ALIGN switch is included.
FOR[M]	Displays characteristics of the current form if no name or switches are in the command.
FOR[m]^/ALIGN	Requests a forms alignment procedure for the current form.
LAS[T]	Prints the most recent message generated by the spooling job.
OFF[LINE]	Immediately terminates all processing by this spooling job (same as ABORT followed by END).
PAU[SE]	Places the spooling job in a sleep state, during which it responds to most commands and resumes normal processing in response to a CONTINUE command.*
REQ[UE]	Stops processing the current request and replaces the request in the queue so that processing later resumes at the terminating point.
RES[TART]	Reprints the current copy of a file from the beginning, including heading burst pages.
RES[TART]^JOB	Reprints the current iteration of the job, including heading burst pages.
RES[TART]:n	Reprints the current copy of the file starting at page n. If n is 0, restarts from the beginning of the file but omits the heading burst pages.
STA[TUS]	Prints a status report for the spooling job.

*Note that processing continues if any job modification command such as ABORT or REQUE is sent.

The symbol ^ marks the location of a required space.

5.5.6 SPOOL Start-Up Examples

This section shows four examples of the start-up procedure. They represent typical cases.

5.5.6.1 Line Printer Start-Up with All Defaults – In response to the SPOOL program prompt, the operator types:

```
#LP0:
```

If SPOOL encounters no errors, it prints the message:

```
DETACHING...
```

SPOOL detaches itself and leaves the terminal logged off the system.

The following are the characteristics of this copy of SPOOL:

- The program prints only job requests that were queued to line printer unit 0 with a form name of NORMAL or queued to the general line printer queue with a form name of NORMAL.
- The program prints on line printer unit 0.
- The program prints one heading burst page before each job and, unless a file is queued with NH, prints one heading burst page before each file. The width of the burst page is 132 columns, and the form length is 66 lines.
- The program assumes the device driver handles LPFORM characters.
- The program assumes the device form length is 66 lines.
- The run burst and priority of SPOOL are those in effect when the job starts.
- When the program receives a job from QUEMAN, it attempts to assign the printer. If the assignment fails, the program retries periodically until it succeeds. When it finishes processing a job request, the program deassigns the device.

The operator may later change all form characteristics (name, length, width, device form length, and headings count) through appropriate interrupt commands.

5.5.6.2 Line Printer Start-Up with Narrow Width – In response to the prompt, the operator types the command line:

```
#LP1:/FORM:NARROW/WIDTH:80/RUNBURST:12/ASSIGN
```

The following are characteristics of this copy of SPOOL:

- The program prints only jobs queued with a form name of NARROW and queued to either unit 1 or the general line printer queue.
- The program prints on the line printer 1 device.
- The program prints one heading page of width 80 columns; form length is 66.

- The program assumes the device handles top of form.
- The program assumes the device driver handles LPFORM characters.
- The program assumes the device form length is 66.
- The priority of the job remains unchanged. The program sets the run burst to 12.
- The program does not begin spooling until it can assign line printer unit 1 and does not deassign the device until it terminates.

In this example, assume that line printer unit 1 is a relatively slow device and that the SPOOL program can generate output data for this unit faster than the device can print. Optimal use of the device results if the unit keeps running at full speed for the entire job request it is printing.

To effect optimal use, it is sometimes necessary to raise either the priority or run burst of the SPOOL program. The priority of a job generally affects how often it runs, whereas the run burst affects the length of time the job runs once it starts running. Because the SPOOL program is usually I/O bound and waiting for the device to complete printing, there is little reason for it to run more often. Changing the priority is usually unnecessary and ineffective. Changing the run burst, however, may improve processing.

When SPOOL starts executing, it remains in the run state until one of the following events occurs:

- The job's run burst expires. That is, the job actually executes for its full run burst.
- The job requests some type of I/O and the system cannot immediately satisfy the request.

For the spooling program, the second event is much more likely because the system buffers only a fixed amount of data before it refuses to satisfy further requests for output. When the refusal occurs, the system activates another job and the spooler must wait until the system sends at least some of the already buffered data to the output device. (This wait condition shows as an LP or TT STATE on a SYSTAT listing.)

As long as the buffers for the device are never completely emptied, the system keeps the device running at or near full speed. To a certain point, raising the spooling job's run burst makes it more likely that the job will continue running until it completely fills the buffers for its output device. After a certain point, raising the run burst has little benefit because the job can run only until the buffers are full. Conversely, decreasing the run burst causes the system to deactivate the program before the buffers are full. There is more likelihood that the printer can empty the buffers before the system reactivates the job.

If a spooling job spends a significant time in the RN state while it is actually printing, the run burst is too low. DIGITAL recommends that you raise the run burst to a value at which the job spends most of its time in an output wait state.

5.5.6.3 Keyboard Start-Up on an LA36 – For this example, assume that keyboard unit 5 is an LA36 terminal. To the prompt, the operator types a command line as follows:

```
#LP2:/PHYSICAL:KB5:/HEADINGS:0/ASSIGN
```

The following are the characteristics of this copy of SPOOL:

- The program prints only jobs queued with a form name of NORMAL and queued to the unit 2 or the general line printer queue.
- The program prints on keyboard unit 5.
- The program prints no heading burst pages. The form length is 66.
- The program assumes that keyboard unit 5 does not handle top of form, and, therefore, it counts lines as it prints them to simulate top of form.
- The program assumes the device driver does not handle LPFORM characters and, therefore, it simulates LPFORM with line feeds.
- The run burst and priority remain unchanged.
- The program assigns keyboard unit 5 at start-up and keeps it assigned until it terminates.

Because the LA36 is a slow device, processing is not slowed any further with burst pages. The /HEADINGS:0 switch suppresses printing of burst pages.

To retain any forms alignment on the LA36, the device is permanently assigned to the spooling job. This assignment prevents any other job from using the device and leaving the carriage at a position other than the top of form. When the spooler starts processing a request, it must assume the device is at its top of form position because the program has no way to force the device to advance to some known position. If the form is not aligned properly when SPOOL starts the output, the entire output is misaligned.

5.5.6.4 Keyboard Start-Up on an LA180 – For this example, keyboard unit 2 is a serial LA180 DECprinter. This terminal has a hardware top of form capability and obeys the XON/XOFF synchronization protocol. In addition, the form length selector switch on the terminal is set to 8.5 inches (51 lines per form) and the paper form is also 51 lines per page.

In response to the prompt, the operator should type a command line as follows:

```
#LP3:/PHYSICAL:KB2:/PAGE_EJECT:YES/DFLENGTH:51/LENGTH:51-
MORE>    /WIDTH:40/FORM:SPECIAL
```

The following are the characteristics of the SPOOL copy:

- The program prints only requests queued with a form name of SPECIAL and queued to either the unit 3 or the general line printer queue.
- The program prints on keyboard unit 2.
- The program prints one heading page that is 40 columns wide and 51 lines in length.
- The program assumes the device handles top of form.
- The program assumes that the device driver does not handle LPFORM characters. The program therefore simulates LPFORM with line feeds.
- The run burst and priority remain unchanged.
- The program assigns the keyboard before it prints each request and deassigns it after printing the request.

Because:

- The LA180 is a much faster device than the LA36, SPOOL can print a small number of burst pages to simplify distinguishing jobs and files. The narrowest width for the burst page is 40 columns.
- The paper form length and the device form length are equal, SPOOL can position the device to top of form by issuing a form feed character.
- The program can position to top of form, it does not need to keep the unit assigned.

The ASSIGN option, therefore, does not appear in the example. If the paper form length did not equal the hardware setting, however, SPOOL should keep the device assigned to preserve the top of form position.

5.5.7 Recovery from Line Printer Errors

The interaction between the spooling job and the OPSER program controls error handling in the SPOOL program. For example, if the paper runs out or jams, SPOOL encounters the error message:

```
?DEVICE HUNG OR WRITE LOCKED
```

It discontinues processing and sends OPSER a message. OPSER generates a message for the operator:

```
MESSAGE         3 : 28-OCT-82 04:33 PM  JOB:21  DET SPLRUN[1,100]
    LP0   : HUNG  --   JOB: [220,40]USER1
(PUT DEVICE ONLINE TO CONTINUE)
```

The message tells the operator that the spooling job for line printer unit 0 has a problem and that the device needs to be made ready again before processing can continue.

For a SPOOL program servicing a line printer, there are certain errors from which you can easily recover. The device handler for the line printer tests the ready status of an off-line unit every 10 seconds. If the operator corrects the error (for example, fixes a paper jam) and puts the line printer on line again, the software detects the ready status and continues printing buffered characters. The SPOOL program finds that the error condition is cleared and continues processing.

If a special form is being processed and SPOOL encounters an error, SPOOL may request the operator to realign the forms before it continues processing on its own.

If the operator cannot correct the error or wishes to perform some other operation, the spooled device may be left off line. This prevents the software from reprinting before the program processes an operator request. The operator uses the INTERRUPT command with the proper text to access the spooling job.

SPOOL allows the operator several ways to control the restart of a queued request. The RESTART command can restart:

- The current copy of a queued request (RESTART JOB)
- The current copy of a file in a queued request (RESTART)
- The current copy of a file at a specific page (RESTART:nnn)

The REQUE command can reenter the current job in the queue for later processing. On receiving a RESTART command, the program waits for any currently buffered data to be printed before actually restarting.

The REQUE command causes SPOOL to remember the point at which processing was interrupted. After resuming the requeued request, SPOOL scans for the page on which processing terminated and continues printing at the start of that page. An error may occur if a user modifies the requeued request before SPOOL can resume the printing.

If the operator responds to an error condition with an ABORT command, which terminates the job, SPOOL clears the output buffers on the line printer unit. If the device is a terminal, SPOOL prints any buffered data. The program then issues a message to show that it is terminating.

Before terminating a queued request, the SPOOL program ensures that all buffers have been emptied successfully.

5.5.8 Line Printer Output

SPOOL generates job header and file header burst pages to identify print requests and files within a print request. Both types of header page contain identification and general accounting information as follows.

- The identification consists of large, easily readable block letters created from the character generation file CHARS.QUE. The job identification contains the account number of the user requesting the job and the name of the job. If no job name appeared in the QUE command, SPOOL prints the name of the first file in the request as the job name. The file identification shows the file name and type.
- General accounting information for the job header is on the burst page and is offset from the identification by two rows of special characters. The accounting information contains five lines of data:

  1. The first line contains the job name, current date, current time of day, and the requester's account number.
  2. The second line comprises the date and time of day of the request and the device for which the request was queued.
  3. The third line gives the system name.
  4. The fourth line gives the QUE options used to process the request.
  5. The fifth line gives the job copy number.

  The accounting information for the file header burst page appears below the identification and is framed, above and below, by two rows of special characters. The first line of information gives the job name used when SPOOL printed the file and shows the current date, time of day, and account as the job header burst page does. The second line gives the copy number, the QUE options used on the file, and the complete specification of the file. The third line gives the record type and carriage control format used to print the file.

If SPOOL does not print the file because of an error, file header identification and accounting information is replaced by an error message framed above and below by five rows of special characters. The error message is standard RSTS/E error text.

5.5.9 Error Messages During User Output

SPOOL reports errors it encounters during printing in the line printer output. SPOOL differentiates the error message from requested output by framing the text within five rows of special characters. Table 5–10 shows the possible SPOOL error texts and related meanings.

Table 5–10: SPOOL Error Text in User Requested Output

Text and Meaning
?CAN'T FIND FILE OR ACCOUNT The related file was deleted between the time it was queued and the time SPOOL tried to print it.

(continued on next page)

Table 5–10: SPOOL Error Text in User Requested Output (Cont.)

Text and Meaning
?FILE RESTART REQUESTED ON PAGE n The operator requested a restart at the indicated page or placed the job request back in the queue with the REQUE command. If the operator restarted the job request, SPOOL scans the file to find the correct page and continues printing on a new page following the message. If the operator requeued the job, this message appears when SPOOL restarts the job.
?JOB ABORTED Either the operator terminated the job request with an ABORT command or the K command of the QUE program was issued while the job was being printed.
?JOB ABORTED BECAUSE OF FILE ERROR(S) Errors in some file in the job prevented SPOOL from printing the remainder of this job.
?JOB REQUED AT PAGE n The operator placed this job request back in the queue with the REQUE command. SPOOL later continues the job request with a new header burst page and the text FILE RESTART REQUESTED ON PAGE n, where n is the same page number shown in the REQUE message.
`error text` SPOOL encountered the RSTS/E error shown, aborted processing the request, and cleared the request from the queue.

5.5.10 Changing and Aligning Forms

The FORM command can:

- Inform the spooler of a change in form on its device
- Request an alignment operation on the device
- Display information about the current form

Switches with the command specify the new form definition and request alignment. The command may request alignment when a new form is specified.

To request a form change, the operator sends the FORM command with the form name and the switches to define the form to the related spooling job. If an alignment is desired for the current form, the /ALIGN switch may be

included with the FORM command. The SPOOL program does not recognize the form change request until it becomes idle. If it is processing a queued request, it completes the processing before it performs the forms change. (To make the change immediately, request that SPOOL requeue the current request. The program requeues the request and becomes idle.)

To perform only an alignment, specify the FORM command without any form name and with only the /ALIGN switch. Again, the command takes effect only when SPOOL becomes idle. The program:

1. Prints characters to delineate left and right margins and prints, centered on the same line, the text TOP OF FORM.
2. Generates characters to position the paper on the last line of the form.
3. Prints characters to delineate the left and right margins, and prints, centered on the same line, the text END OF FORM.
4. Positions the paper at top of the next form.

SPOOL generates an action request and awaits a response. The operator may inspect the alignment and type one of two responses to the action request. If the alignment is not correct, the operator can adjust the hardware and send the response RETRY back to the spooler. In response to RETRY, the spooler performs steps 1 through 4 again and generates another action request. When the alignment is accurate, the operator can send the text GO as a response to the action request. The spooler returns to normal processing.

Whenever SPOOL processes any FORM command, it sends a message to the operator indicating the new form characteristics in effect. In any FORM command, only those options specified in the command change the form characteristics. Thus, if the operator sends a FORM command alone, without either a form name or any switches, the spooler displays the characteristics of the current form on the Operator Services Console.

5.6 Batch Processor Program — BATCH

The BATCH system program runs without user intervention and executes files of standardized commands queued for either a specific or the general batch processor. The program consists of four modules:

- BATCH sets initial conditions for batch processing.
- BATIDL checks status when no requests are being handled.
- BATDEC decodes and performs error checking on control files.
- BATRUN executes the decoded batch commands.

For simplicity, the documentation refers only to one program, BATCH, which functionally includes the four modules. Each module is compiled and stored in the Spooling Package Library, has a protection code of <124>, and requires permanent privilege to run.

To run BATCH, type the following command while logged in to the system under a privileged account:

```
RUN $BATCH
BATCH V8  RSTS V8  TIMESHARING
#
```

If you are not using a privileged account, the system displays an error message to indicate that you do not have access to BATCH.

If the job has permanent privilege, BATCH runs and prints its identification line and the pound sign (#) prompt. In response to the prompt, enter a specification in the following form:

```
logical device:/start-up switch(es)
```

The logical device is the name of the queue and the unit number within that queue from which this copy of BATCH takes job requests. This name does not specify the pseudo keyboard on which BATCH executes requests. The PHYSICAL start-up option described in Section 5.5.1 can specify the pseudo keyboard unit on which BATCH executes. The logical name must have the form BA: or BAn: where n is a unit number from 0 to 7. If BAn: is specified, the batch processor executes requests queued to either that explicit unit or the general batch queue BA:. If BA: is specified, the program executes job requests in the batch queue regardless of the unit to which they are queued.*

The logical device specified as the queue name must be a batch processor; the device name given must not have a logical assignment to some other device.

If only a queue name is given in response to the prompt and BATCH encounters no errors, the program establishes default processing conditions, prints the DETACHING message, and detaches itself from the terminal.

BATCH forms the default receiver identification from the queue name. BATCH also establishes other default conditions:

- BATCH tolerates no errors and terminates any queued request encountering a fatal or warning error.
- User log files are queued to the system default line printer with deletion specified.
- The run burst remains unchanged.
- The priority remains unchanged.

*The general spooling queue is useful only on systems with batch processors running with the same default conditions.

At start-up, BATCH performs the same error checking and processing SPOOL performs. Refer to Section 5.5.4 for the description of SPOOL start-up error processing.

5.6.1 BATCH Start-Up Options

Include options with the BATCH processor designator to control its operation. Table 5–11 summarizes these options, which are in the form of switches.

Table 5–11: BATCH Start-Up Options

Option	Syntax and Meaning
ASSIGN	/ASS[IGN] Reserves a pseudo keyboard device to this job.
ERROR	/ERR[OR]:severity Sets the error severity default for this processor so that errors of that severity are tolerated and queued requests having that or lesser severity end successfully. /ERR[OR]:FAT[AL] Tolerates all errors. /ERR[OR]:WAR[NING] Tolerates only warning errors. /ERR[O]R:NON[E] Tolerates no errors (fatal and warning errors terminate queued requests). This is the default severity.
DELETE	/DEL[ETE] Deletes user log files after printing. This is the default condition unless the NODELETE option is specified.
NAME	/NAM[E]:rcvrid Places, in the system message receiver table, the specified receiver identification for this job instead of the default identification. The default for a batch processor is BAnSPL for specific processors and BASPL for the general processor.
NODELETE	/NODEL[ETE] Does not delete user log files after printing. Without this option, all log files are deleted. Requester may override this option with the /NOQUE switch in the $JOB command.
NOQUEUE	/NOQUE[UE] Does not queue user log files for printing. Without this option, all log files are queued without a queue name and with deletion unless changed by the /QUEUE and /DELETE switches. Requesters cannot override this option.
PHYSICAL	/PHY[SICAL]:dev: Uses the pseudo keyboard (for example, PK1:) as the device on which this processor executes user batch jobs.
PRIORITY	/PRI[ORITY]:nnn Sets the job priority of BATCH to nnn, where nnn is a number from –120 to 120. Without this option, the priority remains unchanged.

(continued on next page)

Table 5–11: BATCH Start-Up Options (Cont.)

Option	Syntax and Meaning
QUEUE	/QUE[UE]:[quenam] Queues all user log files to the specified spooling queue (LP0: through LP7:) rather than to the general print queue. If no queue name is specified, all jobs are queued to the general print queue. Files are deleted after printing. Requesters may override queuing with the /NOQUE switch in the $JOB command. Deletion of the files by the SPOOL program may be overridden by including /NODELETE as a start-up switch.
RUNBURST	/RUN[BURST]:nnn Sets the job run burst of BATCH to nnn, where nnn is a number from 1 to 127. Without this option, the run burst remains unchanged.

5.6.2 BATCH Interrupt Commands

The operator uses interrupt commands sent through the operator services program OPSER to communicate with a BATCH processor. Table 5–12 summarizes these commands. Note that any responses to an interrupt command are displayed on the Operator Services Console.

Table 5–12: BATCH Interrupt Commands

Command and Syntax	Meaning
ABO[RT]	Immediately terminates the current process and removes the request from the queue.
CON[TINUE]	Wakes up the batch processor to continue normal processing after either a PAUSE command or after a user request for operator action (for example, in the $MESSAGE/WAIT command).
END	Terminates processing after completing the current request. The operator must then run the batch processor again to process further requests.
LAS[T]	Prints the most recent message generated by this batch processor.
NOT[ICE] text	Inserts the specified text in the user log file and precedes it with the heading: NOTICE FROM OPERATOR.
OFF[LINE]	Immediately terminates all processing by this batch processor (same as END followed by ABORT).
PAU[SE]	Places the batch processor in a sleep state, during which it responds to most commands and resumes normal processing in response to a CONTINUE command.
STA[TUS]	Prints a status report for this batch processor.

5.6.3 BATCH Start-Up Procedure

BATCH is most conveniently started by commands in the system start-up control file, as described in Section 3.1.

5.6.4 Operator Action Requests from BATCH

The BATCH program, in processing certain commands in a user command file, generates requests for operator action. These requests come through OPSER as action requests that require some operator action and a typed answer before the BATCH processor continues processing. These action requests result from commands requiring the operator either to mount and dismount volumes or to respond to a special message with which a stall is involved. The operator answers an action request with the ANSWER command through OPSER.

On a mount request, BATCH asks the operator to mount a device by an action request with text in the following format:

MOUNT xx:'logical id'/'vid'/WRITE/NOWRITE/DEN:nn/PAR:nn
DEVICE?

If a specific volume is involved, its identification appears as text replacing 'vid'. This text is the visual identification used to distinguish the volume. The specific device type is given by xx: from one of those shown in Table 5–13.

Table 5–13: BATCH Device Type Designators

Designator	Meaning
CD	CD11 punched card reader or the CM11 marked sense card reader
CR	CR11 high speed punched card reader
DB	RP04, RP05 or RP06 disk pack drive
DK	RK05 or RK05F disk cartridge drive
DM	RK06/RK07 disk cartridge drive
DL	RL01/RL02 disk pack drive
DP	RP02 or RP03 disk pack drive
DR	RM02/RM03/RM05 disk pack drive
DT	TU56 DECtape drive
DU	All UDA disks: RA60,RA80,RA81,RD51, RC25, RX50
DX	RX01/RX02 floppy disk drive
LP	Any line printer
MM	TU16, TE16, TU45, or TU77 magnetic tape drive
MS	TS11, TSV05, or TU80 magnetic tape drive
MT	Any magnetic tape drive
PP	High speed paper tape punch
TT	Any keyboard line

If a specific volume is indicated in the text of the action request, the operator should mount the medium on a free unit of the device type specified by xx:. If a volume is not involved, the operator should make sure a unit of the type specified by xx: is ready and on line to the system.

When the unit is ready, the operator must type the device specification (device type and unit number with a colon) as text in the ANSWER command for that action request number. BATCH assigns that unit and continues processing the user's queued request.

When BATCH is done with the device, it generates an action request to dismount the volume or device. If the request involves a volume, the operator should remove the volume from the device specified and place the unit off line. To continue the BATCH processing, the operator should type the CONTINUE command as text in the ANSWER command to OPSER for that action request number. This response tells BATCH to deassign the unit and to continue processing.

5.7 Operator Communication Program — PLEASE

The PLEASE program communicates directly with the operator services program OPSER and, when OPSER is not running, sends text to the system console terminal (KB0:). The program PLEASE is compiled and stored in the system library account [1,2] with protection code of <232>. PLEASE requires privilege to execute, and the protection code allows all users to run it. You can restrict use of PLEASE to permanently privileged users by changing the protection code to <124>.

You, as the system manager or assigned operator, can run PLEASE to send commands to OPSER. Users who are not valid operators may run PLEASE to send text to the Operator Services Console.

5.7.1 Running and Terminating PLEASE

To run PLEASE, type the following command and then press RETURN:

```
RUN $PLEASE
PLEASE V8  RSTS V8  TIMESHARING
#
```

PLEASE prints its identification line and the pound sign (#) character as a prompt to indicate it is ready to accept text.

PLEASE accepts one line of text terminated with the RETURN key. A line of text exceeding the maximum length (255 characters) causes the program to print the LINE TOO LONG error message and reprint the prompt. If the first character of text is not a slash (/) character, PLEASE tries to send the text as a message to OPSER. If the first character of text is a slash, PLEASE tries to send the text as a command to OPSER. To show the operator that it is sending the message or command, PLEASE prints one of the following messages:

```
MESSAGE SENT TO 'OPSER'
COMMAND SENT TO 'OPSER'
```

The program prints the pound sign prompt again, after which the operator can type another line of text. Typing CTRL/Z or CTRL/C in response to the prompt terminates the program.

When OPSER is not active, PLEASE broadcasts messages on the system console terminal but does not send commands. In these situations, the program notifies the user with one of two messages:

```
%'OPSER' NOT ACTIVE - MESSAGE BROADCAST TO KB0:
%'OPSER' NOT ACTIVE - COMMAND NOT SENT
```

The message printed on keyboard unit 0 is preceded by the text:

```
%MSG FOR 'OPSER' BUT ITS NOT ACTIVE:
```

5.7.2 OPSER Commands through PLEASE

Any privileged user or valid operator can communicate with OPSER and with jobs on line to OPSER by typing a command through PLEASE. The user denotes an OPSER command by typing the slash (/) character as the first character in the line of text. Table 5–14 summarizes these commands.

Table 5–14: PLEASE Commands to OPSER

Command	Summary
/ANS^msgnumber:text	Responds to an action request.
/CHA^KBN:	Changes operator services console to keyboard unit n.
/DEL^msgnumber	Deletes outstanding action request.
/DEL^#nn:n	Deletes n oldest action requests for job number #nn.
/DET	Detaches OPSER from the OSC.
/EXI	Terminates OPSER.
/INT^rcvrid:text	Sends unsolicited text to on-line job.
/LIS^JO	Prints on-line job table.
/LIS^OP	Prints valid operator table.
/LOG^file;msglevel	Creates log file and sets message level.
/LOG^;msglevel	Changes message level of current log file.
/LOG	Closes current log file and stops recording.
/MES^msglevel	Sets message level for OSC.
/OPE^KBn:[nnn,nnn]	Adds keyboard and account combination to valid operator table.
/OPE^-KBn:[nnn,nnn]	Removes keyboard and account combination from valid operator table.
/RET^msgnumber	Prints associated message.
/RET^m:n	Prints oldest m action requests for job number #n.
The symbol ^ marks the location of a required space.	

When OPSER receives a command from PLEASE, it ensures that the user's keyboard and account numbers are present in the valid operator table. If the user is not a valid operator, OPSER broadcasts the error message ?INVALID OPER on the terminal from which the command was made.

Commands sent by a valid operator can be executed by OPSER itself or by a program on line to OPSER. If OPSER itself executes the command, any response from the command is broadcast on the originating terminal. For example, because OPSER itself processes the LIST JOBS command, the on-line job list generated appears on the originating terminal. An INTERRUPT command, however, passes text to another program. Any response from the other program is displayed on the OSC and not on the originating terminal. To see a message or response from an INTERRUPT command, the operator should change the OSC to the current keyboard (the line on which PLEASE is running) before transmitting the command. Action requests and messages (responses to commands or text sent to OPSER) are displayed only if the current message level allows.

5.7.3 PLEASE as a CCL Command

If you install PLEASE as a CCL command, the operator can type PLEASE followed by a text line to send commands to OPSER. PLE is the standard abbreviation for the CCL command.

5.8 Terminating Operator Services and Spooling

Usually you terminate OPSER and programs on line to OPSER by running the SHUTUP program. SHUTUP, however, terminates time-sharing operations. If it is necessary to terminate operator services and spooling without shutting the system down, the operator can type a sequence of commands to stop spooling operations in an orderly fashion. To understand the sequence, you must know about OPSER shutdown levels.

5.8.1 OPSER Shutdown Levels

When OPSER receives a command from SHUTUP to perform an orderly shutdown in logical end mode, OPSER sends an END command to each on-line job in a fixed, logical sequence. OPSER selects by shutdown level the order in which jobs are ended. On-line jobs at the lowest level terminate first. When all jobs at one level have terminated and have been removed from OSPER's internal tables, OPSER begins sending END commands to jobs at the next highest level. When all on-line jobs have properly terminated, OPSER kills itself and SHUTUP proceeds with the system shutdown.

5.8.2 OPSER Manual Shutdown Procedure

You may need to shut down the SPOOL program without shutting down the entire system. The procedures to follow are:

1. The operator can issue the END command to the spooler at any time. This command causes the spooler to shut itself down the next time it is ready to get a new job. Thus, if the spooler is printing a job, it completes the job's output before shutting down. If SPOOL is waiting for a job, it shuts down immediately. The operator can use the ABORT command to terminate the job if the operator wants the job currently printing to end.

For example, to manually shut down the LP0SPL spooler in logical end mode, use the command:

```
PLE/INT LP0SPL:END
```

At the completion of the current job, the spooler kills itself. This shutdown procedure merely mimics what OPSER does on command from SHUTUP. The operator sends the END command, in turn, to jobs at each shutdown level. The output of the LIST JOBS command gives the shutdown levels for each on-line job. When OPSER displays END messages for all jobs at one shutdown level, the operator can send the END command to all jobs at the next highest shutdown level. After the on-line job table is clear and OPSER has generated END messages for all jobs, the operator can send the EXIT command to OPSER.

2. A second way of shutting down the spooler is with an OFFLINE command. The OFFLINE command immediately stops any job that is printing and prints an ABORT message on the listing to indicate the action taken. It also causes SPOOL to shut down after the current job finishes.

   Manually shut down the BA0SPL job immediately with the command:

   ```
   PLE/INT BA0SPL:OFFLINE
   ```

   BA0SPL aborts the current job, clears it from QUEUE.SYS, and kills itself.

If the operator wants to terminate a printing job for shutdown or any other reason, the line printer must be on line. Under no circumstances does the spooler consider a job completed until all the data that the spooler tries to print (including termination messages) is actually printed. If the operator aborts the job with the ABORT command, the spooler clears out, at the first opportunity, whatever data is already buffered; the line printer must be on line to allow this. If the operator wants no further output, use the UTILTY program to KILL the spooler job. In this case, no cleanup can be performed. This process may be necessary when certain hardware fails (for example, if the line printer interface does not raise the READY flag). It should only be used in a crisis. The job that was being printed should be killed from the queue file by a QUE/K command from the operator.

5.9 BACKUP as an OPSER Controlled Program

The BACKUP program runs as a job on line to OPSER only after it has finally detached from its terminal. When BACKUP is running detached, the operator can send commands to it through OPSER. If, after having been detached, BACKUP is later attached to a terminal again, the operator can communicate with BACKUP either at its terminal or through the OPSER program. Table 5–15 summarizes the commands that BACKUP recognizes.

Table 5–15: BACKUP Commands through OPSER

Command and Syntax	Meaning
ABO[RT]	Terminates the BACKUP run immediately and removes BACKUP from OPSER internal tables.
CON[TINUE]	Wakes up the BACKUP program to continue processing after a PAUSE command or continues processing after an operator has performed some requested action.
DET[ACH]	Valid only if BACKUP is attached, OPSER is running, the job running BACKUP has permanent privilege, the BACKUP listing file is not on the terminal to which BACKUP is attached, and no messages are currently pending for the specific BACKUP job. If these conditions are met, BACKUP detaches and is given a receiver identification of BACKnn where nn is its job number. All further interaction with BACKUP is done through OPSER.
END	Finishes transferring the current file and terminates processing.
LAS[T]	Prints, at the OSC, the most recent message generated by this BACKUP job.
LEG[AL]	Prints, at the OSC, a list of commands that can legally be given for this phase of the BACKUP run.
NOT[ICE]^text	Inserts the specified text in the BACKUP listing file. Precedes the text with the heading NOTICE FROM OPERATOR.
PAU[SE]	Places BACKUP in a sleep state, during which it responds to most commands, continues normal processing in response to a CONTINUE command, and terminates processing on an ABORT command.
STA[TUS]	Prints a status report for this BACKUP job.
The symbol ^ marks the location of a required space.	

While BACKUP is running detached, it generates all its requests for operator interaction through OPSER as messages. The operator uses the INTERRUPT command to respond to such requests. The operator must refer to Chapter 8 for the proper responses to BACKUP requests.

Chapter 6
System Error Package

Logging of hardware and system level software errors is an automatic function of the RSTS/E monitor. To gain the full advantages of this error detection capability, you must properly use the programs in the System Error Package. The programs in this package fall into three main categories:

1. Extraction and retention of error messages
2. Extraction and retention of error messages not yet retrieved at the time of a system crash
3. Compilation and formatting of saved error messages

When the monitor detects an error, routines save critical error-related data and send a message to the ERRCPY program. The system invokes ERRCPY, which then retrieves the saved data, performs minimal error message processing, and stores the error message contents in a specially formatted disk file (ERRLOG.FIL). Because the number of messages that may be queued to ERRCPY is limited, ERRCPY must be running to prevent the loss of valuable diagnostic information. A separate program, ERRINT, initializes and validates the error logging file to minimize the size of ERRCPY.

When a system crash occurs and the system crash dump facility is enabled, the monitor preserves the contents of certain critical parts of the system in the system file [0,1] CRASH.SYS. You should run the ANALYS program immediately following the recovery from a system crash to extract and format key information from the crash file. One of the functions ANALYS performs is the creation of a separate error logging file (by default ERRCRS.FIL) containing error messages that had not been processed by ERRCPY at the time of the crash and an ERRDIS report for the errors.

6.1 Use of the Error Logging Programs — ERRINT and ERRCPY

The system program ERRCPY reads error-related information stored in the monitor part of memory and writes it to a special disk file called ERRLOG.FIL. Only by running ERRINT, the error file initialization and validation program, can you use the ERRCPY program. ERRINT performs various file checking or generation functions and chains to ERRCPY. You must make sure the proper commands are in the START.CTL or CRASH.CTL files so that ERRINT (and therefore ERRCPY) is started and ERRCPY is active during time-sharing operations.

6.1.1 Error Logging Initialization — ERRINT

When the RSTS/E system starts up, the INIT system program executes commands in either the START.CTL or CRASH.CTL control file. If you include the following command line in the control file, INIT places the command RUN $ERRINT in the input buffer of terminal KB0: as if you had typed it at the terminal:

```
FORCE KB0: RUN $ERRINT
```

When the system executes the command, the ERRINT program tries to locate the error file ERRLOG.FIL. If it finds the file, ERRINT checks certain critical control information in the file. If this information is invalid, ERRINT renames the file ERRLOG.TMP, generates and initializes a new ERRLOG.FIL, and issues a message telling you what was done:

```
Error File was found to be Invalid
Error File was generated and Invalid File renamed ERRLOG.TMP
```

If ERRINT does not find ERRLOG.FIL, it generates and initializes the file.

Two other commands, corresponding to responses to ERRINT questions, must also be present in the control file. The two ERRINT dialogue questions for which you must include responses are:

```
CHANGE SIZE TO < 100 >?
UTILIZE CRASH FILE OUTPUT (YES/NO) <NO>?
```

Once ERRINT has validated or created ERRLOG.FIL, it prints a message, such as the following, telling you what percentage of the file has been used:

```
ERRLOG FILE IS 8% FULL
```

ERRINT also prints a question that includes (in angle brackets) the maximum number of blocks allowed for the file:

```
CHANGE SIZE TO < 100 >?
```

You can accept the current size of the file (originally 100 blocks) or change the maximum size if you want. The program then asks if you want to append

the contents of the special error crash file ERRCRS.FIL to the contents of ERRLOG.FIL in the question:

```
UTILIZE CRASH FILE OUTPUT (YES/NO) <NO>?
```

If you do, ERRINT tries to find the file and add its contents to the end of the main error logging file. If it cannot find the error crash file, ERRINT ignores the command and continues.

After it finishes processing all its functions, ERRINT detaches and chains to ERRCPY. At this point, the terminal on which ERRINT started is free for other use.

The ERRINT program is usually located in the system library account [1,2]. To run ERRINT from this account, type the RUN $ERRINT command, and then press the RETURN key. As an alternative, you can request that the installation procedure for the standard library programs locate the ERRINT program (and all the other programs in the System Error Package) to another account and run ERRINT from the new location. To run the ERRINT program from an account other than the system library account [1,2], you must replace the dollar sign ($) in the RUN $ERRINT command with the new account number enclosed in square brackets or parentheses; for example, RUN [1,10]ERRINT.

You can run ERRINT only if ERRCPY is not currently running. If you start up ERRINT while ERRCPY is running, then when ERRINT chains to ERRCPY, the new version of ERRCPY will hibernate and display the message:

```
?Name or account now exists at line 1025
```

However, the version of ERRCPY that was running in a detached state on the system continues to run unhampered.

In this chapter, error program examples are run from the system library account [1,2].

6.1.2 Examples of ERRINT Dialogue

The following is an example of the ERRINT dialogue:

```
$ RUN $ERRINT(RET)
ERRINT V8  RSTS V8  TIMESHARING
ERRLOG FILE IS 8% FULL
CHANGE SIZE TO < 100 >? 120(RET)
UTILIZE CRASH FILE OUTPUT (YES/NO) <NO>? NO(RET)
DETACHING
```

In this example, a valid error log file exists, has a maximum length of 100 blocks, and is 8% full. You change the maximum size to 120 blocks but decide not to add the special error log file from ANALYS to the error log file. The program then detaches and chains to ERRCPY.

In the next example, ERRINT finds an invalid error log file, renames the invalid file as ERRLOG.TMP, creates a new file, leaves the maximum size at 100 blocks, and attaches the special error log file ERRCRS.FIL to the error log file ERRLOG.FIL. Note that the 2% full message indicates a file containing only control information; this is the state of a newly generated error log file:

```
$ RUN $ERRINT(RET)
ERRINT V8  RSTS V8  TIMESHARING
ERROR FILE WAS FOUND TO BE INVALID
NEW FILE WAS GENERATED AND INVALID FILE RENAMED ERRLOG.TMP
ERRLOG FILE IS 2% FULL
CHANGE SIZE TO < 100 >?(RET)
UTILIZE CRASH FILE OUTPUT (YES/NO) <NO>? YES(RET)
DETACHING
```

6.1.3 Error Logging — ERRCPY

You can execute the ERRCPY program only by a chain from ERRINT. ERRCPY runs detached, processes incoming error messages, and writes each message as a variable-length record to a nonspanned sequential file (ERRLOG.FIL). When its message queue is empty, ERRCPY enters the sleep state and is swapped out.

The monitor queues messages to ERRCPY regardless of the presence or absence of ERRCPY. (If ERRCPY is not running, the messages are simply dropped.) The first message the monitor queues upon startup is always a "power fail/start up" message. On receiving a special error message from the SHUTUP program, ERRCPY logs the message, closes the error file, and kills itself.

6.1.4 Description of the Error File (ERRLOG.FIL)

The ERRLOG.FIL file consists of a one block header followed by a variable number of data blocks. The header block contains 64 8-byte logical records. The first 62 logical records in the header block contain counters for each of 62 possible error types. The last two logical records in the block contain general control information.

The header contains the following information for each error type:

- Number of error records logged
- Number of errors logged (including repeat counts)
- Number of errors received (including repeat counts)
- Maximum number of error records that may be logged

The general control information consists of the following six total fields:

1. Total error records logged (including repeat counts)
2. Total errors received (including repeat counts)

3. Block number for start of next error record
4. Offset within the block for next error record
5. Maximum size of error file in blocks
6. Bytes reserved for future use

The data portion of this sequential file contains variable-length records. These records are packed, but no record spans two disk blocks. Each record consists of the error message as received from the monitor preceded by a one-byte field giving the length of the record.

6.2 Displaying Errors — ERRDIS

The error display program ERRDIS provides you with a convenient method of displaying previously logged errors.

ERRDIS has four functions:

1. Provides a summary report by error type and unit number (where applicable) of all errors logged to the error file
2. Provides a detailed report of one or all error types logged between any two user-selected date and time pairs
3. Zeros the contents of the error file following the generation of a report
4. Provides a list of potentially bad disk blocks.

ERRDIS consists of two modules: ERRDIS and ERRDET. The first module performs functions 1 and 3. To perform the remaining two functions, ERRDIS chains to ERRDET, which returns control to ERRDIS when it finishes processing. Before running ERRDIS for the first time, you must run the ERRBLD program, which then creates the ERRDAT.FIL error file. Generally, this function is automatically performed when the BUILD program builds the Error Package at the time you generate your system.

```
$ RUN $ERRBLD
ERRBLD  V8 RSTS V8  TIMESHARING
```

After ERRBLD prints its header information, it builds ERRDAT.FIL and then returns control to your keyboard monitor.

You can run ERRBLD at any time, before or after running ERRINT. If ERRDAT.FIL should become corrupted, you can rerun ERRBLD with no adverse effects.

6.2.1 Running ERRDIS

You can run the ERRDIS program by typing:

```
$ RUN $ERRDIS
ERRDIS  V8 RSTS V8  TIMESHARING
```

Table 6-1: ERRDIS Dialogue Explanation

Step	Prompt and Description
1	`ERRDIS V8  RSTS V8  TIMESHARING` ERRDIS prints an identification line.
2	`INPUT FILE <[1,2]ERRLOG.FIL>?` Enter a file specification for ERRDIS to process. The file must be formatted as an error logging file.
3	`OUTPUT TO <KB:ERRDIS.OUT>?` Enter a file specification to which ERRDIS should send the report.
4	`HE[LP],BA[D BLOCKS],SU[MMARY] OR FU[LL] REPORT <SUMMARY>?` Select one of four possible reports: a. HE[LP]. Prints help file, followed by a list of mnemonics corresponding to all possible error types. b. BA[D BLOCKS]. Outputs a report of possible bad blocks detected from the disk errors logged in the input file entered in Step 2. c. SU[MMARY]. Outputs a report of the number of errors detected for each error type and for each unit number within a specific error type. The next dialogue step is 10. d. FU[LL]. Outputs a report of the detailed contents of each selected error record in the error file. Selection of a specific record depends on the answers to steps 5 through 9 in the ERRDIS dialogue.
5	`SPECIFIC ERROR TYPE <ALL>?` Enter a two-character mnemonic identifying the specific error type that you want ERRDIS to process. The default answer selects all error types. Note that if you select "ALL", you may also attach the /NOTAPE switch if you want magnetic tape errors to be omitted from the error listing.
6	`STARTING DATE <FIRST ERROR>?` Enter the date of the earliest error that you want ERRDIS to process. The format is dd-mmm-yy. The default answer is the date of the first error in the error log file. If you press RETURN to select the default, the next step is 8.
7	`STARTING TIME <FIRST ERROR>?` Enter the time of the earliest error that you want ERRDIS to process. The format is hh:mm. The default is the time of the first error in the error log file on the selected date.
8	`ENDING DATE <LAST ERROR>?` Enter the date of the latest error that you want ERRDIS to process. The format is dd-mmm-yy. The default is the date of the last error in the error log file. If you select the default, then the next dialogue step is 11.
9	`ENDING TIME <LAST ERROR>?` Enter the time of the latest error that you want ERRDIS to process. The format is hh:mm. The default is the time of the last error in the error log file on the selected ending date. The next dialogue step is 11.
10	`LIST BAD BLOCKS (YES/NO) <YES>?` Press the RETURN key (or type YES) to have ERRDIS generate a list of possible bad blocks following the summary report. Type NO if you do not want a list of bad blocks.
11	`ZERO ERROR FILE UPON COMPLETION (YES/NO) <NO>?` Type YES to have ERRDIS zero the previously specified error log file after it successfully generates the desired report. (You must type all three characters of YES to delete the file.)

After ERRDIS prints header information, you select optional modes of operation by means of an interactive dialogue. In the dialogue description shown in Table 6–1, prompts are numbered for reference. An answer enclosed in angle brackets indicates the default response; select the default by pressing the RETURN key in response to a program prompt. You can abbreviate most responses with two characters. Optional characters are enclosed in square brackets [ ], for example HE[LP].

After you finish the dialogue, ERRDIS processes the selected error records, formats and generates the selected report, zeros the error log file (if requested in step 11), and returns to step 2. At that point, typing CTRL/Z in response to any question returns the terminal to your keyboard monitor prompt.

6.2.2 Help Report

The help report prints information on the use of the ERRDIS program and then prints a list of the error mnemonics other programs use. After you run ERRDIS and select the HELP report, the program prints the HELP text as follows:

```
$RUN [1,10]ERRDIS<RET>
ERRDIS V8  RSTS V8   TIMESHARING
Input File <[1,10]ERRLOG.FIL>?<RET>
Output to <KB:ERRDIS.OUT>?<RET>
He[lp], Ba[d Blocks], Su[mmary] or Fu[ll] Report <Summary>? HELP<RET>

ERRDIS  System Error Display program

ERRDIS questions and required user input are  described  below.  Typing
carriage return (<cr>) specifies the default.

Question                  Default        Legal response(s)

A. Input file?            ERRLOG.FIL in the Error Package library account
                                         ERRDIS readable error log file

B. Output file?           KB:ERRDIS.OUT  File spec or '?' for calculated

C. He[lp], ba[d blocks], su[mmary] or fu[ll] report?

                          Summary        Minimally, either HE, BA, SU  or
                                         FU

     1. HE[LP]  -  list  this file and a list of legal 2 character error
        types (mnemonics) and their meanings.

     2. BA[D BLOCK] - print only a list of possible bad blocks.

     3. SU[MMARY] - print a summary of all errors recorded

     4. FU[LL] - full report

D. Specific error type?   All              'AL[L]' errors  or  one  of  the
                                           legal 2 character mnemonics.
                                           '[ALL]/NOT[APE]' requests all but
                                           magnetic tape errors.

E. Starting date?         First error (date of first recorded error)
                                         Date in the form dd-mmm-yy.
```

(continued on next page)

```
F. Starting time?          First error (on or after the starting date)
                                   Time in the form hh:mm.

G. Ending date?            Last error (date of last recorded error)
                                   Date in the form dd-mmm-yy.

H. Ending time?            Last error (on or before the ending date)
                                   Time in the form hh:mm.

I. List bad blocks?        Yes             Ye[s] or No

J. Zero error file upon completion?
                           No              Yes (3 chars required) or No.
```

*****NOTE:

1. The user may exit from ERRDIS by typing CTRL/Z in response to any question or by typing CTRL/C while ERRDIS is running.

2. Typing CTRL/C while ERRDET (which produces detailed reports and lists of possible bad blocks) is running will stop report generation, cancel the request for zeroing of the input file if it was specified, return control to ERRDIS and cause the interactive session to restart.

3. Completion of all user requests is signalled by the restart of the interactive session.

4. ERRDAT.FIL (the error descriptor file) must exist in the account on which the Error Package is stored in order for ERRDIS to be able to run.

MNEMONIC	DESCRIPTION
MS	Missed Errors
CK	RTS Declared
JO	Jump to 0
UI	Undefined Inter.
T4	Trap to 4
RI	Reserved Instr.
PF	PowerFail/Strtup
PA	Memory System
KT	Memory Mgmt.
DF	RF11/RS11
DS	RS03-04
DK	RK11/RK05/RK05F
DM	RK611/RK06-07
DP	RP11/RP02-03
DR	RM02/3/5/80
DB	RP04-05-06
DL	RL01/RL02
DU	MSCP Disks
KB	Terminals
DT	DECtape
LP	Line Printers
PR	Paper Tape Rdr.
PP	Paper Tape Punch
CR	CR11/CM11
CD	CD11 Card Reader
MT	TM11/TU10/TE10
MM	TU16/TE16
DX	RX11/RX211

(continued on next page)

```
XM              DMC11/DMR11
RJ              RJ2780
XY              X-Y Plotter
NW              DECNET
TS              TS11/TSV05/TU80
DD              TU58
XD              DMV11/DMP11
UM              Unrecognized MSG
SH              MSG from SHUTUP
```

6.2.3 Summary Report

A summary report supplies general information on all errors logged. The first two columns list the two-character error mnemonic and full error description. The column titled TOTAL REC/LOG lists the total errors received by the ERRCPY program and the total it stored in the error file. These two totals may differ due to the limits on:

- The number of errors that may be logged
- The size of the error file

The columns titled UNIT NUMBERS are the total errors generated by each device on a controller.

The last column, "CONTROLLER ERRORS", indicates how many of the errors logged under TOTAL REC/LOG are controller errors, as opposed to unit errors. ("Controller errors" apply to the controller itself, while "unit errors" apply to a specific unit being controlled.) If N/A appears under CONTROLLER ERRORS, that means that ERRDIS is not keeping track of CONTROLLER ERRORS for that particular ERROR CODE-DESCRIPTION.

The "*" next to 239/100 indicates that MM, under the ERROR CODE-DESCRIPTION column, has exceeded the maximum number of errors (100) that the Error Package will log for that particular error type.

After you run ERRDIS and select the SUMMARY report, the program prints the summary as follows. Note that the Summary Report now contains fewer blank lines, to compress the display.

```
$ RUN [1,229]ERRDIS (RET)
ERRDIS  V8      RSTS    BURNSY
Input File <[1,229]ERRLOG.FIL>? (RET)
Output to <KB:ERRDIS.OUT>? (RET)
He[lp], Ba[d Blocks], Su[mmary] or Fu[ll] Report <Summary>? (RET)
List Bad Blocks (Yes/No) <Yes>? NO (RET)
Zero Error File upon completion (Yes/No) <No>? (RET)
  ERRDIS Summary Report taken on 29-Oct-82, 10:50 AM
     Input File: (1,229)ERRLOG.FIL  Output File: KB:ERRDIS.OUT
     Reported Date/Time Range:
        13-DEC-81, 12:21:22 AM through 28-Oct-82, 12:35:47 AM
```

(continued on next page)

```
      ERROR                TOTAL                UNIT NUMBERS                CONTROLLER
CODE-DESCRIPTION         REC/LOG     0    1    2    3    4    5    6    7      ERRORS

PF PowerFail/Strtup        1/1                                                   N/A
DU MSCP Disks              7/7           1              2         1            3
DB RH11/RP04-05-06         3/3                3                                N/A
DL RL01/RL02               3/3       1        2                                N/A
MM RH11/TU16/TE16    * 239/100      49   7         5     2        37           N/A
SH MSG from SHUTUP         1/1                                                   N/A

Total of 115 Errors Logged out of 254 Received
34 out of 100 Blocks have been used in (1,229)ERRLOG.FIL

*******************************************************************************
Input File <[1,229]ERRLOG.FIL>? ^Z

$
```

6.2.4 Bad Block Report

The ERRDET program reports possible bad blocks that were detected by the monitor by checking the error logging file for such entries. The criteria for a possible bad block depends on the type of disk.

Each bad block report includes the logical block number, the logical name or pack identification of the disk, and the physical device name of the disk. Using the logical block number that ERRDIS prints, you can add the possible bad block to the bad block file with the BADS suboption of REFRESH. Refer to the *RSTS/E System Generation Manual* for information on adding bad blocks.

The following example shows a list of bad blocks that the ERRDIS program prints:

```
$ RUN [1,229]ERRDIS (RET)
ERRDIS  V8       RSTS     BURNSY
Input File <[1,229]ERRLOG.FIL>? (RET)
Output to <KB:ERRDIS.OUT>? (RET)
He[lp], Ba[d Blocks], Su[mmary] or Fu[ll] Report <Summary>? BAD BLOCKS (RET)
Zero Error File upon completion (Yes/No) <No>? (RET)
  ERRDIS Bad Block Report taken on 29-Oct-82, 10:51 AM
     Input File: (1,229)ERRLOG.FIL  Output File: KB:ERRDIS.OUT
     Requested Date/Time Range:
        First Error through Last Error
     (1,229)ERRLOG.FIL will not be Zeroed upon completion

List of Possible Bad Blocks:
----------------------------
Logical Name             RAIDER
Physical Name            DM1:
Logical Block Number     3401

Logical Name             OFTHE
Physical Name            DR5:
Logical Block Number     237968

Logical Name             LOST
Physical Name            DR5:
Logical Block Number     432952
```

(continued on next page)

```
Logical Name              ARK
Physical Name             DM2:
Logical Block Number      8

Total Number of Possible Bad Blocks: 4

*******************************************************************************
Input File <[1,229]ERRLOG.FIL>? ^Z
```

NOTE

Use caution before adding possible bad blocks (that ERRDIS reports) to the bad block file. Hardware controller or disk drive problems may have caused the errors to occur rather than corrupt disk surfaces.

6.2.5 Adding Bad Blocks to the Bad Block File

The ERRDIS program prints a list of potential bad blocks. If you decide to add a block to the bad block file BADB.SYS on a disk, use the BADS suboption of the REFRESH option in INIT.SYS, described in the *RSTS/E System Generation Manual*. In preparation for adding bad blocks, which requires you to bring the system down, use the PIP system program to copy the data in a bad file to a new file. This allows you, as your system discovers bad blocks, to collect this information in a single file where you can, when you normally take your system down, add the bad blocks to BADB.SYS. Follow these instructions to use PIP for this purpose:

1. Specify the /GO switch (along with other necessary switches, such as /CL, and so forth) to ignore possible ?DATA ERROR ERROR ON DEVICE errors. Rename the file that contains the bad block, but do not delete it:

```
$ RUN $PIP
*FILE.TMP=FILE.OLD/GO
*FILE.BAD=FILE.OLD/RE
*FILE.OLD=FILE.TMP/RE
```

2. When it is convenient, shut down the system and use the BADS suboption of REFRESH to add any bad blocks to the disk's bad block file. Each new bad block is now allocated to two files ([0,1]BADB.SYS and FILE.BAD in the above example).

3. The CLEAN suboption automatically rebuilds the allocation tables. It then tells you the file has a bad block and allows you to delete the file (FILE.BAD in the above example). The *RSTS/E System Generation Manual* describes this operation.

When CLEAN deletes the file, it frees the blocks in the file that are not bad.

You must slightly modify the above procedure if the bad block is in a file directory. In this case, use the BACKUP or SAVE/RESTORE program to copy all accessible files in the account (or disk) to an archive medium. When you add the bad block and clean the disk, the CLEAN suboption lets you delete the account whose directory contains the bad block. After starting timesharing, use SAVE/RESTORE to restore the entire disk, or use REACT to enter the account on the disk and BACKUP to restore the files from the archive medium.

To determine bad blocks for an MSCP type disk (such as an RA80 or RA60), run ERRDIS and request the bad block report.

6.2.6 Full Report

Each error displayed includes a title line containing the error code mnemonic, the error description, a sequence number, and the date and time the error occurred. The monitor maintains the sequence numbers from start-up to shutdown. A crash and subsequent automatic restart does not reset these numbers. The sequence numbers start at 1.

There are four basic categories for individual errors displayed within the full error report:

1. Nonperipheral errors such as processor traps and memory parity
2. Disk errors
3. Nondisk peripheral device errors such as magnetic tape errors
4. Other errors such as the shutdown error received from SHUTUP or missed errors

The program automatically checks for and reports possible bad blocks for disk errors included in the scan. The following sections contain partial listings of a FULL report.

When the monitor detects a hardware or software error, it sends information about the error to the ERRCPY program. ERRCPY can accept and file error information until the number of queued errors reaches 39. All errors the monitor detects beyond this are logged as "missed errors." Only after the queue begins to empty and the number of errors queued to ERRCPY falls below the maximum of 39 can the monitor begin to send the normal error information. This means that while the queue is full, the monitor reports all subsequent errors as "missed errors," rather than sending the usual error information to ERRCPY.

The monitor also reports missed errors if it does not have a sufficient number of general small buffers available to store error information. When the number of general small buffers drops below 75, the monitor reports any detected error to ERRCPY as a missed error.

6.2.6.1 User Description in Full Report – All errors except shutdown, missed errors, and MSCP controller errors usually include a user description. The name fields in the user description relate to a specific job whose number

appears in the first field (Job number). Under certain conditions, a job number of 0 is possible. In this case, the user description is not listed. The following example shows a complete error report for a tape device; the report includes both a user description and a detailed description:

```
$ RUN [1,229]ERRDIS (RET)
ERRDIS  V8       RSTS    BURNSY
Input File <[1,229]ERRLOG.FIL>? (RET)
Output to <KB:ERRDIS.OUT>? (RET)
He[lp], Ba[d Blocks], Su[mmary] or Fu[ll] Report <Summary>? FULL REPORT (RET)
Specific Error Type <All>? (RET)
Starting Date <First Error>? 13-Dec-81 (RET)
Ending Date <Last Error>? (RET)
Zero Error File upon completion (Yes/No) <No>? (RET)
  ERRDIS Full Report (All Types) taken on 29-Oct-82, 02:17 PM
     Input File: (1,229)ERRLOG.FIL          Output File: KB:ERRDIS.OUT
     Requested Date/Time Range:
        13-DEC-81 (First Error) through Last Error
     (1,229)ERRLOG.FIL will not be Zeroed upon completion

*************************************************************************
MM  RH11/TU16/TE16 Seq  92 Occurred on 28-Oct-82 at 02:11:44 PM

User Description:
-----------------
Job Number                  18
KB Number                   2
Account                     [1,170]
Program Name                PASIKO
User Job Physical Addr.     02310000
User Job Size               28K
Control Parameters          200
RTS Name                    RT11
RTS Physical Address        01714000

Detailed Description:
---------------------
Timeout Indicator           000
Physical Name               MM0:

DDB                         017600  173444  104252  000011
                            045471  005171  001630  177400
                            000000  147240  000000  001007
                            000056  002004  127311

CSR Address:                176700
------------

Contents of Registers:
----------------------
MTCS1       145270          TRE Xfer Err
MTWC        000400
MTBA        132444
MTFC        000000
MTCS2       000100
MTDS        150660          PE Mode
MTER        100000          COR/CRC
MTAS        000001
MTCK        000000
MTMR        000000
MTDT        000000          TU77            TM03
MTSN        000000
MTTC        000000
MTBAE       000016
MTCS3       002000
```

Table 6–2 summarizes the user description data.

Table 6–2: User Description Data

Heading	Meaning
Job Number	For disk errors, the number of the job that requested the I/O, for nondisk peripheral errors, the job owning the device, for nonperipheral errors, the job running at the time of the error.
KB Number	The keyboard number of the job, followed by DET if the job is detached.
Account	The project-programmer number of the user who creates the job.
Program Name	The name of the program running in the job virtual address space (low segment).
User Job Physical Addr	The current physical memory address of the job.
User Job Size	The size of the user program running in the low segment of the job virtual address space.
Control Parameters*	Monitor control information.
RTS Name	The name of the run-time system mapped into the highest segment of the user's virtual address space.
RTS Physical Address	The address of the location in physical memory where the run-time system resides.
* A detailed description of this entry is beyond the scope of this manual. The entry is intended for use by DIGITAL maintenance personnel.	

6.2.6.2 Disk Error Detailed Description – A FULL report for a disk error includes three section:

1. User description
2. Detailed description
3. Contents of register

Table 6–3 describes the information contained in the detailed description portion of a disk error report. Before going to the table, study the following example. The example shows what a single complete disk error looks like. If the entire Full Report of DB type errors were reproduced here, you would see two more identically structured errors messages following this one. As you can tell from the sample Summary Report in Section 6.2.3, there are 3 DB errors under TOTAL REC/LOG.

```
$ RUN [1,229]ERRDIS (RET)
ERRDIS  V8       RSTS     BURNSY
Input File <[1,229]ERRLOG.FIL>? (RET)
Output to <KB:ERRDIS.OUT>? (RET)
He[lp], Ba[d Blocks], Su[mmary] or Fu[ll] Report <Summary>? FULL REPORT (RET)
Specific Error Type <All>? DB (RET)
Starting Date <First Error>? 03-SEP-82 (RET)
```

(continued on next page)

```
Starting Time <First Error>? RET
Ending Date <Last Error>? RET
Zero Error File upon completion (Yes/No) <No>? RET
  ERRDIS Full Report (DB only) taken on 29-Oct-82, 11:00 AM
     Input File: (1,229)ERRLOG.FIL          Output File: KB:ERRDIS.OUT
     Requested Date/Time Range:
        03-SEP-82 (First Error) through Last Error
     (1,229)ERRLOG.FIL will not be Zeroed upon completion

********************************************************************************
DB  RH11/RP04-05-06 Seq  81 Occurred on 28-Oct-82 at 07:12:14 AM

User Description:
-----------------
Job Number                  17
KB Number                   33 (Det)
Account                     [1,100]
Program Name                BACKTO
User Job Physical Addr.     02744000
User Job Size               23K
Control Parameters          200
RTS Name                    ...RSX
RTS Physical Address        00000000

Detailed Description:
---------------------
I/O Status                  000
Timeout Indicator           000
Offset Position             Previous
Overlapped Seek Ind.        377
Unit Size in DC's           121510
Device Cluster Size         8
Pack Cluster Size           8
Logical Name                D
Physical Name               DB2:
Logical Block Number        19272

DSQ                         004500  173442  104252  000011
                            045501  006171  064540  174000
                            000000  146300  000000  001000
                            000056  002004  127310  001003

WCB                         100000  004442  000002  000021
                            146274  000000  000000  001310
                            113570  004550  004551  005165
                            005166  005413  005414  005415

FCB                         147040  070560  125064  006274  [170,52]
                            034354  006273  037404  000001  BATIDL.BAS
                            001300  113570  001270  113570
                            000011  000226  000010  146300

CSR Address:                176700
------------

Contents of Registers:
----------------------
RPCS1       145270          TRE Xfer Err
RPWC        000400
RPBA        000000
RPDA        000000
RPCS2       000000
RPDS        000000          Not Ready
RPER1       000000
RPAS        000000
```

(continued on next page)

```
RPLA      000000
RPMR      000000
RPDT      000000
RPSN      000000
RPOF      000000
RPDC      000000
RPCC      000000
RPER2     000000
RPER3     000000
RPEC1     000000
RPEC2     000000
```

Table 6–3 contains information about the detailed description portion of a disk errors report. The program prints an additional field if the error signaled a possible bad block.

Table 6–3: Disk Error Detailed Description

Heading	Meaning
I/O Status	See Timeout Indicator below.
Timeout Indicator	Timeout and I/O Status are considered as a pair. One of the following states can exist: Timeout Indicator / I/O Status / Meaning 0 / 0 / Idle x / 1 / SEEK in progress 0 / 1 / Waiting for Read/Write 1 / x / Timeout Where x = any nonzero value
Offset Position	The contents of Offset Position for disks having offset capabilities follows: 0 = centerline position 20 = +x offset 220 = −x offset 40 = +2x offset 240 = −2x offset 60 = +3x offset 260 = −3x offset 100 = offset unknown Where x = the number of micro inches of offset (a drive type parameter).
Overlapped Seek Ind.	377 = overlap seek 000 = non-overlap seek The system manager determines which of these two values appears at system generation.
Unit Size in DC's	Size (in octal) of the disk expressed as the number of device clusters.
Device Cluster Size	Device cluster size for this drive.
Pack Cluster Size	Pack cluster size for this disk.

(continued on next page)

Table 6–3: Disk Error Detailed Description (Cont.)

Heading	Meaning
Logical Name	The pack identification if no system wide logical name was given at mount time or the system logical name entered at mount time.
Physical Name	The physical name and unit number.
Logical Block Number	The block number where the error occurred.
DSQ*	The disk request queue entry block.
WCB*	The WCB (Window Control Block) is present for disk errors that resulted from a file request by the user job.
FCB*	The File Control Block.
CSR Address	The CSR address is the base address of the set of registers in the I/O page belonging to the device that caused the error.
Contents of Registers	This section contains the name of each readable device register, its contents and, possibly, an abbreviated description of various error bits that were found to be set to one in the register. A maximum of 21 error bits are stored for each error type. The register mnemonics, error bit descriptions, and error code descriptions for all error types are contained in a data file ERRDAT.FIL. ERRDIS requires this file in order to run. A detailed description of registers and bit descriptions may be found in the *PDP–11 Peripherals Handbook*. The register mnemonics displayed by ERRDIS correspond to those given in the manual.

* A detailed description of this entry is beyond the scope of this manual. The entry is intended for use by DIGITAL maintenance personnel.

6.2.6.3 MSCP Variations on the Full Report – For systems that support MSCP type disk hardware, (for example, the RA80), there are two variations on the Full Report:

1. If there is a disk error, then the Full Report looks just like the normal Full Report, except that it has an extra section called the MSCP Description. Therefore, it includes these four sections:
    1. User Description
    2. Detailed Description
    3. Contents of Registers
    4. MSCP Description
2. If there is a controller error, then the Full Report shows an abbreviated error message. The report does not have a User Description, and its Detailed Description is not as large as in the normal Full Report. The report includes these three sections:
    1. Detailed Description
    2. Contents of Registers
    3. MSCP Description

If you refer back to the Summary Report example in Section 6.2.3, you will notice that the first example that follows is the error that was logged under UNIT NUMBER 1. The second sample error message that follows is one of the 3 errors logged under CONTROLLER ERRORS.

Example #1:

```
$ RUN [1,229]ERRDIS (RET)
ERRDIS  V8       RSTS     BURNSY
Input File <[1,229]ERRLOG.FIL>? (RET)
Output to <KB:ERRDIS.OUT>? (RET)
He[lp], Ba[d Blocks], Su[mmary] or Fu[ll] Report <Summary>? FULL REPORT (RET)
Specific Error Type <All>? DU (RET)
Starting Date <First Error>? (RET)
Ending Date <Last Error>? (RET)
Zero Error File upon completion (Yes/No) <No>? (RET)
  ERRDIS Full Report (DU only) taken on 29-Oct-82, 02:17 PM
     Input File: (1,229)ERRLOG.FIL          Output File: KB:ERRDIS.OUT
     Requested Date/Time Range:
        First Error through Last Error
     (1,229)ERRLOG.FIL will not be Zeroed upon completion

*******************************************************************************
DU  MSCP Disks  Seq  94 Occurred on 28-Oct-82 at 02:35:39 PM

User Description:
-----------------
Job Number                  12
KB Number                   30
Account                     [1,229]
Program Name                MAIL
User Job Physical Addr.     02654000
User Job Size               24K
Control Parameters          204
RTS Name                    ...RSX
RTS Physical Address        00000000

Detailed Description:
---------------------
I/O Status                  000
Timeout Indicator           377
Offset Position             000
Overlapped Seek Ind.        000
Unit Size in DC's           163646
Device Cluster Size         4
Pack Cluster Size           4
Logical Name                SYSLIB
Physical Name               DU1:
Logical Block Number        23832

DSQ                         000000  177430  106130  000014
                            056425  005571  054000  060000
                            177404  007160  000000  101234
                            051117  002005  000000  000403

CSR Address:                172150
------------

Contents of Register:
---------------------
SA        000000            Controller On Line
```

(continued on next page)

```
MSCP Description:
-----------------
MSCP Envelope              000070  000020

MSCP Packet                021640  140234  000001  000000
                           040403  000353  000000  000000
                           000000  000402  000402  011203
                           000077  000000  000000  001001
                           000012  000000  013722  000030
                           055327  000000  002023  000000
                           000000  000000  000000  000000
                           000000  000000

Status Code of Packet      Drive Error
MSCP Intnl Ctrl Sttus Wd   000300
MSCP Intnl Unit Sttus Wd   100004
MSCP Error Code            000006
BBR Flag Word              000000
LBN Being Replaced         000000  000000
Replacement Block Number   000000  000000
RBN Being Replaced         000000  000000
```

Example #2:

```
$ RUN [1,229]ERRDIS (RET)
ERRDIS  V8       RSTS     BURNSY
Input File <[1,229]ERRLOG.FIL>? (RET)
Output to <KB:ERRDIS.OUT>? (RET)
He[lp], Ba[d Blocks], Su[mmary] or Fu[ll] Report <Summary>? FULL REPORT (RET)
Specific Error Type <All>? DU (RET)
Starting Date <First Error>? (RET)
Ending Date <Last Error>? (RET)
Zero Error File upon completion (Yes/No) <No>? (RET)
  ERRDIS Full Report (DU only) taken on 29-Oct-82, 02:19 PM
     Input File: (1,229)ERRLOG.FIL        Output File: KB:ERRDIS.OUT
     Requested Date/Time Range:
        First Error through Last Error
     (1,229)ERRLOG.FIL will not be Zeroed upon completion

*******************************************************************************
DU  MSCP Disks  Seq  2 Occurred on 19-Apr-82 at 12:21:22 AM

Detailed Description:
---------------------
I/O Status                 000
Timeout Indicator          377

CSR Address:               172150
------------

Contents of Register:
---------------------
SA        000000           Controller On Line

MSCP Description:
-----------------
MSCP Envelope              000030  000020

MSCP Packet                000000  000000  000000  000000
                           000400  000012  044101  100513
                           000000  000402  040401  000000
                           000000  000000  000000  000000
                           000000  000000  000000  000000
                           000000  000000  000000  000000
                           000000  000000  000000  000000
                           000000  000000
```

(continued on next page)

```
Status Code of Packet     Controller Error
MSCP Intnl Ctrl Sttus Wd 000102
MSCP Intnl Unit Sttus Wd 000001
MSCP Error Code          000000
BBR Flag Word            000000
LBN Being Replaced       000000  000000
Replacement Block Number 000000  000000
RBN Being Replaced       000000  000000

$
```

6.2.6.4 Nondisk Peripheral Device Error Detailed Description — Table 6–4 lists the fields in the detailed description portion of the nondisk peripheral device error report. Before going to the table, study the example that follows. The example shows a single complete tape error. As you can see from the Summary Report example in Section 6.2.3, this is one of the 239 tape errors that were received. This particular error was one of the 100 logged, and also one of the 7 logged under UNIT NUMBER 1. If the entire Full Report were reproduced here, it would contain 99 more tape error messages.

```
$ RUN [1,229]ERRDIS (RET)
ERRDIS  V8      RSTS     BURNSY
Input File <[1,229]ERRLOG.FIL>? (RET)
Output to <KB:ERRDIS.OUT>? (RET)
He[lp], Ba[d Blocks], Su[mmary] or Fu[ll] Report <Summary>? FULL REPORT (RET)
Specific Error Type <All>? MM (RET)
Starting Date <First Error>? (RET)
Ending Date <Last Error>? (RET)
Zero Error File upon completion (Yes/No) <No>? (RET)
  ERRDIS Full Report (MM only) taken on 29-Oct-82, 02:17 PM
     Input File: (1,229)ERRLOG.FIL        Output File: KB:ERRDIS.OUT
     Requested Date/Time Range:
        First Error through Last Error
     (1,229)ERRLOG.FIL will not be Zeroed upon completion

********************************************************************************
MM  RH11/TU16/TE16 Seq  44 Occurred on 28-Oct-82 at 05:51:08 AM

User Description:
-----------------
Job Number                11
KB Number                 33 (Det)
Account                   [1,100]
Program Name              BACKTO
User Job Physical Addr.   02610000
User Job Size             23K
Control Parameters        200
RTS Name                  ...RSX
RTS Physical Address      00000000

Detailed Description:
---------------------
Timeout Indicator         000000
Physical Name             MM1:

DDB                       000016  000426  002102  100001
                          000001  000000  000000  000000
                          105536  120540  010004  000021
                          000361  001002  010760  000053
                          000000  007314  004400

CSR Address:              172440
------------
```

(continued on next page)

```
Contents of Registers:
----------------------
MTCS1       145260          TRE Xfer Err
MTWC        000000
MTBA        010004
MTFC        000000
MTCS2       000100
MTDS        150760          PE Mode
MTER        100000          COR/CRC
MTAS        000001
MTCK        000010
MTMR        001100
MTDT        142011          TU16/TE16
MTSN        060023
MTTC        102301
```

Table 6–4 describes the information found in the detailed description portion of the previous example.

Table 6–4: Nondisk Peripheral Device Format

Heading	Meaning
Timeout Indicator	Nonzero indicates that an expected response was not received from the device in the allotted time.
Physical Name	The physical name and unit number of the device.
DDB*	The DDB (Device Data Block) contains various parameters required by the system to control the device.
DDB* Extension	Supplementary DDB information.
CSR Address	See description under disk format.
Contents of Registers	See description under disk format.

* A detailed description of this entry is beyond the scope of this manual. The entry is intended for use by DIGITAL maintenance personnel.

6.2.6.5 Nonperipheral Error Detailed Description – Detailed descriptions of nonperipheral error fields are available from the related processor handbook. Table 6–5 gives an abbreviated description of these items. Before going to the table, study the example that follows. The example shows a single complete error message extracted from the ERRLOG.FIL using ERRDIS to reproduce only the PF type errors.

```
$ RUN [1,229]ERRDIS (RET)
ERRDIS  V8       RSTS     BURNSY
Input File <[1,229]ERRLOG.FIL>? (RET)
Output to <KB:ERRDIS.OUT>? (RET)
He[lp], Ba[d Blocks], Su[mmary] or Fu[ll] Report <Summary>? FULL REPORT (RET)
Specific Error Type <All>? PF (RET)
Starting Date <First Error>? (RET)
Ending Date <Last Error>? (RET)
Zero Error File upon completion (Yes/No) <No>? (RET)
  ERRDIS Full Report (PF only) taken on 29-Oct-82, 02:21 PM
     Input File: (1,229)ERRLOG.FIL          Output File: KB:ERRDIS.OUT
     Requested Date/Time Range:
        First Error through Last Error
     (1,229)ERRLOG.FIL will not be Zeroed upon completion

*******************************************************************************
```

(continued on next page)

```
PF  PowerFail/Strtup Seq  1 Occurred on 13-Dec-81 at 02:15:01 PM

Detailed Description:
---------------------
R0                          000000
R1                          004200
R2                          000000
R3                          000002
R4                          004100
R5                          000000
Virtual PC                  000077
Physical PC                00000077
Processor Status            000000
Stack Pointer               002074
(SP)                        000002
(SP+2)                      000000
(PC-6)                      034156
(PC-4)                      004356
(PC-2)                      034156
(PC)                        004357
CPU ID                          -1
CPU ERR                     000000
```

The following table describes the information contained in the previous example.

Table 6–5: Nonperipheral Error Format

Heading	Meaning
R0-R5	Processor registers 0 through 5. Each register is listed on a separate line.
Virtual PC	The address (within the user virtual address space) of the instruction being executed at the time of error.
Physical PC	As above except the physical memory address of the instruction being executed at the time of the error.
Processor Status	The processor status word.
Stack Pointer	The address of the current top of stack.
(SP) (SP + 2)	The contents of the top two words in the stack.
(PC–6) (PC–4) (PC–2) (PC)	The contents of the three words preceding the current program PC and the contents of the current PC.
CPU ID	If it exists, the contents of the CPU identification register.
CPU ERR	If it exists, the contents of the CPU error register.
MED X	If it exists, the CPU has Maintenance Examine/Deposit Instruction.
Contents of Registers	Certain nonperipheral errors include contents of key registers. The format is identical to the contents of registers for disk and nondisk peripheral errors.

NOTE

The PDP–11 computer always executes start-up by interrupting through a vector address. The same vector is used for normal start-up and for a powerfail recovery. The error log for a normal start-up reflects a sequence number of 1 and the contents of R0-R5 are all zeroes. This is not the case for a powerfail restart.

6.3 Analyzing System Crashes — ANALYS

Time-sharing operations halt when a crash occurs on a RSTS/E system. If you enabled the crash dump facility at start-up time, the system writes an image of read/write memory, tables, and XBUF to the CRASH.SYS file in account [0,1]. The system bootstraps the system disk, loads the initialization code into memory, and executes an automatic restart when the CPU switch register has bit number 0 set.

Unless you run the ANALYS program to save crash dump information, the next system crash causes the CRASH.SYS file to be overwritten and the information in the file to be lost. Before using ANALYS to document system crashes, you must have created the CRASH.SYS file at REFRESH time and have the crash dump feature enabled at system start-up. Refer to Chapter 2 for a description of automatic system recovery procedures.

The crash analysis program ANALYS consists of four modules:

- ANALYS
- ANALY1
- ANALY2
- ANALY3

For simplicity, the documentation refers to only one program, ANALYS, which functionally includes the four modules.

6.3.1 Running the ANALYS Program

Run the ANALYS system program by typing the command:

```
$ RUN $ANALYS(RET)
```

After you press the RETURN key, the ANALYS program prints an identification line and three dialogue questions:

```
ANALYS V8  RSTS V8   TIMESHARING
INPUT <[0,1]CRASH.SYS>? (RET)
OUTPUT <ANALYS.DMP>? (RET)
CRASH ERROR LOG FILENAME<[1,2]ERRCRS.FIL>? (RET)

$
```

Table 6–6 explains how to answer the ANALYS program questions.

Table 6–6: ANALYS Program Dialogue

Question and Response
`INPUT <[0,1]CRASH.SYS?` Asks for the name of the file to be analyzed. Type the file specification of the file you want ANALYS to analyze, which by default is CRASH.SYS in account [0,1]. Press the RETURN key to accept the default response. However, you may wish to keep copies of different crash files. ANALYS uses the currently installed monitor SIL file to extract symbolic references. If the crash file being analyzed is not associated with the currently installed monitor, you must append the /SIL: switch with the name of the related monitor SIL. This switch allows DIGITAL to analyze crash and .SIL files submitted with an SPR.
`OUTPUT ANALYS.DMP?` Requests a disk file or a device designator for the output medium. You can select the default output file ANALYS.DMP by pressing the RETURN key. Enter another file specification if you do not want ANALYS to place the output in the default account. ANALYS always prints an annotated version of a memory dump and automatically prints a memory dump in 132-column format unless you: 1. Request output to your terminal (KB:ANALYS.DMP, for example). 2. Append the /NARROW switch to a file specification (or type /NARROW(RET) in response to the OUTPUT question if you want ANALYS.DMP to be the output file). In either case, you get the output in 80-column format, also annotated but with fewer memory locations printed on each line. If you do not want a dump of memory, append the /NODUMP switch to your response.
`CRASH ERROR LOG FILENAME<[1,2]ERRCRS.FIL>?` Asks for the file specification of the file you want to hold certain error information from the crash file. Press the RETURN key to have ANALYS write this information to the ERRCRS.FIL in the system library account [1,2]. ANALYS retrieves error information saved at the time of a system crash but not written to the system error logging file ERRLOG.FIL. The ERRDIS program accepts the file you specify as input and produces an error log report from its contents. To retain a single continuous error logging file, use an ERRINT option.

Normally, ANALYS takes at least 15 minutes to run. When the output stops, the program automatically terminates and returns to your keyboard monitor prompt.

6.3.2 ANALYS Output

Output of the ANALYS system program supplies valuable hardware and software information that a software specialist can use to determine possible causes of system crashes. It includes:

- A report similar to SYSTAT
- A memory dump of the critical contents of memory
- A listing of all monitor symbols

The report also contains DECnet/E information if you configured your system with DECnet/E. Refer to the *RSTS/E System User's Guide* for an explanation of SYSTAT.

ANALYS reports an error code in the crash dump data. Table 6–7 shows these error codes.

Table 6-7: System Crash Error Code

Error Code (octal)	Meaning
–1(177777)	Power fail error.
–2(177776)	Jump to 0.
–3(177775)	This code is returned when RSTS/E does a crash dump as a result of halting the CPU and continuing from location 56 (octal).
41	Trap to 4.
42	Trap to 10.
43	Trap to 250 (Memory management violation).
44	Kernel SP Stack overflow.
46	Trap to 114 (Parity memory error).
0 or other	Forced dump.

Error logging printouts supplement the ANALYS output. You should compare items in the data labeled VIRTUAL PROGRAM COUNTER and PROCESSOR STATUS in the ANALYS printout with the data labeled Virtual PC and Processor Status in the ERRDIS report. If the values match, it is the error in the ERRDIS report that caused the crash.

The PROCESSOR STATUS data show the current and previous modes of the processor. RSTS/E does not use supervisor mode; the only valid modes are kernel and user. If bit 11 of the PSW is 1, the processor has two register sets; if bit 11 is 0, the processor has only one register set.

To obtain crash information printouts automatically, include the proper commands in the CRASH.CTL file. The commands run ANALYS to preserve the crash information and run ERRDIS to create a report. Refer to Chapter 3 for a description of the commands you should include in the CRASH.CTL file.

6.4 Octal Debugging Tool — ODT

The system program ODT opens a file, a peripheral device, or memory as an address space and allows you to examine and change word or byte locations within the address space. You can also list the contents of certain table locations in the operating system.

The program immediately interprets and executes each character as you type it. This is called ODT submode or delimiterless character input mode and differs from the procedure other system programs use. Other programs

interpret input only after you enter an entire line of characters. Because ODT performs processing based upon single characters typed at the terminal, its language is highly interpretive and interactive.

NOTE

ODT was designed for expert system programmers with thorough knowledge of file formats and operating system tables. Use extreme care with ODT; mistakes can have drastic results.

The program accesses and manipulates data in word and byte locations based on octal values. The word is the 16-bit PDP–11 word and can have a value between 0 (octal) and 177777 (octal), the limit imposed by 16 bits. A word has a high order (odd address) and low order (even address) byte. A byte can have a value between 0 (octal) and 377 (octal) – the limit that can be represented by 8 bits. For clarity, the following symbols express the octal values that ODT prints:

Symbol	Meaning
n	Represents an octal integer between 0 and 17. The use of 8 or 9 generates an error.
k	Represents an octal value up to 6 digits in length. If more than six digits are specified or a value greater than 177777 (octal) is specified, ODT truncates the value to the low order (rightmost) 16 bits. If a minus sign (-) precedes the octal value, ODT uses the two's complement value of the number.

For example, ODT interprets the following values as shown:

Value	Interpretation
1	000001
–1	177777 (two's complement)
400	000400
-177730	000050 (two's complement)
1234567	034567 (truncated to low-order 16-bits)

You can represent a location within the permissible address space by typing an octal value or an expression that reduces to an octal value. The correct forms and their interpretations are:

Symbol	Meaning
k	The six-digit octal value of k.
n,k	The resultant address is the value of k added to the contents of the relocation register specified by n. Relocation registers are numbered from 0 to 17 (octal). See Section 6.4.4 for more information about relocation registers.

The special characters and symbols in Table 6–8 are recognized by RSTS/E ODT and explained in the remainder of the section.

Table 6-8: ODT Characters and Symbols

Character(s) or Symbols	Meaning
/ k/	Opens the previously open location as a word or opens the location designated by k as a word.
\ k\	Opens the previously open location as a byte or opens the location designated by k as a byte.
" k"	Gives the ASCII representation of the currently open or last open location or of the location specified by k.
% k%	Gives the ASCII representation of the Radix-50 value in the currently open or last open location or in the location specified by k.
RET k followed by RET	Closes the currently open location or modifies the contents of the currently open location with the value k and closes it.
LF k followed by LF	Closes the currently open location and opens the next sequential location or modifies the contents of the currently open location with the value of k before closing it and opening the next sequential location.
^	Closes the currently open location and opens the preceding sequential location.
_	Takes the contents of the currently open location as a PC relative offset and calculates the next location to be opened; closes the currently open location and opens the location thus evaluated.
@ k@	Takes the contents of the currently open location as an absolute address, closes the currently open location, and opens and prints the contents of the location thus evaluated. If k precedes @, the value k replaces the contents of the currently open location before it is closed.
> k>	Takes the low order byte of the currently open location as a relative branch offset and calculates the address of the next location to be opened; closes the currently open location and opens and prints the contents of the relative branch location thus evaluated. If k precedes >, the value k replaces the contents of the currently open location before it is closed.
<	Closes the currently open location and opens the last location explicitly open. Returns ODT to the origin of a sequence of relative locations determined by _, @, and > character operations.
,	Separates a relocation register number from an octal value. ODT adds the contents of the specified relocation register to the octal value following the comma and forms a relocatable address.
;	Separates multiple values in a list request using the L character and in a register operation using the R character.
.	Specifies the last explicitly open location similar to that used by the < character.
+ space bar	Adds the preceding value and following value and uses the result.
–	Subtracts the following value from the preceding value and uses the result.
R	Resets all relocation registers to –1 (177777).

(continued on next page)

Table 6-8: ODT Characters and Symbols (Cont.)

Character(s) or Symbols	Meaning
nR	Resets relocation register n to -1 (177777).
k;nR	Sets relocation register n to the value k.
F	Sets relocation calculation for list requests using the L character.
1F	Disables relocation calculation set by the F character.
C	Prints out monitor table symbolic names and memory addresses.
$S	Prints out the processor status word.
Q	Uses the last quantity printed by ODT.
k1;k2L	Prints the contents of locations k1 through k2 at the terminal.
1;k1;k2L	Prints the contents of location k1 through k2 on line printer unit 0.
2;k1;k2L	Prints the contents of location k1 through k2 on another device. ODT asks for a device designator by printing the DEVICE question.

6.4.1 Running and Terminating ODT

Run ODT by typing:

```
$ RUN $ODT(RET)
```

After you press the RETURN key, ODT prints an identification line and the FILE question, and waits for your response:

```
ODT V8  RSTS V8    TIMESHARING
FILE<MEMORY>?
```

Your response to this question determines the way ODT runs and what address space ODT accesses. Table 6–9 lists the possible responses to the FILE question.

Table 6–9: ODT File Question Responses

Response	Meaning
Type the RETURN key only.	Allows read access to memory only if the user is privileged.
Type a file specification followed by the RETURN key	Allows read access to the file on the device specified. If you specify no device, ODT uses the system disk.
Type the file specification followed by the LINE FEED key.	Allows read and write access to the file specified.

ODT determines the address space by the response to the FILE question and can accept commands after printing the asterisk character. For example:

```
FILE<MEMORY>? ABC.DAT
*
```

ODT opens for read and write access the file ABC.DAT on the system disk under the current account. To terminate ODT, type CTRL/Z in response to the asterisk character. For example:

```
*^Z
$
```

ODT closes any file currently open and returns control to your keyboard monitor.

6.4.2 Access to Locations in the Address Space (/ and \)

ODT accesses the address space as either a word or a byte. You indicate the type of access by specifying the slash (/) or backslash (\) character. For example:

```
*1000/
```

Type the address 1000 (octal), followed by the / character. ODT opens the location as a word, generates a space, prints the six-digit octal contents of the word, generates another space, and leaves the location open for change. For example:

```
*1000/ 004100(RET)
*
```

ODT prints the contents of location 1000 as 004100. To close the location, press the RETURN key. ODT closes that location, prints an asterisk (*) character on the next line, and does not open a new location.

You use the backslash character (\) to open a location as a byte. For example:

```
*1000\ 100(RET)
*
```

ODT opens location 1000 as a byte, generates a space, prints the three-digit octal contents of the byte, generates another space, and leaves the location open for change. To close the location, type the RETURN key. ODT closes the location, prints an asterisk on the next line, and does not open a new location.

To change location 1000, type a new six-digit octal value followed by the RETURN key. For example, if location 1000 is open as a word:

```
*1000/ 004100 004000(RET)
*
```

ODT replaces the current contents 004100 with the specified contents 004000, closes the location, and prints the asterisk character on the next line.

To determine the contents of the current word location, type the slash (/) character. For example, if the current location is 1000, the following occurs:

```
*/ 004000
```

ODT opens the current location, generates a space, prints the six-digit contents and generates another space.

If you press the LINE FEED key while a word location is open, ODT closes the current location and opens the next sequential location. For example, if location 1000 is open and you press LINE FEED, the following occurs:

```
*1000/ 004000LF
001002/ 012345
```

ODT generates a carriage return and line feed and prints the address of the next location, followed by the slash and space characters and the contents of the word. The new location 1002 is open. Repetitive use of LINE FEED causes ODT to open and display the contents of sequential locations.

If you press the LINE FEED key while a byte location is open, ODT performs the same actions as described for a word location, except that the next location is treated as a byte.

6.4.2.1 Opening the Preceding Location (^) — Typing a circumflex (^) character when a location is open causes ODT to close the currently open location and to open the immediately preceding location.

For example, if two sequential locations are successively opened, typing the circumflex character opens the immediately preceding location:

```
*1000/ 002345LF
001002/ 012740^
001000/ 002345
```

Typing the circumflex character closes location 1002 and opens location 1000; ODT prints its contents. If a byte location is currently open, typing the circumflex character opens the immediately preceding byte location. If you type a value followed by the circumflex character, ODT replaces the current contents with the specified value before closing the location.

If you type the circumflex character and a location is not currently open, ODT opens and prints the contents of the last currently open word or byte location. For example:

```
*1000/ 002345
*^
001000/ 002345
```

ODT prints the address and the contents of the word location on the next line.

6.4.2.2 Opening a PC Relative Location (_) – If you type the underscore (_) character when a location is currently open, ODT adds 2 to the address of the current location, adds the resultant sum to the contents of the current location, and opens the location specified by the final sum. For example:

```
*1000/ 000040_
001042/ 012345
```

ODT closes location 1000, adds 2 to the address (1000), and adds the resultant 1002 to the contents (40) of the current location. As a result, ODT opens location 1042 and prints its contents. This method of calculating the next location to open is similar to that used in relative addressing by the program counter in the PDP–11 computer. Such a method of address calculation is for position-independent code.

If the contents of the current location is an odd value, ODT opens and prints the contents of the low-order byte of the PC relative location. If you type a value followed by an underscore character, ODT modifies the current contents and uses the new value to calculate the PC relative address.

6.4.2.3 Opening an Absolute Location (@) – Typing the at sign (@) character when a location is currently open causes ODT to take the contents of the current location as the address of the next location to open. As a result, ODT closes the current location, opens the calculated location and prints its contents. For example:

```
*1006/ 001024 @
001024/ 000500
```

ODT uses the contents of the current location (1024) as the next location to open.

If you type a value followed by the at sign character, ODT changes the contents of the current location to the value and uses the new value to determine the next location to open. The method is equivalent to absolute addressing on the PDP–11 computer, where the contents of the location following an instruction is taken as the address of the operand. The address is absolute because it remains constant regardless of where in memory the assembled instruction is executed.

6.4.2.4 Opening a Relative Branch Offset Location (>) – Typing the greater than (>) character when a location is currently open causes ODT to use the signed value of the low-order byte of the current location to determine an offset from the current location. ODT uses the final sum to open the next location and print its contents. For example:

```
*1032/ 000407>
001052/ 001456
```

ODT takes the contents of the low order byte (007) and multiplies it by 2 to give 16 (octal). Next, ODT adds 2 to the address of the current location (1032) to give 1034. Finally, ODT adds these two quantities (1034 + 16) to

give the address of the word (1052) to open. ODT closes the currently open location, opens the calculated location, and prints the address and the contents on the next line.

If you specify a value followed by the greater than (>) character, ODT modifies the contents of the currently open location and uses the low order byte of the new value to calculate the relative branch offset location. For example:

```
*1032/ 000407 301>
000636/ 000010
```

ODT interprets the byte value 301 as a negative value because the high order bit is 1. The absolute value of 301 is 77 (octal), which is multiplied by 2 to give 176 (octal). ODT subtracts the relative branch offset (176) from the address plus 2 of the current location to give 636 as the address of the next location to open. ODT opens the new location and prints its contents.

6.4.2.5 Returning to an Interrupted Sequence (<) – Typing the less than (<) character causes ODT to close the currently open location and open the last explicitly open location. This command is useful when you have typed the _, @, and > characters, or any sequence thereof, and wish to open the locations from which ODT calculated subsequent relative locations. For example:

```
*1032/ 000301 >
000636/ 000010 @
000010/ 123456 <
001032/ 000301
```

After you type the greater than (>) and at sign (@) characters, ODT opens location 10. You return to the last explicitly open location by typing the less than character. ODT opens and prints the contents of location 1032, the last location explicitly opened.

6.4.3 Printing the Contents of Locations

You can type the L character in three ways to print the contents of locations in the address space opened by ODT. For example, to print the contents of a certain range of addresses, specify the start and end addresses with the L command:

```
*0;776L
```

ODT prints at the terminal the octal contents of each word between address 000000 and 000776. At the beginning of each line of the printout, ODT prints the address of the first word on the line. Type a CTRL/O to turn the printing on and off. (That is, when you type CTRL/O, printing of the output stops; when you again type CTRL/O, the printing resumes.)

To print a listing on line printer 0, type the L command.

```
*1;0;776L
```

ODT prints, on line printer unit 0, the octal contents of each word between addresses 000000 and 000776. To specify another unit, type 2 preceding the command. ODT prompts you for the device name:

```
*2;0;776L
DEVICE? LP1:
```

Note that if you type any number other than 1 before the L command, ODT prompts you for the device designator.

6.4.3.1 Printing ASCII Format (") – If you type quotation marks (") when a location is currently open, ODT prints the ASCII representation of the word or byte. For example:

```
*1000\ 101  " A(LF)
001001\ 103  " C(LF)
*1000/ 041501
```

If the currently open location is open as a word, typing the quotation mark character causes ODT to print the two-character representation of the word. For example:

```
*1032/ 034567  " w9
```

The low order byte contains 167 (octal), which is w, and the high order byte contains 071 (octal) which is 9.

If you type the quotation mark character and a location is not currently open, ODT prints the ASCII representation of the previously open location.

6.4.3.2 Printing Radix–50 Format (%) – If you type the percent (%) character when a location is currently open, ODT prints the three-character ASCII representation of the Radix–50 word. For example:

```
*1000/ 034567  % IG1
```

If you type a value preceding the percent character, ODT interprets it as the address whose contents is to be interpreted and printed. For example:

```
*1000% IG1
```

ODT interprets 1000 as the address to use. If you type the percent character and a location is not currently open, ODT prints the three-character ASCII representation of the previously open location.

6.4.4 Relocation Registers

ODT has 16 relocation registers that you can use to specify relative addresses. ODT initially sets the relocation registers to −1 (177777, the highest possible address) to prevent inadvertent errors in address calculation. You set a relocation register by typing the relative address, followed by

a semicolon and the specification of one of the 16 relocation registers. For example, to set relocation register 0 to 1000, type:

```
*1000;0R
*
```

You can subsequently use the value in relocation register 0 as an offset or a base address in specifying a location. For example, to open location 1032, as a word, you type:

```
*0,32/ 000010
```

ODT adds the offset 32 to the contents of relocation register 0 to open the location. Because relocation register 0 contains 1000, ODT opens location 1032.

To reset the contents of all relocation registers to −1, you need to type only the R character. For example:

```
*R
*
```

ODT generates the carriage return and line feed operation and prints the asterisk character again.

To reset the contents of any one register to −1, simply specify the register number followed by the R character. For example:

```
*1R
*
```

This command resets relocation register 1 to 177777 (octal).

ODT treats registers 0 through 7 differently from 10 (octal) through 17 (octal) when used with disk files. For registers numbered 0 through 7, the leftmost three digits specify a block number and the rightmost 3 digits specify a byte location within the block. For example, 000017 designates byte 17 (octal) in block 0 of the file. The value 3412 designates byte 412 (octal) in block 3 of the file. (These registers are helpful in accessing data that is partitioned in 512-byte blocks.)

For registers numbered 10 (octal) through 17 (octal), the value specifies an absolute block number in the disk file. For example, the value 1000 (octal) specifies block 1000 (octal) of the disk file.

Use relocation registers 10 (octal) through 17 (octal) to access blocks in large files (greater than 65K blocks) that are beyond block number 65,535. To display the contents of an address in this range, type the offset in blocks and a semicolon, followed by one of the relocation registers 10 through 17. ODT accepts your input, performs the line feed operation, and issues the asterisk prompt. One of the relocation registers now contains a block address. You can use this value as a base address to access other locations. To begin

accessing addresses beyond the block offset contained in the relocation register, type:

1. The number of the relocation register with the block address (10 - 17)
2. A comma
3. The offset beyond the block address that must be accessed
4. A slash (/) character to terminate the response

ODT then adds the block address to the offset and displays the contents of the newly calculated address. For example, if you need to display the contents of a large file starting at block number 303240 (which is decimal block number 100,000), enter:

```
*303240;10R
```

You must first convert the decimal block number 100,000 to octal and enter this value along with the number of a relocation register, in this case register 10. ODT loads relocation register 10, and then prints the asterisk prompt. The relocation register now contains the block offset, 303240. This represents the address (30324000) from which you can access other locations. If you want to examine the contents of address location 303240030, for example, type:

```
*10R,30/ 000010
```

Because the increment beyond the base address 303240000 is offset 30 (octal), you must enter this value to access the contents of address location 303240030. ODT adds the octal number 30 to address 303240000 and displays the contents of location 303240030 to the right of the slash character. The contents of the resulting address location is 000010.

To print the contents of locations based on a fixed offset from the relocation registers, type the F character and subsequently use the L character. For example:

```
*F
*1;0;3000L
*
```

The F character conditions ODT to calculate relocated addresses for a printout. The next command tells ODT to add the offsets from 0 to 3000 to the value of relocation register 0 and print the contents of those resultant locations on line printer unit 0. The procedure is repeated for the values of relocation registers 1 through 7. The printout contains a listing of the contents of addresses n,0 through n,3000 where n is between 0 and 7 (the relocation registers).

To turn off calculation of relocation addresses for a printout, type 1F. For example:

```
*1F
*
```

Subsequent listing requests print out actual addresses rather than relocated addresses.

6.4.5 Interpretive Address Quantities (Q and .)

ODT uses the variable Q to store the last value that it printed at the terminal. You can type Q to designate the value so stored.

ODT performs any valid operation requested and automatically extracts the value from Q. For example, if you want to increase the value in an open location by a certain increment:

```
*1342/ 173214 Q+10
*/ 173224
```

You type 1342 to open that location as a word. ODT prints the contents of the location and stores that value in Q. You next type Q+10 and press the RETURN key. ODT adds 10 to the value in Q and modifies the current location with the sum. You thus do not need to retype the number or calculate the sum. To verify that ODT has changed the location properly, type the slash character. ODT opens the last previously open location (1342), prints the contents, and updates the value Q with the most recently printed quantity.

The period (.) character indicates the currently open or last explicitly open location. It indicates the current address for ODT operations. This is the same address used by the less than (<) character described in Section 6.4.2.5. In most cases, the period character value is the address used by ODT when you type the slash (/), backslash (\), quotation mark ("), percent (%), and LINE FEED characters. For example, to open as a word a location 16 bytes from the last explicitly open location, type:

```
*.+16/ 012345
```

ODT adds 16 (octal) to the address given by the character, opens the resulting address, and prints its contents.

6.4.6 Error Procedures

If you type an invalid or unrecognized character, ODT prints a question mark (?), generates a carriage return and a line feed, and prints the asterisk prompt. For example:

```
*1008?
*
```

ODT indicates that the character 8 is an error. Retype the number correctly.

If ODT encounters an error while performing output to a device, it prints the message I/O ERROR? followed by an asterisk. You must correct the device error and type the command again.

6.4.7 Reading a Pack Identification with ODT

Each RSTS/E file-structured disk has a pack label (or identification) that is required for logically mounting the disk. The pack label consists of one to six alphanumeric characters. You should record this information in a conspicuous place either on the pack or cartridge cover. Thus, when you need to mount a disk, you can easily see what its pack identification label is.

If you get a ?PACK IDS DON'T MATCH error, you may have mistyped the pack identification label on the MOUNT request. You may have also mounted the wrong pack or cartridge on the drive. The external label should tell you what the pack label is. If not, you can use ODT to determine what it is.

NOTE

It is much easier and safer to use the SAVE/RESTORE or DSKINT program to tell you what the pack label is. This manual contains explanations of both these programs.

The pack identification is stored in Radix–50 format at a relative, fixed location on all disks – offset 14 (octal) in the first pack cluster. For disks with a device cluster size of 1, the absolute address is 1014 (octal). For disks with a device cluster size of 2, the address is 2014 (octal). For cluster sizes of 4, 8 and 16, the addresses are 4014 (octal), 10014 (octal), and 20014 (octal) respectively. The following printout shows the procedure for a disk with a cluster size of 1:

```
$ RUN $ODT
ODT V8   RSTS V8   TIMESHARING
FILE<MEMORY>? DL1:(RET)

*1014/ 062053 % PAC(LF)
1016/ 043054 % KID(RET)
*

$
```

In response to the file question, type the name and unit of the device on which the pack or cartridge is mounted. Terminate the response with either the RETURN or ESC key to open the device for read access in non-file-structured mode. ODT prints an asterisk when it is ready to accept commands. Typing 1014 followed by the slash (/) character opens location 1014 (octal) as a word. ODT responds by printing the octal contents of the location opened. Typing the percent (%) character causes ODT to print the first three characters of the pack identification, which is in Radix–50 format.

If you press the LINE FEED key after the first three characters are printed, ODT closes the current location and opens the next location. Repeating this procedure gives you the second three characters of the pack identification, after which you press the RETURN key to close the location and place ODT in command mode again. Finally, you type CTRL/Z in response to the asterisk prompt, but the ^Z does not echo.

Chapter 7
System Utility Operations

This chapter describes nine system programs that can help you manage your RSTS/E system:

UTILTY	Consists of a set of commands that can control system operations.
SYSTAT	Provides a set of program switches to check system status.
VT50PY	Dynamically displays the status of major software and hardware components on your system.
TTYSET	Sets the characteristics of terminals on your system.
DSKINT	Prepares a disk on line for system use.
ONLCLN	Repairs corrupt disk file structure during timesharing.
REORDR	Restructures disk directories to improve disk access time.
GRIPE	Stores user suggestions the system manager can read and process.
TALK	Allows a user to send a message to another user on the system.

7.1 General Utility Operations — UTILTY

You can control system operations on line with UTILTY program commands. Table 7–1 lists and briefly describes the commands available to you. The sections in this chapter explain each command in more detail. Because UTILTY has system-wide impact, RSTS/E assigns it a protection code of <124>. This allows only users with privileged accounts to execute its privileged operations. Restricting the use of privileged accounts to those with RSTS/E experience and system responsibility decreases the chance that your system operations are disrupted through the use of programs such as UTILTY.

7.1.1 Running and Terminating UTILTY

You run the UTILTY program by typing:

```
$ RUN $UTILTY
```

After you press RETURN, UTILTY prints a header line and a pound sign (#) prompt:

```
UTILTY V8 RSTS V8 TIMESHARING
#
```

The program is now ready to accept the commands described in Table 7–1. When you need information about UTILTY commands and you have no documentation handy, type HELP. The HELP text contains a list and a description of the commands available to you. Once you choose the command you want and execute it, UTILTY returns another pound sign prompt. You can then enter another command or end the program.

You can stop the execution of UTILTY with the EXIT command, CTRL/Z, or CTRL/C. If you type EXIT at any time during the execution of a command, UTILTY completes the command you issued, prints a pound sign prompt, and returns control to your keyboard monitor. Control also returns to your keyboard monitor when you type EXIT in reply to the program prompt. CTRL/Z works in the same way as the EXIT command. In contrast, CTRL/C stops execution of UTILTY before execution is complete. The system immediately echoes the ^C characters and returns control to your keyboard monitor. Because the effects of uncompleted internal system operations are unpredictable, use the EXIT command or CTRL/Z to end the UTILTY program.

There are two ways to run the UTILTY program:

1. The RUN command
2. A Concise Command Language (CCL) command

While the RUN command does allow you to execute UTILTY program commands, the CCL command format makes using UTILTY more convenient. If you add UT–ILTY as a CCL, privileged users can then run UTILTY by typing UT, entering a command, and pressing the RETURN key. UTILTY executes the command and returns control to their keyboard monitor. In the following example, UTILTY lists the CCL commands located in the CCL command table:

```
UT LIST CCL(RET)
```

If you use UT–ILTY as a CCL, typing UT followed by the RETURN key produces the same results as typing RUN $UTILTY.

Table 7–1: UTILTY Commands

	Command and Meaning
Operational Control	LOGINS Raises, to the maximum, the number of jobs allowed logged in to the system at one time. The number of jobs (JOB MAX) and the amount of available swapping space (SWAP MAX), both set during system generation, limit the maximum number of jobs allowed on a system. The smaller of these two values determines the maximum number. SET LOGINS *n* Sets to n the number of user jobs that the system can log in. NO LOGINS Prevents users from logging on to the system by setting the number of logins to one. SEND KB*n: xxx* Causes UTILTY to print the text xxx string on keyboard unit n or on all keyboards. This commands displays a message only on those terminals that are not gagged. ALL *xxx* Causes UTILTY to print the text string xxx on all keyboards. The command displays a message only on those terminals that are not gagged. FORCE KB*n:xxx* Causes UTILTY to force the text string xxx into the input buffer of keyboard number n as if you typed it at the KBn: terminal. This command displays the text if the terminal is not gagged. ALL *xxx* Causes UTILTY to force the text string xxx into the input buffer of all keyboards on the system, except those that are gagged. ALL is not a switch, so do not prefix it with a slash (/) character. Unlike the FORCE KBn:xxx command, which uses a colon (:) character between the keyboard number and the text, you cannot use a colon as a separator with ALL. SEIZE *dev:* Reassigns to your job number a device (dev:) currently assigned to another job. The SEIZE command returns an error message if a file is open on the device. KILL *n* Immediately terminates user job number n. PRIORITY *n p* Sets to p the priority of job n. Priority can be any number from –128 to +120. The system rounds the priority down to a multiple of eight. (For example, if you select a priority of –19, the system sets the priority to –24.) SUSPEND *n* Sets the priority of job n to –128. RESUME *n p* Resumes job n at priority p. Same as PRIORITY command. RUNBURST *n Q* Changes the run burst of job n to Q. The value for Q can be any number from 1 to 127.

(continued on next page)

Table 7–1: UTILTY Commands (Cont.)

	Command and Meaning
Operational Control (Cont.)	SIZE *n Q* Changes the maximum size of job n to Q. You can set Q to a maximum of 31K. DETACH *n* /CLOSE Detaches job n and closes all nonzero channels on which the console terminal is open. /NOCLOSE Detaches job n but does not close all nonzero channels on which the console terminal is open. HANGUP KB*n:* Disconnects the remote line specified by KBn:. DISABLE KB*n:* Disables the terminal interface of keyboard number n until the start of the next time-sharing session. DATE *dd-mmm-yy* *yy.mm.dd* Sets the RSTS/E system date to the value you specify, in either day, month, and year (dd-mm-yy) format or year, month, and day (yy.mm.dd) format. For example, you enter the date for October 31, 1982 by typing: 31-OCT–82 or 82.10.31. TIME *hh:mm* Sets the RSTS/E 24-hour clock to the value of hours and minutes (for example, 21:52 means 9:52 P.M.).
Disk Management	STALL Halts the scheduling of all jobs except the one issuing the STALL command. (This allows you to change or back up removable media on RC25 disks.) UNSTALL Allows the system to resume operations that were suspended with the STALL command. MOUNT *dev:id/switch(es)* Logically mounts the disk device on the drive you specify. Include the pack identification name with your device assignment. UTILTY must know the pack ID before it can mount the disk. (You assign the pack ID to the disk when you initialize it with the DSKINT option of INIT.) /LOGICAL:*xxxxxx* Uses the name you specify as the logical name for the disk in place of the pack ID. The name can have up to six characters. /NOLOGICAL Does not use any logical name. /PRIVATE Mounts this public disk as a private disk. /RONLY Does not allow write access to this disk. DISMOUNT *dev:* Logically dismounts a disk on the drive you indicate in the device specification. You must use the DISMOUNT command before removing the disk from its drive.

(continued on next page)

Table 7–1: UTILTY Commands (Cont.)

	Command and Meaning
Disk Management (Cont.)	LOCK *dev:* Prevents nonprivileged users from opening files on the disk and drive you indicate in the device specification. For example, the LOCK DL1: command allows only privileged users to open files on the RL02 placed on drive number one. UNLOCK *dev:* Allows nonprivileged users to open files on the disk and drive you indicate with the device specification. QUOTA *dev:[n,m] q* Set the maximum number of 512-byte disk blocks that account [n,m] can retain on the specified device (dev:) at log-out time. The value for q can be any decimal number from 0 to 65535. A zero value for q gives the user in account [n,m] an unlimited quota on the specified device. CHANGE *[n,m]password* Changes the current password of account [n,m] to the six-character name you include. For example, the CHANGE [1,244]BEYLE command line replaces the current password in account [1,244] with the new password BEYLE. If you use an asterisk (*) character as the password or use any question marks (?) in the password, this prevents any user from logging in to the system on this account. ZERO *dev:[n,m]* Deletes all files that are not write-protected from account [n,m] on the disk drive specified by device.
Run-Time System Control	ADD *dev:[n,m]name[/switch(es)]* Adds a six-character name to the run-time system table. Indicate the device on which you want the run-time system to reside and include an account number [n,m]. If you do not enter a device specification, UTILTY selects the public disk structure as the default. Account [0,1] becomes the default when you do not include a value for [n,m]. /EMT[:*x*] Causes the run-time system to use the RSTS/E special prefix EMT feature. The value x can be any number from 0 to 255. UTILTY selects the code 255 for the default if you do not specify a value for x. /EXT:*xxx* Uses xxx as a default file type for files executed under this run-time system. /MIN:*s* Uses s in K words as the minimum size job allowed. /MAX:*s* Uses s in K words as the maximum job size. /ADDR:*n* Uses n (1K-word section of memory) as the default load address. /KBM Indicates that this run-time system should act as keyboard monitor. /1USER Indicates that this run-time system supports only one job at a time. /RW Indicates that this run-time system allows read/write memory access.

(continued on next page)

Table 7–1: UTILTY Commands (Cont.)

	Command and Meaning
Run-Time System Control (Cont.)	/LOGERR Indicates that this run-time system logs errors occurring under its control to the system error file. /REMOVE Removes this run-time system from memory when all users are using another run-time system or when no jobs are using this run-time system. /POSITION[*:n*] Places the run-time system block in position n after the primary run-time system in the linked list. /STAY Indicates that the run-time system should reside permanently in memory. REMOVE *name* Removes the name of a run-time system module as an entry in the run-time system table. LOAD *name* Loads name as a run-time system into memory at the previous load point. The name you use can contain up to six characters. /ADDR:*n* Indicates the run-time system should be loaded into memory in 1K-word sections beginning at address n. /STAY Makes the run-time system reside permanently in memory. UNLOAD *name* Unloads the run-time system name from memory. NAME *name=filename* Associates name as a run-time system with the related executable file on disk. The file name in the NAME command line can be any valid RSTS/E file specification. When the system tries to execute this file, it uses name as the run-time system to load the executable file.
Resident Library Control	DEFAULT KBM *run-time system name* Changes the default keyboard monitor of your primary run-time system to the keyboard monitor of the run-time system you specify. The run-time system you select must already be installed on your system and have a keyboard monitor. The run-time systems RSX, RT11, and DCL all have keyboard monitors. ADD LIBRARY *dev:[n,m]filename.type* Adds name as a resident library. UTILTY uses [0,1] as the default account, the public disk structure as the default device, and .LIB as the default file type. /ADDR:*n* Indicates that the resident library should be loaded into memory in 1K-word sections beginning at address n. You must specify the /ADDR:n switch each time you use the ADD LIBRARY command. /STAY Makes the resident library reside permanently in memory.

(continued on next page)

Table 7–1: UTILTY Commands (Cont.)

	Command and Meaning
Resident Library Control (Cont.)	/1USER Allows only one user to access the resident library. /RW Allows the resident library to be mapped read/write instead of read-only. /NOLOGERR Prevents errors that occur within the resident library from being recorded in the system error log. /REMOVE Removes the resident library from memory when there are no jobs using it. REMOVE LIBRARY *name* Removes name as a resident library. LOAD LIBRARY *name* Loads name as a resident library into memory. /ADDR:*n* Indicates the resident library should be loaded into memory in 1K-word sections beginning at address n. /STAY Makes the resident library reside permanently in memory. UNLOAD LIBRARY *name* Removes from memory the resident library you specify by name.
System Logical Names Control	ADD LOGICAL *dev:[p,pn]name* Adds, as an additional name in the system logical name table, the name for the specified device type and unit. If you include an account number, the ADD LOGICAL command also associates that project-programmer number with the logical name. REMOVE LOGICAL *name* Removes the entire entry for name from the system logical table. It can also remove logical names for devices on your system. Device mnemonic names for system devices have permanent entries in the logical table. So, if you use the REMOVE LOGICAL command to remove a logical name associated with a device, only the logical name goes away, not the device mnemonic. You can then give a new name to that device. CHANGE LOGICAL *dev:name* Replaces the old logical name for a device with a new name. LIST LOGICAL Prints a list of device specifications that currently have a system logical name assigned. The list includes any account specification associated with the logical name.
Concise Command Language Control	LIST CCL Lists the Concise Command Language (CCL) commands that are in the CCL command table. UTILTY prints the CCL commands in the same order they were entered. That is, UTILTY does not sort the CCL commands before listing them.

(continued on next page)

Table 7–1: UTILTY Commands (Cont.)

	Command and Meaning
Concise Command Language Control (Cont.)	CCL command = program Defines in the Concise Command Language table the CCL command that can run the program you specify by program. The program name has the format: *dev:[p,pn]filename.type;PRIV n* If you do not include a device specification, the system assigns the program to the public disk structure. The PRIV indicator allows the program to retain temporary privileges for those programs that are entered at a nonzero line number. If you do not include PRIV in your file specification, any entry (at a nonzero line number) to a program stored with temporary privilege causes the program to drop the temporary privileges. The value n specifies the line number at which the program starts execution. The line number n defaults to zero. CCL *command* Deletes the command from the Concise Command Language table.
Caching Control	LIST CACHE Prints the current cache settings. ENABLE CACHE Enables data and directory caching. /ALL Caches all data transfers regardless of UFD entry or OPEN MODE. /FILE Caches files depending on their UFD entry or OPEN MODE. /NOFILE Caches no files. /BUFF Uses general small buffers for directory caching (not legal if data caching is generated). /NOBUFF Uses neither FIP small buffers nor general small buffers for directory caching. /CL:*n* Specifies a cache cluster size of n with values equal to 1, 2, 4, or 8 blocks. /LIMIT:*n* Specifies the maximum number of clusters (based on available memory) used for directory or directory and data caching. /DIR:*n* Specifies the maximum number of cache clusters used for directory caching. /DATA:*n* Specifies the maximum number of cache clusters used for data caching. FLAG *filename.type* Modifies the cache settings for a file.

(continued on next page)

Table 7–1: UTILTY Commands (Cont.)

	Command and Meaning
Caching Control (Cont.)	/CACHE Always caches the file on OPEN. /NOCACHE Does not automatically cache the file on OPEN. /SEQ Caches the file sequentially, if cached. /RAN Caches the file randomly (unless otherwise specified), if cached at all. DISABLE CACHE Disables data and directory caching.
System File Control	ADD SWAPFILE *n dev:[filename.type]/[/SIZE:n]* Adds swap file n with a file specification of [0,1]filename.SYS to a disk device. Valid swap file numbers (n) are 0, 1, and 3. If the device you specify is a file-structured disk, UTILTY requires swap files to have a file type of .SYS and be in the system account [0,1]. You can also use a non-file-structured disk as a swap file (ADD SWAPFILE 0 DS0:, for instance). /SIZE:*n* Creates a swap file with n contiguous blocks. REMOVE SWAPFILE *n* Removes swap file n (0, 1, or 3) from your system. LIST SWAPFILE Lists the swap files, overlay file, error message file, and the Network Services Protocol (NSP) system file at your terminal. The DECnet/E communications package requires the NSP file in order to run on a RSTS/E system. ADD OVERLAY dev:filename.type Uses a file on the device you specify to store FIP (File Processor) overlay code. REMOVE OVERLAY Removes the overlay file. The system resumes using the overlay portion of the installed .SIL file. ADD ERROR dev:filename.type Adds a file to the device from which the system can extract error message text. REMOVE ERROR Removes the error message file. The system resumes using the established default error message file. SNAP Records in the CRASH.SYS file the current state of the system so that you or field service has a way to investigate reasons for system malfunctions.
Program Control	HELP Displays a list of UTILTY commands and their definitions at your terminal.

7.1.2 Operational Control of the System

Certain UTILTY commands control the operation of the system. Table 7–1 lists and briefly describes these commands under the category of operational control. This section describes the commands in more detail and gives examples of their use.

7.1.2.1 Controlling the Number of Logged-In Jobs – You can monitor and control system operation while logged in to a RSTS/E system. With the SYSTAT or the VT50PY system program, you can observe how the system is performing. If performance declines, you can remedy the problem, in some cases, by preventing more users from logging in to the system. You adjust the number of logged-in jobs allowed with the SET LOGINS, NO LOGINS, and LOGINS commands.

NO LOGINS sets to 1 the number of jobs that are allowed logged on to the system. After UTILTY executes the NO LOGINS command, the system allows no more users on the system. If a user attempts to log in or to execute a logged-out command, the monitor prints:

```
?NO LOGINS
```

Jobs already logged in to the system can continue running. However, jobs such as BATCH that log other jobs in to the system cannot successfully process further requests. (The SHUTUP system program disables further logins in preparing to shut down timesharing.)

To keep you from being locked out of the system, RSTS/E allows jobs to be logged in at the system console terminal (KB0:), regardless of the number of logins currently allowed. (This feature is a special characteristic of the terminal you designate as the system console – the default is KB0:. By installing a patch, you can change the terminal having this characteristic. However, only one terminal on a system can have this capability.) Of course, this capability remains in effect as long as a job slot is available to handle a new job.

The SET LOGINS command places a ceiling on the number of users that can log in to the system. This command is useful for limiting or extending the load allowed on the system. For example:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#SET LOGINS 35
#
```

You have decided to limit the number of logins to 35; if there are 35 users logged in to the system and another user tries to log in, the 36th user gets the ?NO LOGINS message and must wait until someone else logs off before being allowed to log in.

The system does not let you set the number of logins to zero or to a number greater than the value of JOB MAX set at system start-up time. If you specify zero in the command, UTILTY sets the number to 1. The number of

logins allowed can never exceed the capacity of the swap space installed on the system. If you enter a number greater than the maximum allowed, UTILTY sets logins to the maximum possible. The free buffer status report of SYSTAT or the VT50PY program includes the maximum number of logins currently allowed.

You normally use the SET LOGINS and LOGINS commands with ADD SWAPFILE commands to adjust the number of logins allowed. After you add swap space to the system, the LOGINS command:

1. Recalculates the swap capacity
2. Factors in the currently allowed maximum swap size (SWAP MAX)
3. Increases the number of allowed logins

If you are preparing to remove swap space, use the SET LOGINS command to decrease the number of allowed logins.

7.1.2.2 Broadcasting Messages to Terminals – The SEND command lets you communicate with one user or with all users on the system. It places a specified text string in the output buffer of a terminal or all terminals and displays the text on the terminal. For example, if a user at the terminal KB32: assigns a peripheral device for an unreasonably long time, you can send a message asking the user to deassign the device:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#SEND KB32: Harry, can you deassign MM0: for me; I need it ASAP.
#
```

By specifying ALL in place of the device designator of a single keyboard, you can broadcast the message to each online terminal in the RSTS/E system (unless the terminal is gagged); for example:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#SEND ALL Please read $NOTICE.TXT; the system is coming down!!
#
```

The UTILTY program displays your message as follows:

```
** KB43 ** Please read $NOTICE.TXT; the system is coming down!!
```

This message appears on all terminals that are on line, except your own. UTILTY encloses your keyboard number in double asterisks, leaves a space, and then displays the text. Note that, when you send a message using ALL, you must leave a space between it and the beginning of your message. You cannot place a colon (:), as you do when you specify a device designator (KB32: for example), immediately after ALL. For instance:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#SEND ALL: Is anyone using DB0:?
Illegal Format - ALL: IS ANYONE USING DB0:? - in SEND
#
```

The ILLEGAL FORMAT message appears in uppercase letters. You must then determine the cause of your error and enter the message again.

7.1.2.3 Controlling Jobs – The FORCE command allows you to send a text string to another terminal (or all terminals). UTILTY places the text string in the input buffer of the specified terminal as if it were typed by the user. The command has the form:

```
FORCE KBn:xxx
```

If you want to send a CTRL/C to a terminal, KB32 in this case, to stop the execution of whatever is running at the terminal and then log the user off the system, type:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#FORCE KB32:^BYE/F
#
```

The circumflex (^) character before the command forces a CTRL/C to the terminal, and the BYE/F command logs the user off the system.

You can also send only a CTRL/C (^C) to a terminal by placing a circumflex character in the first position after the keyboard number:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#FORCE KB32:^
```

Do not place any text after the circumflex if you want to send only a CTRL/C to the terminal and stop the job that is running. UTILTY sends a CTRL/C to the terminal and returns control to the keyboard monitor prompt:

```
^C

$
```

To force a control character combination (CTRL/Z for example) to a terminal, enter the circumflex (^) character as the first character of the text, followed only by the proper control letter (such as Z):

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#FORCE KB32:^Z
#
```

Typing ^Z and pressing the RETURN key executes a CTRL/Z at the KB32: terminal. No other text should follow the control character combination. If the circumflex (^) is the first character of more than two characters of text, however, UTILTY forces a CTRL/C to the terminal before sending the text that immediately follows the first character:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#FORCE KB32:^KB32:^!H, Harry, sorry but your job is out of control!
#
```

The ^C characters appear on Harry's terminal to end the job he was running and UTILTY prints on the line below the ^C the text of the message. (The text after the first character can contain up to 80 characters.)

When you need to communicate with all terminals on your system, you can use the FORCE command as follows:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#FORCE ALL !Come to my office immediately. Good news!
#
```

Be sure not to place a colon (:) character between ALL and the text as you do when forcing a message to a single keyboard. UTILTY displays an ILLEGAL FORMAT message:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#FORCE ALL: Come to my office immediately. Good news!
Illegal Format - ALL: COME TO MY OFFICE IMMEDIATELY. GOOD NEWS!
#
```

The program returns to the pound sign prompt to let you enter the command again.

The KILL command, like FORCE, allows you to terminate a user's job. Before using KILL, run the SYSTAT program to learn the user's job number, and type:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#KILL 10
#
```

The monitor clears job 10 and frees it for other system use. The user is no longer logged in to the system and must log in again before attempting any other system activity. Run the SYSTAT program immediately after issuing the KILL command to verify that the job is no longer on the system.

There are times when you do not want to terminate a job, but rather want to place it in a hold state, a state from which it can resume normal processing. A job may be using up too many system resources, or you may be running a batch job that you need to correct. The SUSPEND command automatically sets the priority of a job to –128. This causes the system to stop giving the job any time to run. Use SUSPEND as follows:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#SUSPEND 10
#
```

Job 10 stops processing but does not terminate. SUSPEND always sets the priority to –128, which means you cannot include a priority number with the command. You must set the priority of the job to –120 or higher with the

RESUME or PRIORITY command to reactivate the job. RESUME is the logical command to use when you want to restart the job:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#RESUME 10 -8
#
```

It gives job 10 a –8 priority which under normal system load should allow the job to run to completion successfully. Use the PRIORITY command in the same format to produce the same results:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#PRIORITY 10 -8
#
```

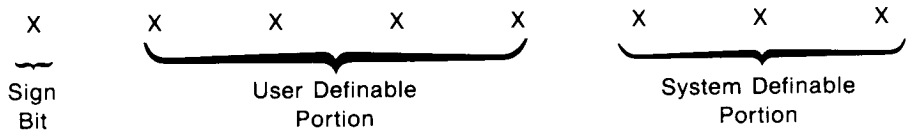
SUSPEND and RESUME are the logical commands to use when you want to temporarily stop a job and then to start it again. While PRIORITY does the same job, it is not apparent from the command name. There are tasks, such as decreasing the priority of a job using large amounts of CPU time, in which you may find it more appropriate and logical to use the PRIORITY command. The next section describes the concept of priority on RSTS/E and should provide the information you need to correctly use the PRIORITY command on your system.

7.1.2.4 Setting Job Priority, Run Burst, and Maximum Size – The UTILTY system program allows you to set the priority, run burst, and maximum size of an existing job. You can change any of the current values to:

1. Increase or decrease the chance of gaining run time in relation to other running jobs
2. Determine how much CPU time the job can have when it is compute bound
3. Increase the area a job can occupy

The system runs jobs on the basis of priority. The higher a job's priority, the better its chances are of obtaining run time in relation to other jobs that are running. An 8-bit priority byte determines priority, as shown in Figure 7–1.

Figure 7–1: Priority Byte Format



Using the PRIORITY command, you or another privileged user can set the user definable portion of the priority byte for any job on the system. Because the three system definable bits are normally zero, standard priorities are multiples of 8 between –120 (lowest priority) and +120 (highest priority). Zero is a legal priority. When UTILTY changes a priority, it truncates any value not a multiple of eight to the next lowest legal priority. For example, if you specify a priority of +10, UTILTY sets the value to +8.

All of the system definable bits are normally off (zero). The least significant bit is set when a keyboard delimiter is typed and the job was waiting for keyboard input. Keyboard delimiters are the CR, LF, FF, CTRL/Z, and ESC characters. The next significant bit is set whenever a CTRL/C is typed and can be set at any time. Finally, a system function call sets the most significant bit of the system definable portion. All system definable bits are cleared when another program is chained to or when the job is at the system command level.

The system definable portion of the priority byte is always less significant than the user definable portion. Therefore, the system definable bits affect priority only within the user definable priority range. If two jobs are running under priority –8, for example, the user who types a CTRL/C has a higher priority (that is, priority –6 in this case) than the user who does not. However, a third user with priority 0 supersedes two users whose priorities are –8 and –6.

When you log in, LOGIN runs with priority 0 and automatically sets your job to priority –8. This is the default priority with which most or all jobs are run. Only in unusual circumstances should you assign priorities other than –8.

On occasion, you may want to run a program that requires a great deal of computation. If time is not a factor in obtaining results, the privileged user can decrease the job priority to improve efficiency for the users on the system. Conversely, infrequently used detached programs often have higher priorities (typically priority 0) because they must run quickly when needed but do not run compute bound for an extended period and do not run often.

Run burst is the maximum time a job can run compute bound before another job obtains access to the CPU. On systems using the KW11L line frequency clock, each unit of run-burst time is equal to 1/60th or 1/50th of a second, depending on the system's power line frequency. Systems running with the KW11P clock at crystal speeds, rather than at line frequency, have a run-burst unit of 1/50th of a second. If the system is operating off a 60 Hz power line, one run-burst unit equals 1/60th of a second. In that case, six units equal 1/10th of a second, which is the run burst default value.

If you assign a run burst of six to a specific job which does not require that much compute bound time, the system automatically transfers control to the next user before the six units have been used. One tenth of a second is generally considered the best run-burst time period to insure efficient overall system operation. If a job is guaranteed to become I/O bound (that is, I/O

stalled) after a certain amount of computation, use UTILTY to specify a run burst larger than six. In many cases, a run burst greater than six has a significant effect on long computational programs.

The maximum size assigned to a job refers to the allowable memory space. You can allow certain jobs to run programs larger than 16K words by assigning a job a maximum size up to 32K words. This assigned limit does not affect privileged, compiled programs. Thus, a user with a small amount of space can still run system programs that would normally exceed the memory limit.

7.1.2.5 Suspending System Operations – The STALL and UNSTALL commands allow privileged users to temporarily suspend all disk I/O and other system operations. When the STALL command is issued, all jobs on the systems are suspended, except for the job issuing the STALL. Operations resume when the UNSTALL command is issued.

The STALL and UNSTALL commands are used primarily on systems where the system disk is an RC25. The main reason for using these commands is to suspend operations when you need to spin down the disk drive. With STALL and UNSTALL, a total system shutdown is not necessary.

To use the commands, type:

```
$ UT STALL (RET)
SYSTEM STALLED
```

The system displays SYSTEM STALLED to show that the stall is in effect. When the disk has spun down, you can back up one of the disks or change removable media.

When you are ready to restore operations, type:

```
$ UT UNSTALL (RET)
SYSTEM NO LONGER STALLED

$
```

7.1.2.6 Controlling Keyboards and Remote Lines – If you determine that a dataset line is in use but no keyboard activity is taking place (by SYSTAT or the VT50PY program job status report), you can disconnect the dataset. The HANGUP command disconnects the remote line specified by the KBn: keyboard. The hangup capability prevents a user from monopolizing the line without being charged for connect time and frees the line for other remote users.

The DETACH command frees a terminal from its job. If the job does not have the terminal open on a nonzero channel, DETACH frees the terminal for other use. The command normally forces the system to close all nonzero channels on which the terminal is open. The /NOCLOSE switch suppresses this action. When the job specified is currently detached, UTILTY prints the message:

```
JOB NUMBER x IS ALREADY DETACHED.
```

The DISABLE KBn: command can remove a single line terminal interface from use for the current time-sharing session. To disable an interface for subsequent sessions, use the SET option of the initialization code. (See the *RSTS/E System Generation Manual* for a description of the SET option.) An interface remains disabled until the start of the next time-sharing session, at which time it is reenabled unless it was disabled by SET.

7.1.2.7 Changing System Date and Time – The DATE and TIME commands allow you to change the system date and the 24-hour clock. The commands are important on systems where strict accounting data is kept and incorrect values may have been entered at system start-up time. You can enter the date in either of two formats:

DATE dd-mmm-yy
DATE yy.mm.dd

The following example shows how to enter the date using the first format:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#DATE 31-OCT-82
```

You can enter the same date in the second format as well:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#DATE 82.10.31
```

Be sure to place a space between the DATE command and the beginning of the date field when entering a date with either format.

7.1.3 Principles of Disk Management

You manage disks on the RSTS/E system with certain UTILTY system program commands. Table 7–1 briefly describes these commands under the category of disk management. The following sections describe each disk management command.

7.1.3.1 Preparing a Disk for Use on a Drive – Before using a disk on a RSTS/E system, you must first initialize it. This consists of:

1. Formatting the disk, if it is not already formatted by DIGITAL.
2. Performing pattern checks on the disk for bad blocks.
3. Writing a RSTS/E file structure on the disk.

Initialization is necessary for most disks, except ones such as swap and non-RSTS/E disks.

7.1.3.2 Formatting Disks – Formatting means writing the necessary timing and sense marks onto the disk and erasing any extraneous information. You must format the following disks before using them on your RSTS/E system for the first time:

RK05 RP02

RK05F RP03

RP04 RP05

RP06

You must initialize and format disks of this type with the DSKINT option of the initialization code (INIT.SYS). This is necessary because they are not formatted and the DSKINT option is the only RSTS/E software able to perform this operation. Refer to the *RSTS/E System Generation Manual* for a description of the DSKINT option.

You can but do not need to format the following removable disks (DIGITAL formats them for you):

RK06 RM03

RK07 RM05

RM02

The RL01 and RL02 disks are also formatted by DIGITAL at the factory, but cannot be formatted at the site. This means you can reformat RK06, RK07, RM02, RM03, and RM05 disks with DSKINT, but DSKINT will not reformat the following disks:

RL01 RL02

RS03 RS04

RM80 RC25

RA80 RA81

RA60 RD51

RX50

For disks that have been formatted, such as RL01 and RK06 disks, you can use the DSKINT system program described in Section 7.6 instead of the DSKINT option of INIT.SYS. DSKINT lets you initialize disks that have been formatted, but not disks that have NOT been formatted. Once you use the DSKINT option of INIT.SYS to format a disk, however, you can use the DSKINT system program to initialize it.

7.1.3.3 Converting Disks to Version 8.0 Format (RDS1) – If you have disks on your system that were initialized before Version 8.0 of RSTS/E (that is, disks in RDS0 format, or "RSTS Disk Structure 0"), DIGITAL strongly recommends that you upgrade the formats to RDS1 with the DSKCVT program. DSKCVT is available on your system if the BIGPRG programs are installed.

To run DSKCVT, you must start with a disk that does not need rebuilding. (Use the ONLCLN utility if necessary to rebuild the disk first.) As a precaution, DIGITAL recommends that you back up any disk before conversion.

DSKCVT requires some work space on the pack being converted. Therefore, you must ensure that there are 32 blocks times the number of different group numbers used on the pack, in order to allow conversion. (For example, if there are ten group numbers, then the pack should have 320 blocks of free space for conversion.)

DSKCVT accepts either logically mounted or logically dismounted packs. However, the pack should not be in use when you run DSKCVT. To verify that no one is using the pack, DSKCVT dismounts and remounts it at the start of processing.

The following is an example of the DSKCVT program:

```
$ RUN $DSKCVT (RET)
DSKCVT  V8  RSTS  V8  Timesharing

Disk to convert? DL0: (RET)
Ready to convert DL0:NEWODS
Proceed (Y or N)? Y (RET)

Sorting PPNs
Processing [0,1]
Processing [1,1]
Processing [1,2]
Processing [1,3]
Processing [1,211]
Processing [2,25]
      .
      .
      .

Conversion of DL0: complete
```

In the example, DL0: is the disk pack to convert to RDS1. When you type DL0: in response to the prompt, DSKCVT displays the pack ID DL0:NEWODS. This allows you to verify that you are converting the proper pack. If the pack was already converted, DSKCVT displays an error message and then repeats the "Disk to convert:" prompt. Otherwise, DSKCVT continues with the dialogue.

When you answer "Y" to the "Proceed (Y or N)?" prompt, DSKCVT begins the conversion. The accounts are processed in ascending order. When the pack is converted, DSKCVT dismounts it and updates the pack label. The pack is now in RDS1 format. However, the storage allocation table (SAT) does not yet reflect the free disk space on the pack. Therefore, the pack is marked "dirty" until DSKCVT updates the SAT. Then the pack label is updated to show that the pack is now rebuilt. If the pack was mounted when DSKCVT started, it will be mounted once again, with the pack ID as its logical name.

DSKCVT will not lose data even if the program aborts because of lack of work space, a system crash, or a disk hardware problem. If there is a problem during the conversion, you need only correct the problem and rerun the

conversion. In such a case, it is not necessary to restore the pack from a backup pack.

If the conversion proceeded far enough that DSKCVT was able to convert the pack, it is then necessary only to mount it. Otherwise, DSKCVT restarts the conversion from the beginning, after deleting the work files that were left behind from the first attempt at conversion.

7.1.3.4 Mounting Disks – To use a disk pack or cartridge on a RSTS/E system, you must logically mount it. This gives the system information about the device you plan to mount. You include the:

1. Device mnemonic
2. Unit number on which the device is mounted
3. Logical name of the disk
4. Pack-ID of the disk
5. Command to logically mount the device

The system compares the pack-ID label with the information on the disk. If the pack-ID label you supply does not match the one on the disk, UTILTY prints the error message:

```
Pack ID's don't match
```

Otherwise, the system records the presence of the disk in the table of mounted disks.

When the RSTS/E system is started, only the system disk is logically mounted. Make sure other packs in the public structure, and perhaps other private packs, are mounted by commands from the START.CTL or CRASH.CTL file. You can logically mount a private disk on the system by means of the UTILTY system program MOUNT command or use the CCL MOUNT command described in the *RSTS/E System User's Guide.*

The procedure to mount a disk is:

1. Use the SYSTAT system program to ensure that the drive unit is free.
2. Place the pack in the drive. When the drive is ready, write-enable it for normal use. Write-lock the disk if it is to be mounted read-only.
3. Run UTILTY and use the MOUNT command to logically mount the new pack. (If you have DCL on your system, use the DCL MOUNT command instead, as described in Chapter 11.)
4. Rebuild the pack if necessary. To rebuild a disk mounted read-only, it must be dismounted, mounted write-enabled, rebuilt, dismounted, and mounted read-only. Use the ONLCLN program to rebuild the disk, as described in Section 7.7.
5. Use the UNLOCK command to free the device for access by nonprivileged users.

To remove a disk from the system:

1. Run UTILTY and use the LOCK command on the drive unit containing the pack to be removed.
2. Use the SYSTAT program's disk status report to determine the number of open files. If zero, proceed. If nonzero, wait until all files are closed before proceeding.
3. With no files open on the device unit, run UTILTY and use the DISMOUNT command to notify the system that the pack is being removed.
4. Remove the pack from the disk drive unit.

Table 7–2 contains several common errors you encounter when using disks on your RSTS/E system:

Table 7–2: Disk Error Messages

Message and Meaning
`?Device hung or write locked` The disk you are trying to access is mounted read-only, and thus you cannot write to it. Under no circumstances should you write-protect a public disk, mount it read-only, or dismount it during normal time-sharing operations. Users create and access files in the public structure without explicit reference to a device. By removing a public disk during timesharing, you deny users access to files that reside on that device.
`?Not a valid device` The disk type or unit you specified is not configured on the system.
`?Illegal SYS() usage` The disk you specified is either already mounted or is the system disk.
`?Pack IDs don't match` The pack identification does not correspond to the one on the disk. (Use the SAVE/RESTORE or DSKINT program to tell you the pack identification.)
`?Fatal disk pack mount error` The disk has an unreadable storage allocation table, a cluster size larger than 16, or some other nonrecoverable error condition. The disk may have been used on an operating system other than RSTS/E.

Procedures for removing a private disk or cartridge from a drive unit are similar to those used to prepare the disk for system use. For example, if you want to replace a private disk with another private disk, you must:

1. Make sure no files are open on the drive unit. Use the SYSTAT or VT50PY program to check the disk status report.
2. Lock the device unit by using the UTILTY LOCK command. After you use LOCK, nonprivileged user programs cannot open any more files on the disk.

When the disk you are about to remove has no open files and is write-locked (with LOCK), you must logically dismount the device with the DISMOUNT

command. If any files are open on the disk, UTILTY prints an error message and does not dismount the disk:

```
?Account or device in use
```

After successfully completing the dismount operation, you can safely remove the disk from the drive.

If you physically remove a disk before logically removing it, and you afterwards try to use the UTILTY MOUNT command to mount the disk, the program prints the warning message:

```
?Disk pack needs REBUILDing
```

The disk is mounted but locked and read-only. The SYSTAT display program prints the text LCK in the disk structure report. The lock condition prevents nonprivileged jobs from accessing the disk but allows privileged users to rebuild the disk.

An improperly dismounted disk requires rebuilding because the Storage Allocation Table (SAT) in the SATT.SYS file does not reflect the actual allocation of storage. This happens because the SAT is manipulated in memory and is not written back to the disk immediately. Proper dismounting procedures ensure the SAT is updated on disk before the disk is physically removed from the drive.

Because the MOUNT command of UTILTY does not rebuild disks, you must use the ONLCLN program to rebuild the SAT. (However, the DCL MOUNT command rebuilds the disk if required.) The ONLCLN program:

- Reads all the directory blocks and creates a new SAT that reflects occupied storage
- Resets all file access counts to zero
- Deletes all files with a .TMP file type

The MOUNT command of UTILTY does not unlock the disk. To allow nonprivileged jobs access to the disk, you must use the UNLOCK command. After you execute the UNLOCK command, nonprivileged jobs can open files on the disk.

The MOUNT command not only mounts a disk pack or cartridge but also determines whether a logical name should be placed in the system logical name table. If the mount operation succeeds, the system scans the entire name table. If the name is not in use, the system places the pack identification in the table entry for the specific device and unit. The MOUNT routine does not enter in the table a pack identification that duplicates a logical name in use. In this case, UTILTY prints a message:

```
Logical name was not unique
```

This means the disk is properly mounted but you cannot access it by its disk logical name. Use the CHANGE LOGICAL command to give a unique logical name to that specific disk unit entry in the table.

7.1.3.5 Disk Mounting Switches – MOUNT command switches let you:

- Change the logical name of the disk
- Suppress any logical name
- Mount a public disk as private
- Mount a disk for read-only access

The /LOGICAL and /NOLOGICAL switches control the system logical name for a specific device and unit. By specifying the /NOLOGICAL switch, you prevent the pack identification from being entered in the system logical name table. This keeps the pack identification from being reported by SYSTAT or the VT50PY program and limits access to the disk by its physical name. Later, the CHANGE LOGICAL command can establish a logical name for the disk, or the ADD LOGICAL command can create an additional logical name entry by which the disk can be accessed.

The /LOGICAL switch replaces the pack identification with the logical name you include in the switch. UTILTY mounts the disk and then determines whether the logical name is unique. If the name is not in use as a system logical name, the system places the specified name, instead of the pack identification, in the table entry for the specific device and unit. If the name duplicates one currently in use for any device, UTILTY prints the message:

```
Logical name was not unique
```

The system mounts the disk, but it has no associated logical name. You can use the CHANGE LOGICAL command to establish a unique logical name for the disk or the ADD LOGICAL command to create an additional logical name entry by which you can access the disk.

By adding the /PRIVATE and /RONLY switches to the MOUNT command, you can mount a public system disk from another RSTS/E system. The /PRIVATE switch causes the system to treat the disk as a private pack and the /RONLY switch prevents users from creating or altering files on the disk. The effect of these switches preserves the integrity of the foreign and host system's public file structures and allows privileged users read access to data files on the public pack or cartridge. Also, you can include the /RONLY switch with the MOUNT command to restrict write access to any private disk. When using the /RONLY switch, write-lock the disk.

7.1.3.6 Removing All Files from an Account – The ZERO command of UTILTY removes all files from an account on a device, except the files that are write-protected. If the account contains write-protected files, use one the following to change the protection code before zeroing the account:

1. DCL SET PROTECTION command
2. /RENAME switch of PIP
3. BASIC-PLUS NAME-AS statement

Then use the REACT program to delete the account. Refer to Section 4.1.2 for a description of the DELETE function of REACT.

7.1.3.7 Changing the Quota and/or Password of an Account – Each user account in the RSTS/E system has associated with it a quota of disk storage that the account can retain at log-out time and a password that allows access to the system. You set the quota and enter the password when creating the account with the ENTER function of the REACT program. (See Section 4.1.1) In order to log out, users must be under quota for each disk on which they have an account. The monitor checks all read-write mounted disks at the time of logout.

The QUOTA command of UTILTY changes the maximum amount of disk storage an account can retain while still allowing the user of the account to log out. If the user accumulates any number of 512-byte blocks above the number you set with the QUOTA command, the user must delete enough blocks in the account to fall below the quota size. To set the quota, specify the disk, account number, and the decimal number of 512-byte blocks of disk storage you want the account to contain. If you do not specify a disk, the system disk SY: is assumed.

For example, you set the quota to 5000 for account [1,210] on the system disk as follows:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#QUOTA [1,210]5000
#
```

Therefore, account [1,210] has a quota of 5000 512-byte blocks. Because no disk is specified, the system disk is assumed.

If you do not want to restrict the number of blocks a user can retain at log-out time, assign a quota of zero to the account:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#QUOTA DB1:[1,210]0
#
```

A quota of zero tells the monitor not to check the quota for account [1,210] on DB1: on logout. The user can now log out with any number of blocks in the account.

When you specify a quota size of zero, the only restriction is that the account cannot contain more blocks than the disk has available (free blocks). Therefore, you must be aware of the capacity of the disk when you create or change a quota for a user account. The quota must be in the range of 0 to 65,535 blocks.

You can change the password of an account with the CHANGE command. (Passwords are useful only on system disks.) For example, if you want to

change your password to QWERTY, and your account number on disk DR0: is [1,211], type:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#CHANGE DR0:[1,211]QWERTY
#
```

If you want to make an account unavailable for use, place an asterisk (*) character after the account number, or assign a password which contains at least one question mark (?):

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#CHANGE DU1:[1,196]*
#
```

Before anyone can log in to the account, you must use the CHANGE command again to give the account a valid password. Deactivating an account is useful when you no longer have use for the account but may in the future. The MONEY program, described in Chapter 4, prints six question marks (??????) in the password field to help you identify the accounts deactived with the CHANGE command.

The QUOTA and CHANGE commands do not affect the entries in the ACCT.SYS file.

7.1.4 Run-Time System Control

A run-time system on RSTS/E is the common, shareable part of a user job and, in the case of BASIC-PLUS, is the interface between the user's executable code and the monitor. Run-time systems control execution of jobs when the monitor allows the jobs to run. The monitor sets up jobs and establishes an environment in which a run-time system can function. The BASIC-PLUS run-time system interprets requests made by a user job, translates those requests into a format the monitor understands, and transmits the requests of the monitor. The monitor processes the requests and passes either data or an error to the job. The run-time system decides whether to interpret the data or log an error. Jobs running under the BASIC–PLUS run-time system cannot make direct requests to the monitor.

Other run-time systems are not language interfaces in the same sense the the BASIC-PLUS run-time system is. Jobs under their control can make direct requests of the monitor. In such cases, the major role of the run-time system is to set up compiled programs for running, handle system errors, and return control, on program termination, to the job keyboard monitor. Some of these run-time systems can also emulate the action of certain system directives from other operating systems, such as RT11 or RSX.

A run-time system interprets user requests when it is a language interface for a user job. A computer language provides a standard means by which many users can define data, process it, and obtain results. The language

relieves each user of the need to learn the workings of the monitor. The run-time system under which the language operates is more efficient because multiple users can share the same code. This ability to share is more efficient for the system because each user need not have his own copy of the common code.

All RSTS/E installations have a primary run-time system and can optionally have auxiliary run-time systems. The primary run-time system, BASIC-PLUS at many RSTS/E sites, allows users access to system resources, interprets system command requests, and performs housekeeping chores. An auxiliary run-time system can provide any other shared computer processing functions and usually enables processing in a language other than BASIC–PLUS. Such an auxiliary run–time system may compile and/or execute BASIC–PLUS–2, COBOL, DIBOL, FORTRAN–IV, FORTRAN–77, or other language operations.

The system account [0,1] stores all run-time systems as contiguous files with .RTS file types. The primary run-time system must be stored on the system disk because, at the start of timesharing, the system disk is the only disk mounted on the system. You can store auxiliary run-time systems on any disk, either public or private. (Auxiliary run-time system files can be stored in accounts other than [0,1].) All run-time system files are contiguous because they must be loaded into memory in the fastest time possible.

You create the primary run-time system during system generation. Other run-time systems, each having a distinct name, can exist simultaneously on a system disk. With the DEFAULT initialization option, you select one of them as the primary run-time system that is loaded into memory automatically when timesharing starts.

The DEFAULT initialization option creates the monitor parameters that describe the primary run-time system:

1. The name of its file
2. Its location on the disk
3. Its characteristics

These parameters are fixed for a given time-sharing session and cannot be altered during a session. Through the DEFAULT option, you can optionally locate the primary run-time system in memory. This location cannot be changed during a given time-sharing session.

At system generation time, you not only can select your primary run-time system, but you also can choose the keyboard monitor you want as your default. After you start timesharing, RSTS/E allows you to select a run-time system other than your primary run-time system to be the default keyboard monitor. DEFAULT KBM is the command you use to change the default keyboard monitor (selected at system generation time) to another run-time system, one that must include a keyboard monitor. For example you might use RSX as your permanent primary run-time system, but choose DIGITAL

Command Language (DCL), which cannot be your primary run-time system, as the default keyboard monitor:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#DEFAULT KBM DCL
#
```

This procedure changes your default keyboard monitor to DCL, which remains the default KBM during the current time-sharing session. The system default does not need to be permanently resident, though it must be added.

Auxiliary run-time systems are created during system generation or afterwards. The monitor structures that describe these run-time systems do not automatically exist at the start of timesharing. Therefore, you must provide the commands to add monitor structures for auxiliary run-time systems you want to make available for any given time-sharing session. The structure for an auxiliary run-time system is dynamic and can be removed during a time-sharing session.

The UTILTY run-time system commands allow you to control operations of auxiliary run-time systems. The run-time system commands execute system function calls to affect the particular operations. The UTILTY commands, however, perform additional processing for certain commands. To assist the system programmer, the command descriptions distinguish between the monitor call action and the UTILTY command action. Because run-time systems under development need certain special controls, some features of run-time system control are intended for DIGITAL personnel only. Nevertheless, all features are described to give you a broader perspective.

It is recommended that you make auxiliary run-time systems available by commands in the START.CTL file. By reading the descriptions in this section, you can understand what actions should occur for each run-time system command. The descriptions in Section 3.1 tell you how to organize the run-time system commands so that the desired actions are performed automatically at the start of each time-sharing session.

7.1.4.1 Adding and Removing Auxiliary Run-Time Systems — Because the structures that define auxiliary run-time systems are transient, you must create them for each time-sharing session. The ADD command creates the necessary structure to enable an auxiliary run-time system to function. ADD command switches can alter predefined characteristics of the run-time system.

The ADD command creates a run-time system description block with a general small buffer. A file with the name of the run-time system and a file type of .RTS can be any account on the disk specified in the command. A run-time system with the same name cannot already exist on the system. The system ensures that the file found is contiguous, has the proper format, and has proper parameters.

You can specify switches to override predefined characteristics. The ADD command extracts information from the run-time system file to establish the proper entries in the run-time system description block. To establish the entries, UTILTY opens the file, reads it, and sets up the predefined characteristics to be placed in the description block. The predefined characteristics are changed by the switches: /EMT:n, /REMOVE, /LOGERR, /RW, /1USER, and /KBM.

You can also change the predefined characteristics by the negation of these switches. For example, if a run-time system that has no special command decoding capabilities were being tested, you would use the switches /1USER, /RW, /NOLOGERR, and /REMOVE to add the run-time system. The /1USER ensures that no other job tries to share the untested code; /RW is necessary for setting breakpoints with ODT; /NOLOGERR prevents spurious errors generated by the run-time system from cluttering the system error log; and /REMOVE ensures that the run-time system's image is not left in memory but rather is reloaded from disk immediately preceding every entry to it. The switches and negations of the switches do not alter the predefined characteristics in the run-time system file; they alter only the characteristics defined in the description block. The characteristics remain in effect until the run-time system is removed.

The /ADDR:n switch allows you to take advantage of high speed memory on the system or to avoid fragmentation of memory. Without the /ADDR:n switch, the monitor decides where to load the run-time system each time residency is required. With the switch, the monitor loads the run-time system at the specific 1K-word section of memory. You must include the /ADDR:n switch whenever you use the /RW switch.

Because 1K-word section numbering begins at 0 and ends at n–1 (where n is the total size of memory), the 1K section number in the /ADDR:n switch is one less than the physical section number. For example, to load the RT11 run-time system (4K words in size) into the 61st through 64th 1K sections of memory, specify /ADDR:60. The run-time system is loaded from low memory to high memory at its defined initialized size. To be loaded without error, enough contiguous user space must be available starting at that location. The location specified in the /ADDR:n switch becomes the default location at which the run-time system is loaded during the current time-sharing session. You need to change the location only if you change the allocation of the section of memory with either the DEFAULT or the START initialization option.

One precaution is necessary when specifying the address at which the run-time system is loaded. The section of memory chosen must not fragment the user job space to prevent the run-time system from executing a job. For example, assume a system has 24K words of user space available between the 36K and 60K sections of memory. Assume also that a job requires 18K words of user space to run and that the run-time system requires 4K words when resident. If the loading address is 36K, the space between 40K and 60K remains available for an 18K job to run. If the loading address is 42K, the user space is fragmented into two sections — one from 36K to 42K and one from 46K to 60K. An 18K-word job area is not available to execute a job using this auxiliary run-time system.

The system verifies that the memory section given in the /ADDR:n switch is reasonable. If the entire range of memory starting at the load address is not available, the system prints:

```
?Illegal byte count for I/O
```

You should consult the memory status report of a display program to select an available range of memory. If the range of memory results in fragmenting that would cause a swapping violation, the system prints the error message:

```
?No room for user on device
```

A swapping violation occurs if the memory to be occupied by the run-time system does not allow enough contiguous space for a maximum sized job to run. To avoid excessive disk activity on small systems, you should include the /ADDR:n switch when adding auxiliary run-time systems.

Table 7–3 summarizes the errors that can occur when you are adding a run-time system.

Table 7–3: Run-Time System ADD Command Errors

Text and Meaning
`?Can't find file or account` A file with the name specified and with a file type of .RTS cannot be found in the account on the device given.
`?Illegal byte count for I/O` The range of memory starting at the load address given is not available. Refer to the memory status report of a display program (SYSTAT or VT50PY) to select an available range of memory.
`?Name or account now exists` A run-time system with the same name currently exists.
`?No buffer space available` Adding a run-time system description block requires a general small buffer, but one is not currently available.
`?No room for user on device` If the monitor were to load this run-time system at the address specified, memory would be fragmented and a swapping violation would occur.
`?Protection violation` The file to be added as the run-time system has a bad format. For example, the file is not contiguous or has illegal entries in the SIL index.

The /POSITION:n switch denotes the position in the linked list of blocks in which the run-time system block is to be placed. The primary run-time system block is always first in the list. If the run-time system you are adding is to be accessed frequently, you can reduce system overhead by placing it in the right position in the list. Without the switch, the system adds the block to the end of the list. If n in the /POSITION:n switch is one, the block is placed immediately after that of the primary run-time system. If n is zero or a value greater than the number of currently defined run-time systems, the block is added to the end of the list.

The position of a run-time system description block in the linked list affects how the system treats a RUN request for a file without a file type. On receiving such a RUN request, the system checks the indicated directory for all files with the specified name and a executable file type. For example, if the directory contained three files of the same name with file types .BAC, .TSK, and .SAV, the system checks for the run-time system nearest the primary run-time system in the linked list of description blocks. If BASIC–PLUS were the primary run-time system (whose block is always at the root), the system runs the .BAC version of the file.

The /EXT:xxx switch on the ADD command changes the default file type used for a executable file. If the switch is not given, UTILTY extracts the file type defined in the run-time system file. This default file type is applied when a user types a RUN command for a program and does not specify a file type.

The /MIN:s and /MAX:s switches alter the minimum and maximum job sizes (in K words) defined in the file for the run-time system. Generally, the minimum value allowed is one and the maximum is 31, but a given run-time system may not allow this full range of job sizes. For BASIC–PLUS, the minimum and maximum sizes are two and 16, respectively.

The /STAY switch makes the run-time system permanently resident. Usually, an auxiliary run-time system is temporarily resident. It occupies memory as long as the currently active job is running under its control. The system frees the memory it occupies when that memory is needed to load another run-time system or to load a job running under another run-time system. (Thus, the run-time system may be nonresident if there are active jobs using it.) The system automatically loads the run-time system when a user requires its services. If you specify the /STAY switch, the auxiliary run-time system becomes permanently resident, as the primary run-time system is. The only way to make such a run-time system nonresident is to use the UTILTY command UNLOAD.

The REMOVE command reverses the steps the ADD command performs. You must remove all auxiliary run-time systems before you can shut down the system. The SHUTUP program performs this removal operation automatically. The system checks to ensure that no jobs are currently running under control of the run-time system to be removed. If the usage count is 0, the description block is removed from the list and the run-time system file in account [0,1] is closed.

7.1.4.2 Loading and Unloading a Run-Time System – The LOAD and UNLOAD commands load and unload run-time systems. The commands perform explicitly the same actions the system performs automatically. You have control over placing the run-time system in memory. This control is useful in cases when a run-time system resides in high speed semiconductor memory and you want to specify its position.

The /ADDR:n switch on the LOAD command specifies a loading address as does the same switch on the ADD command. On the LOAD command, however, the effect of the positioning depends on what was specified in the ADD

command. Having specified a permanent load address in the ADD command, you indicate a special need to have the run-time system loaded at a fixed location. With the /ADDR:n switch on the LOAD command, the system changes that special location to the new address. Without the /ADDR:n switch on the LOAD command, the system uses the location defined in the ADD command.

If you did not specify an address in the ADD command, the system assumes there is no need for a special address. An address specified in the LOAD command applies, therefore, only for the current residency. If the monitor must load the run-time system later, it reverts to deciding the load address.

The /STAY switch on the LOAD command makes the run-time system permanently resident unless the run-time system is currently running a job. If this is the case, a LOAD/STAY is equivalent to a LOAD with no /STAY switch. The /STAY switch is helpful in cases where a special load address was not defined in the ADD command. The /STAY and /ADDR:n switches in the LOAD command ensure that the run-time system resides in the desired memory space.

The UNLOAD command frees the memory occupied by the run-time system. If a job is currently resident or being loaded and intends to use the run-time system, the system prints:

```
?Account or device in use
```

7.1.4.3 Associating a File with a Run-Time System – The NAME command changes the name of the run-time system associated with a file. Every disk file on RSTS/E has stored, in its directory, the name of the run-time system under which it was created, except files that are larger than 65,535 blocks. On a RUN request for the file, RSTS/E checks the name to find out what run-time system to use for the job. RSTS/E automatically switches control to that run-time system which, in turn, executes the file.

To change the name, UTILTY attempts to open the file to establish write access. If write access is denied, it prints:

```
?Protection violation
```

The name of the run-time system is written in the directory of the file specified. This naming operation is done during the system library build procedures when a file designed to run under an auxiliary run-time system is transferred from the distribution medium using PIP under BASIC–PLUS.

7.1.5 Resident Library Control

A resident library is a collection of shareable routines or data that the task builder links together into a task image file on disk. The MAKSIL program (see the *RSTS/E Programmer's Utilities Manual*) formats this disk file into Save Image Library (SIL) format. You can store resident libraries on any disk, either public or private. (Resident library files can be stored in

accounts other than [0,1].) All resident library files are contiguous because they must be loaded into memory in the fastest time possible.

You then use the UTILTY system program ADD LIBRARY command to assign the task image portion of the SIL file to a contiguous region of physical memory. Note that you can also use monitor SYS call –18 to assign the task image portion of the SIL file to memory. Once the body of shareable routines or data is linked, formatted, and assigned to memory, it becomes a resident library that is accessible to user tasks as part of their virtual address space.

You must be a privileged user to add a resident library to the monitor's list. There are two ways to add a resident library:

1. Monitor SYS calls to FIP
2. UTILTY system program commands

This section describes the UTILTY commands, and the *RSTS/E Programming Manual* contains a description of the monitor SYS calls.

7.1.5.1 Resident Library UTILTY Commands – You can use the UTILTY program to add, remove, load, or unload a resident library. To distinguish between resident library operations and run-time system operations that use the same command names, UTILTY requires the keyword LIBRARY in its syntax for library operations. You include the keyword LIBRARY in the UTILTY command line between the operation keyword and the target resident library name.

The UTILTY command to add a resident library is:

```
ADD LIBRARY name/ADDR:n</switch>
```

The name you include in the ADD LIBRARY command identifies the resident library. To identify it in more detail, you can specify a device, a project-programmer number and a protection code in addition to the library name. If you do not specify these variables, the device defaults to SY:, the project-programmer number becomes account [0,1], and the library's protection code defaults to <42>. When you add a resident library, UTILTY requires that you use the /ADDR:n switch to locate the library at a specific point (n) in memory. The n variable can be any number from the lowest to the highest available memory locations in increments of 1K-word sections. You can also attach switches to the ADD LIBRARY command:

/STAY	Makes the resident library permanently resident.
/1USER	Allows only one user access to the resident library code.
/RW	Maps the library read/write rather than read-only and is necessary for setting breakpoints with ODT.
/NOLOGERR	Prevents spurious errors generated by the resident library from entering the system error log.

/REMOVE Ensures that the resident library is removed from memory immediately after the last user has detached from the library.

The /STAY switch allows you to make a resident library permanently resident. Usually, a resident library only temporarily remains in memory. That is, it occupies memory only as long as currently active jobs run under its control. However, when you need to access the library again, the system automatically reloads it. To force the library to remain permanently in memory, you must specify the /STAY switch in the ADD LIBRARY command.

If you want to remove a resident library from memory, delete the monitor structure that defines the library, and close the library file. Use the UTILTY command:

```
REMOVE LIBRARY name
```

Note that the REMOVE command makes the resident library inaccessible for sharing, but the /REMOVE switch, which can be used only with the ADD LIBRARY command, only temporarily removes the library from memory while it is not in use.

7.1.5.2 Loading and Unloading a Resident Library – The LOAD and UNLOAD commands perform the same functions the system performs automatically. You can use LOAD along with the /ADDR:n switch to place the resident library where it can most benefit the system or use UNLOAD to remove the resident library to free memory for other system functions. Usually, LOAD/ADDR:n becomes useful when you want to position the library in high speed memory. The format of the LOAD and UNLOAD commands is:

```
LOAD LIBRARY name<⁄switch>
UNLOAD LIBRARY name
```

You can specify two switches with the LOAD LIBRARY command, /ADDR:n and /STAY. If you do not use the /ADDR:n switch, UTILTY places the resident library in the location you specified in the ADD LIBRARY name/ADDR:n command. When you include an address with LOAD LIBRARY name/ADDR:n, you override any location previously set by ADD. If you want to ensure that the resident library remains in the desired memory space, attach the /STAY switch to the LOAD command:

```
LOAD LIBRARY name/ADDR:n/STAY
```

Note that the /ADDR:n switch forces the system to use the LOAD specified address.

The UNLOAD command removes a resident library from memory and thus frees memory that can then be used to load another library or be available for other system purposes. If you attempt to unload a resident library that is in the process of being loaded or is in use by the currently running job, the system prints the message:

```
?Account or device in use
```

To prevent inadvertent loss of data, the monitor does not automatically unload a library you have loaded with the /RW switch. Unloading a library removes it from memory but does not write it back to disk. This means UTILTY discards the library and retrieves a new copy from the system library when it is needed again. Thus, anything you write to a library that you mark read/write is lost after you unload it with the UNLOAD command. You must use the UNLOAD command to unload a resident library that is designated read/write.

Whether a library remains in memory depends on the switches you attach to the ADD LIBRARY command. The /STAY and /RW switches prevent the monitor from removing it automatically. In addition, the monitor does not remove a library when it is attached to it.

7.1.6 System Logical Names

RSTS/E allows users to access devices by logical names as well as by physical names. Logical names that apply to all users are termed system logical names. On all systems, users can refer to a disk by its pack identification or by a name that replaces the pack identification. Thus, each disk unit configured on RSTS/E systems has the capability of being accessed by a system logical name. Logical names that apply to a single job are referred to as job-related logical names.

You can define system logical names for nondisk devices and additional names for disk devices. The number of names allowed is a system generation parameter and varies from system to system.* Such a system logical name can have an account number associated with it. Consequently, use of the name refers not only to the related device but also to the account on the device.

The system treats device names in the following manner:

1. Checks the list of job-related logical names first. Each job can have up to four assigned logical names.
2. Scans the table of disk logical names for a matching pack identification or equivalent.
3. Checks the additional names.
4. Checks the list of valid physical device names.

If individual users have not defined job-related logical names that duplicate currently defined system logical names, all users have access to devices by the logical names the system manager adds.

---

* If the number of names configured is zero, an attempt to define a system logical name always returns the ?NO ROOM FOR USER ON DEVICE error because no table space is available. See the *RSTS/E System Generation Manual.*

A system logical name must be unique. Thus, a pack identification or its logical name cannot conflict with a pack identification or other system logical name. You can, however, define multiple system logical names for the same device.

If the pack identification (or a logical name to replace the pack identification) of a disk you are to mount is the same as an existing system logical name, then the disk is mounted with no system logical name. Unless you specify a different, unique name, the disk, at mount time, has no logical name associated with the device and unit being used. (For more information on the disk logical name, see the description of the LOGICAL switch for the MOUNT command.)

System logical names apply for a single time-sharing session but you can remove or change them during the session. It is suggested that you place the proper commands in the system start-up control files to automatically define names at system start-up time. For more details on system logical names, refer to the *RSTS/E Programming Manual.*

7.1.6.1 Adding New Names – The ADD LOGICAL command defines a system logical name for nondisk devices and an additional name for a disk device. If the name duplicates one currently defined, UTILTY prints the error:

```
?Account or device in use
```

To change a currently defined name, use the REMOVE LOGICAL command and then enter the ADD LOGICAL command again.

The logical name must be a legal file name. (That is, it must contain from one to six alphabetic or numeric characters.) The following message indicates that the name contains illegal characters:

```
?Illegal filename
```

If all entries in the table are occupied, the system prints:

```
?No room for user on device
```

You must configure on the system the device with which the name is associated. If you do not, the system prints:

```
?Not a valid device
```

7.1.6.2 Removing Logical Names – The REMOVE LOGICAL command deletes the association defined for the logical name specified. Because a system logical name must be unique, you need to specify only the name in the command. If the name you include is not currently defined, the system prints:

```
?Can't find file or account
```

If you do not enter a name or use an illegally formed name, the system prints:

```
?Illegal filename
```

To delete all logical names for a device, you must use the REMOVE LOGICAL command for each currently defined name.

7.1.6.3 Changing a Disk Logical Name – The CHANGE LOGICAL command replaces any currently defined pack identification or logical name. SYSTAT and the VT50PY program include this name in their disk status reports. The command does not affect any additional name for a disk. For example:

```
#CHANGE LOGICAL DB1:SYSTST
#
```

The device you specify must be a disk or the system prints the message:

```
?Not a valid device
```

If the device you specify is a disk but is not configured on the system, the system prints:

```
?Can't find file or account
```

The system checks the uniqueness of the new logical name. If it duplicates a name already defined, UTILTY prints:

```
?Account or device in use
```

You must specify a new name in the command. If you omit the name or specify the name in an illegal format, UTILTY prints:

```
?Illegal filename
```

To change the name of a nondisk device or to change an additional name for a disk device, use the REMOVE LOGICAL and ADD LOGICAL commands in sequence. To delete all logical names for a disk, specify the REMOVE LOGICAL command for each currently defined name.

7.1.6.4 Listing System Logical Names – The LIST LOGICAL command prints, for each device assigned a logical name (or pack identification), the device designation and unit number, any account number associated with the logical name, and the logical name itself. The entries are printed in the order in which they are found in the monitor tables.

7.1.7 Defining Concise Command Language (CCL) Commands

CCL commands on RSTS/E allow users to type system-level commands that load and run programs from disk. The programs must be coded to recognize a

CCL entry and to extract any command string passed to them. The *RSTS/E Programming Manual* describes the operation and interpretation of CCL commands.

Certain RSTS/E programs can interpret a standard set of CCL commands. The UMOUNT program runs only by the CCL commands MOUNT and DISMOUNT. A RSTS/E installation can have any number of unique CCL commands. You make all CCL commands available to users by the UTILTY CCL command which creates the monitor structure required.

Because the monitor structure that defines a CCL command is transient, CCL commands must be defined at the start of each time-sharing session. It is suggested that you place the proper command definitions in the start and crash control files. In this way, INIT creates the proper structures automatically when it starts the system.

Each CCL command definition occupies one general small buffer on the system. All definitions are in a linked list of small buffers. UTILTY executes a SYS call to add or remove CCL definitions and provides a way to list all currently defined CCL commands.

7.1.7.1 Adding a CCL Definition – To add SY-STAT as a CCL command, run the UTILTY program with the RUN command and type:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#CCL SY-STAT=[1,2]SYSTAT.*;PRIV 30000
#
```

This command adds the definition for the SYSTAT system program. The hyphen designates the abbreviation point so that typing SY runs the SYSTAT program. If you do not include a hyphen in your CCL definition, UTILTY places a hyphen at the end of the command and allows no abbreviation. UTILTY assumes the public disk structure (SY:) when you do not specify a device. Programs you run with a CCL command can reside on any disk device, but you must include the the device in the command definition, if it is not in the public structure. The account designation [1,2] in the example means SYSTAT must reside in the system library. If you do not specify an account with the CCL command, the system assumes the program is located in the account in which the job (that is, the job of the user later invoking the CCL) is running.

The asterisk in SYSTAT.* indicates a wildcard file type. In the case of multiple versions of SYSTAT, the version of SYSTAT that is run depends on the order of the run-time systems in your run-time system list.

The number following the semicolon is the line number at which execution starts when the SYSTAT program runs by CCL command. The value 30000 is standard for most BASIC–PLUS programs. "PRIV" in the example means that the program must retain its temporary privilege when it runs at a nonzero line number.

The system performs error checking before adding the CCL command. If the command is invalid, UTILTY prints:

```
?Illegal file name
```

A valid command can consist of:

- A string of one to nine alphanumeric characters, the first of which must be a letter, or
- Any one of the following characters: @, $, #, %, or &

Lowercase letters are equivalent to uppercase in the definition and use of CCL commands.

A CCL command may not begin with a number because BASIC–PLUS processes line numbered input as a statement to be compiled. The command can have a maximum of nine characters. If more than nine characters are present or if the equal (=) character is omitted, UTILTY prints:

```
?Illegal format
```

When you add two or more commands that begin with the same character or set of characters, you must define the program with the largest number of characters (to the left of the hyphen) first. You then define the program with the next fewest characters to the left of the hyphen, until you have made all the assignments. For example, you must define MACR–O before adding MAC–RO, or UTILTY prints:

```
?Account or device in use - in CCL
```

Not only does the program print this message when the definition contains an abbreviation that would cause misinterpretation but also when it finds the command is currently defined.

You may not add two or more commands that begin with the same character or set of characters. If you need to add a longer CCL defintion after you have already added a similar but shorter one, you must first remove the shorter entry. You then add the longer CCL command which allows you to reinstall the CCL with the shorter command name. For example, if you had installed MAC–RO and at some later date needed to add MACR–O, you would have to remove MAC–RO, install MACR–O, and finally install MAC–RO again. Section 7.1.7.3 describes how to remove CCL entries.

To add a definition for a program that must retain its temporary privilege when it runs at a nonzero line number, include PRIV in the command:

```
*CCL COM-MAND=DK1:[1,18]FILE.BAC;PRIV 30000
#
```

Include PRIV in your command line to ensure BASIC-PLUS keeps a program's privilege when you run the program with a CCL command.

7.1.7.2 Listing Currently Defined CCL Commands – The LIST CCL command prints a listing of currently defined CCL commands. The list of the CCL commands on your system may contain some of the following:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#LIST CCL
$-=SY0:[0,1]DCL.DCL;PRIV 0
ATT-ACH=[1,2]LOGIN.*;PRIV 30000
BYE-=[1,2]LOGOUT.*;PRIV 0
DCL-=SY0:[0,1]DCL.DCL;PRIV 0
DIR-ECTORY=[1,2]DIRECT.*;PRIV 30000
DIS-MOUNT=[1,2]UMOUNT.*;PRIV 30000
HELL-O=[1,2]LOGIN.*;PRIV 0
HE-LP=[1,2]HELP.*;PRIV 30000
LBR-=[1,2]LBR.TSK;0
LIBR-=[1,2]LIBR.SAV;8208
LIN-K=[1,2]LINK.SAV;8208
MACR-O=[1,2]MACRO.SAV;8216
MAC-=[1,2]MAC.TSK;PRIV 0
MOU-NT=[1,2]UMOUNT.*;PRIV 30000
PIP-=[1,2]PIP.SAV;8208
PL-EASE=[1,2]PLEASE.*;PRIV 30000
QU-EUE=[1,2]QUE.*;PRIV 30000
SET-=[1,2]TTYSET.*;PRIV 30000
SUB-MIT=[1,2]QUE.*;PRIV 30000
SW-ITCH=[1,2]SWITCH.*;PRIV 30000
SY-STAT=[1,2]SYSTAT.*;PRIV 30000
TE-CO=[1,2]TECO.TEC;3584
TKB-=[1,2]TKB.TSK;PRIV 0
UT-ILTY=[1,2]UTILTY.*;30000
EDT-=[1,2]EDT.TSK;0
#
```

The $ CCL and DCL CCL allow you to issue DCL commands following the $ and DCL from any command environment. For example, from BASIC-PLUS you can type:

```
DCL MOUNT DL0:MINDY/REBUILD/NOSHARE (RET)
```

UTILTY extracts a pointer to the root of the linked list of CCL definition entries, reads information from each CCL definition and prints the commands in the order in which they were defined. If no CCL commands are currently defined, UTILTY prints the message:

```
No CCL commands now installed
```

7.1.7.3 Removing a CCL Definition – To remove a CCL definition, type the command CCL and the definition followed by the equal (=) sign. You can type either a full definition or a valid abbreviation to indicate the command you plan to remove:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#CCL MYPR-OG=
```

If the definition does not currently exist, the system prints:

```
?Can't find file or account
```

The command removes the CCL definition immediately.

7.1.8 Data Caching Control

Data caching stores blocks from a user file for direct memory access. Directory caching stores the Master File Directory (MFD), User File Directory (UFD), monitor overlay code, and other frequently accessed system file directories.

When a user job executes a read request, the RSTS/E monitor performs a disk access and transfers the requested block(s) of data from the disk to the user job's buffer. With data (and/or directory) caching, the monitor stores the most recently read data blocks in an area of memory called the cache. If a user job executes a read request for a data (and/or directory) block in the cache, the monitor copies the requested data directly from the cache into the job's buffer. Because the system can thus retrieve data blocks from memory instead of disk, physical disk accesses are decreased. This can result in improved I/O throughput and faster response time.

Data caching is most useful for read operations because it can minimize disk transfers. Every write operation causes an actual write to the disk. In a write operation that modifies existing cached data, the data is updated both in the cache and on the disk.

You must explicitly specify a choice of data caching during system generation (see the *RSTS/E System Generation Manual*). There are three types of data caching you can generate:

1. No caching
2. Directory caching only
3. Data and directory caching

If you select directory caching, caching is automatically enabled at system start-up. If you select data caching, caching is automatically enabled provided that at least 2K words of XBUF are allocated for data caching. Note that you cannot select data caching without also selecting directory caching.

You can also enable data caching with the UTILTY command ENABLE CACHE as described in Section 7.1.8.6. Moreover, UTILTY commands allow you to specify caching for a particular file by marking the file's UFD entry.

Under the BASIC–PLUS or MACRO programming languages, you can use MODE values in the OPEN statement or monitor directives to open a file for cached data access. Refer to the *RSTS/E Programming Manual* for information on the BASIC–PLUS OPEN statement and to the *RSTS/E System Directives Manual* for information about MACRO directives. However, the use of caching MODE values and monitor directives is privileged. As system

manager, you can use the UTILTY program to designate a file for caching by marking its UFD entry. Once you mark a file's UFD entry for caching, it is cached on OPEN regardless of the user's privilege, as long as caching is enabled on the system.

When caching is generated and enabled, the cache receives all data transfer requests that are otherwise directed to the disk driver. Read operations on data that is in the cache occur without placing a load on the disk driver. The monitor constantly updates the cache so that it contains the most recently requested data for cached files.

7.1.8.1 Size of the Cache – The RSTS/E monitor allocates space for the cache from the Extended Buffer Pool (XBUF). You specify the amount of this allocation with UTILTY command switches that set the size of a cache cluster (/CL:n) and the number of clusters in the cache (/LIMIT:n, /DIR:n, and /DATA:n). Section 7.1.8.6 describes these command switches. The size of a cache cluster (1, 2, 4, or 8 blocks) determines the amount of data that is treated as a unit in a read request and, in many cases, the number of read requests that can be resolved in the cache before access to the disk driver is required.

For example, when the cache cluster size is eight blocks, any read operation that installs data in the cache causes eight physically contiguous blocks (including the requested blocks) to be installed. To ensure that only requested data is read into the cache, make the cache cluster size equal to or less than the pack cluster size (set during disk initialization) of the disk with the most files to be cached.

7.1.8.2 Sequential and Random Caching Modes – The data in a file can be cached in two ways:

1. Sequential
2. Random

You can specify either of these modes for a particular file with UTILTY command switches as described in Section 7.1.8.8.

If a job executes a read operation on a cached file and the data is not in the cache, that data is installed in the cache if there is free space or if a current cache cluster is eligible for replacement.

A cluster is eligible for replacement if either of the following conditions is in effect:

1. The last block of the cluster has been read in sequential mode.
2. The cluster has been in the cache without being read for more than the minimum residency time (one minute). A feature patch allows you to change the minimum residency time (see the *RSTS/E Maintenance Notebook*).

7.1.8.3 Random Mode Caching – Caching a file in random mode is recommended if the file is index structured. That is, with an indexed file structure, the monitor does not access the data in the file sequentially but rather must check the file index each time it gets more data. So, it reads the index, the data, and the index again.

As an example of random cache operation, consider a read operation executed on a file whose UFD entry is marked for random caching. When a read on the cached file occurs, the monitor examines the contents of the cache to determine if the requested data is present. If the data is in the cache, the data is copied from the cache cluster that contains it. The data is made available to the program and the cache cluster's time of last access is updated.

If the requested data is not in the cache, the monitor first attempts to allocate more of XBUF to install the new data (cache limits permitting). If XBUF cannot accommodate more data or the request exceeds the caching limits set with UTILTY, then the monitor examines the list of cache clusters to find one that is eligible for replacement. If it does not find one, the monitor cannot install the requested data in the cache and a normal disk read is automatically performed. If the cache cluster is eligible for replacement, the monitor installs the requested data in that cache cluster and makes it available to the program.

7.1.8.4 Sequential Mode Caching – Caching a file in sequential mode is recommended if you access the contents of the file sequentially. As an example of sequential cache operation, consider a read operation executed on a file whose UFD entry is marked for sequential caching. When a read on the cached file occurs, the monitor examines the contents of the cache to determine if the requested data is present. If the data is in the cache, the data is copied from the cache cluster that contains it. The data is made available to the program. If, in the process, the last block of a cache cluster was read, that cluster is made available for replacement in the cache.

Furthermore, when a read operation is performed on any block of a cache cluster (except the last block), the monitor installs a full cluster of data in the cache. That is, if you use the UTILTY command switch /CL:n to set a cache cluster size of eight blocks and then execute a read on the first block of a file whose cluster size is eight or greater, requested data plus the next seven blocks from the disk are read into the cache. Thus, the contents of the next seven blocks can be satisfied from the cache. When the last block of the cache cluster is read, the cluster is immediately made available for the installation of new data as requested by other read operations.

When a read operation from a sequentially cached file causes more than one cache cluster to be read, all of the requested data blocks are made available to the program. However, no data is installed in the cache for any cluster whose last block was read. (That is, only the last cluster can be installed and only if it was incompletely read.) Thus, if the cache cluster size is defined as one block and sequential mode is specified, no data blocks are installed in the cache (that is, every data block is the last block in a cache cluster).

7.1.8.5 LIST CACHE Command – The LIST CACHE command displays the current caching parameters for your system. The listing includes information on cluster sizes, data caching, and whether general small buffers are used. For example:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#LIST CACHE
Caching enabled
Data caching available

Current settings:
        Cluster size : 4
        Data caching : File selectable
        Small buffers: No

Cluster  All      Dir      Data
Limits:  65535    65535    65535
#
```

In this example, LIST CACHE reports that:

1. Directory and data caching were installed on the system during system generation, and caching was enabled.
2. The current cache cluster size is 4 blocks, only selected files are cached, and general small buffers are not used.
3. No limit is set on the use of XBUF for the cache.

7.1.8.6 ENABLE CACHE Command and Switches – The ENABLE CACHE command and switches allow you to set caching parameters for the system as a whole. The ENABLE CACHE command lets you enable data and directory caching on the system. Note that commands in the system start-up command file normally enable caching automatically.

When you use ENABLE CACHE with no switches, the last settings specified are applied. The initial system defaults are:

```
/FILE/LIMIT:65535/DIR:65535/DATA:65535/CL:n
```

The value n was specified in the DEFAULT option of INIT or was allowed to default to 4 blocks during system initialization.

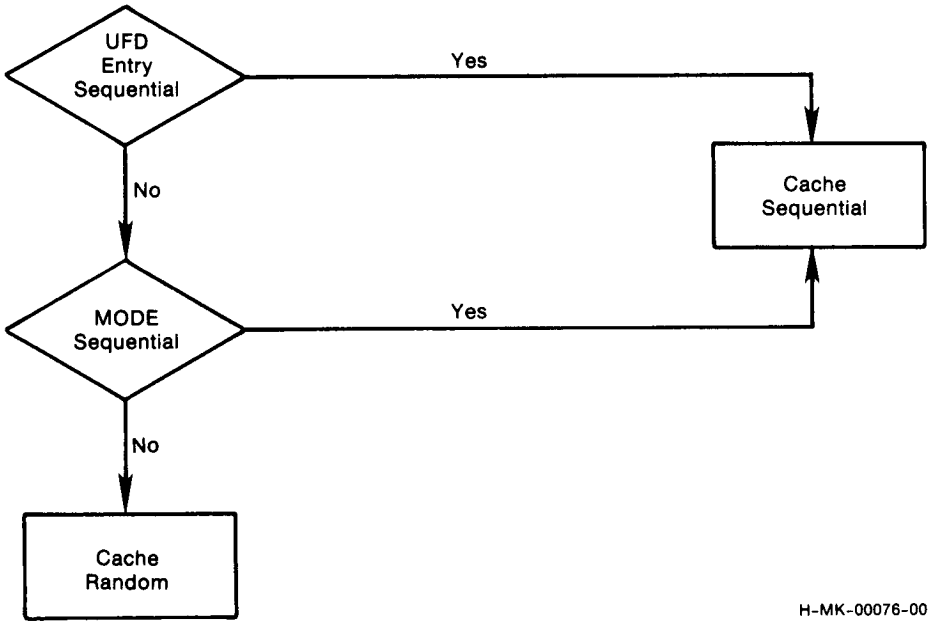
If you use ENABLE CACHE/ALL, all read requests are cached. The type of caching (sequential or random) for a particular file is determined by the file's UFD entry as specified in the FLAG command or by an OPEN MODE specification.

If you use ENABLE CACHE/FILE, a file is cached on the basis of its UFD entry and/or the specified OPEN MODE. That is, a file is cached if either the UFD entry or OPEN MODE specifies caching; it is cached sequentially if either the UFD entry or OPEN MODE specifies sequential.

If you use ENABLE CACHE/NOFILE, files are not cached. This switch is used to enable directory caching but disable data caching on the system.

Figure 7–2 illustrates the mechanism by which the monitor decides the amount of caching to be performed. The figure shows the flow of UTILTY commands and caching specifications examined by the monitor.

Figure 7–2: Monitor Caching Checks



The /BUFF and /NOBUFF switches specify the use of general small buffers for directory caching. These switches are meaningful only if directory caching was selected during system generation but data caching was not selected. If you specify an ENABLE CACHE/BUFF command on a system that has data caching installed, the following error message is printed:

```
?Missing special feature
```

If data caching is installed, you cannot use either FIP small buffers or general small buffers for caching.

Use the ENABLE CACHE/CL:n command to specify the cache cluster size as 1, 2, 4, or 8 blocks for each cluster. The cache cluster size controls the number of contiguous data blocks that are copied from disk to the cache whenever file data is cached. The cache cluster size should be small enough to contain only the target blocks but large enough to reduce the number of disk accesses. This means you must anticipate data requests and ensure that the cache cluster size is equal to the file cluster size of the most often accessed files.

The ENABLE CACHE/LIMIT:n command (where n is in the range of 0 to 65535) specifies the total number of cache clusters used by both directory and data caching. The default limit is 65535, which means that cache cluster allocation is limited only by the amount of XBUF available. The ENABLE CACHE/DIR:n and ENABLE CACHE/DATA:n commands (where n is in the range from 0 to 65535) specify the cache cluster allocation for directory and

data caching, respectively. The specified allocation is an upper limit. Thus, if a 40K-word XBUF is defined at system generation and /DIR and /DATA both specify a number of cache clusters equivalent to 25K, data can use space in the cache up to a maximum of 25K, which leaves a 15K minimum for directory caching. The reverse is also true. In this manner, data and directory caching are guaranteed a minimum allocation and the amount of overlap is controlled, which permits the cache to dynamically adjust to system and program requirements.

The default settings are:

/CL	Four blocks (unless altered during INIT.SYS)
/LIMIT	No limit on total cache allocation
/DIR	No limit on directory caching
/DATA	No limit on data caching

7.1.8.7 DISABLE CACHE Command – The DISABLE CACHE command disables all data and directory caching on the system. If you reenable caching following a disable and do not specify caching parameters (/CL, /LIMIT, /DIR, and /DATA), the default parameters are those that were in effect before the disable.

7.1.8.8 FLAG Command and Switches – You can set certain characteristics of a specific file with the FLAG command. Most of the switches that select these characteristics specify caching parameters. The three switches that do not are: /NOCTG, /PLACE, and /NOPLACE.

The format of the FLAG command is:

```
FLAG filename/[/switch]
```

The file name identifies the file for which you want to change characteristics. Replace the file name with a RSTS/E file specification. (See the *RSTS/E System User's Guide* for information on file specifications.) Table 7–4 contains a list of the switches you use with the FLAG command. You can specify more than one switch on a single command line, which saves you from issuing multiple command lines.

7.1.8.9 Caching Guidelines – The relationship between cache cluster size, which you set with a /CL:n switch, and the pack cluster size of the file can have an effect on the efficiency of caching. The cache cluster size determines the number of blocks on disk that are copied into the cache when a cluster is installed. Thus, if the cache and pack cluster sizes are equal, there is a direct correspondence between the data that will probably be read and the data copied from the disk to the cache. If the cache cluster size is larger than the pack cluster size, the data you request plus some possibly unrelated data is copied to the cache, thereby wasting cache clusters. If the cache cluster size is less than the pack cluster size, a read request may require multiple disk accesses to move all of the data into several different cache clusters.

Table 7–4: FLAG Command Switches

Switch	Meaning
/CACHE	Specifies that the file is automatically cached when open. The type of caching depends on the file's UFD entry and the specified OPEN MODE. If you use /CACHE alone on a file with no UFD setting or MODE specification, the default is random caching.
/NOCACHE	Specifies that the file is not automatically cached when open. To cache such a file, you must specify ENABLE CACHE/ALL (see Section 7.1.8.6) or specify a MODE value (see the *RSTS/E Programming Manual*).
/SEQ	Causes UTILTY to mark the file's UFD entry such that, if the file is cached, it is cached sequentially.
/RAN	Causes UTILTY to mark the file's UFD entry so that, if the file is cached, it is cached randomly. Note that you can override the random cache UFD setting with a sequential caching MODE specification. That is, a sequential caching specification with UTILTY or MODE always overrides random caching. A file that has never been flagged is equivalent to a file with the /NOCACHE/RAN switches attached.
/NOCTG	Allows a contiguous file to be extended by marking it as noncontiguous. This is done by clearing the contiguous bit. Once you mark the file as noncontiguous, access to the file may be slower than to a contiguous file, particularly with random access. Note that if you do not have write access to the file, the /NOCTG switch returns a "?Protection violation" error.
/PLACE	Indicates that the file must be located at a particular place on the disk.
/NOPLACE	Allows you to indicate that the file need not be located at a particular spot on the disk.

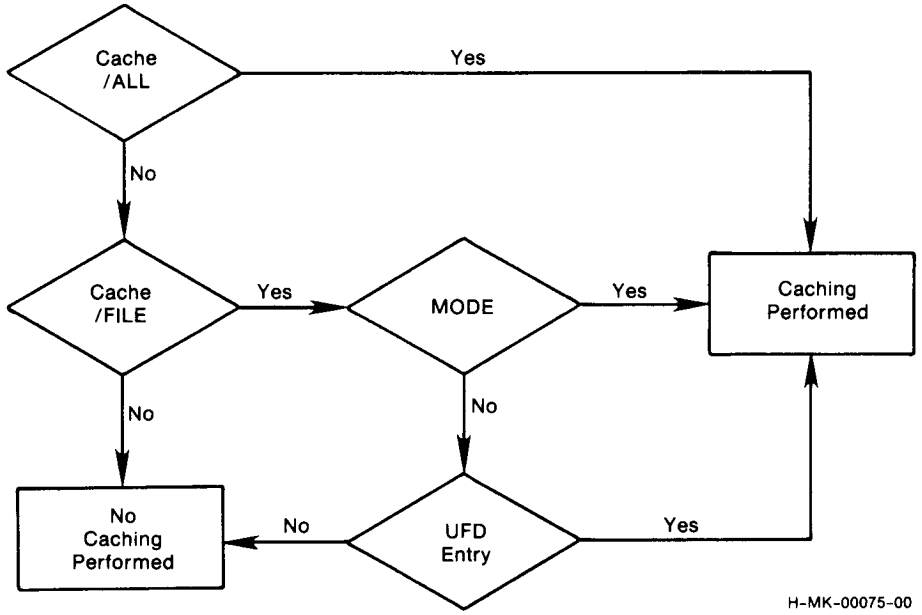
Figure 7–3 illustrates the flow of UFD entry and MODE value checks performed by the monitor. The result of these checks determines the type of caching to be used on the file.

The relationship between the cache cluster size and RMS indexed file bucket size is similar. That is, because RMS always reads an entire bucket at once, efficient caching is obtained when the cache cluster and bucket sizes are equal.

Consider the following guidelines:

- The amount of XBUF allocated to data and directory caching while setting defaults should be as large as possible while maintaining sufficient memory for user jobs, resident libraries, and run-time systems. However, an exceptionally large allocation can increase swapping activity, especially on systems with less than 128K words of memory.
- On systems that use RMS heavily, consider the use of the RMS resident library and the library's effect on XBUF allocation. Typically, on systems with less than 128K words of memory, the resident library should receive priority in memory allocation, to the extent of limiting XBUF allocation. In such a case, data caching may become ineffective and probably should be disabled.

Figure 7–3: Caching Mode Checks



- To optimize directory caching, set all UFD cluster sizes to 16 blocks if possible.
- Frequently accessed files should have their UFD entries marked for caching with the UTILTY command FLAG.
- Files that are frequently accessed sequentially (including RMS sequential and relative files) should have their UFD entries marked for sequential caching with the UTILTY command FLAG/SEQ/CACHE.
- Files that are frequently accessed randomly (including RMS relative and indexed files) should have their UFD entries marked for random caching with the UTILTY command FLAG/RAN/CACHE. Because of the way RMS indexed files are maintained, indexed files should always be flagged as random even if they are accessed sequentially. Also, where possible, RMS indexed file bucket sizes should be a power of two in order to correspond with caching cluster sizes.
- Use data caching judiciously on RMS indexed files. You gain the most benefit from caching an indexed file when it is subject to many read requests and is open for long periods of time. In addition, you should allocate more XBUF to such files to compensate for the lower caching "hit" rate on indexed file data. The "hit" rate is generally lower on indexed files than it is on sequential files or directory blocks.
- Where possible, all files should be contiguous to reduce window turning.

  Where contiguous files are not possible, you should ensure that you allocate the proper file cluster size for the files — that is, the file size divided by seven rounded up to the next power of two, not greater than 256 nor less than the pack cluster size.

- The minimum residency time is meant to keep some useful data in the cache, even when the cache is very small, and to make sure that if you want to access previously accessed data the system may find it in the cache, rather than having to read the information off disk each time you request more information. For systems with large caches or rapidly changing disk access patterns, it may be advantageous to reduce the minumum residency time from its default value of one minute. Section 3.1 in the *RSTS/E Maintenance Notebook* describes how to change the residency time.

Caching is not the most efficient data access method for all files under all circumstances. But, in general, a high-access file would be a logical candidate for caching. A high-access file is a file that meets one or more of the following conditions:

1. Accessed by more than one user at a time
2. Opened for long periods of time
3. Frequently accessed
4. Accessed sequentially (indexed files)
5. Primarily read
6. Is an overlayed program (such as a user program, TKB, or DTR)

However, because of the large number of different applications, some experimentation is needed to arrive at the best use of caching on your system.

7.1.9 System File Control

By using the REFRESH initialization option, you allocate and position the following system files on file-structured disks:

- SWAP0.SYS
- SWAP1.SYS
- SWAP3.SYS
- OVR.SYS
- ERR.SYS

These optional files optimize system performance by taking advantage of higher speed disks and the characteristics of user jobs running on the system. Because these files are contiguous and contiguous space becomes scarce as user files are created during timesharing, preallocation and prepositioning are necessary. The *RSTS/E System Generation Manual* describes the planning considerations for preallocating and positioning these system files.

RSTS/E does not automatically access these system files. You control RSTS/E access to these files with UTILTY commands. Thus, the use of these

files during timesharing is dynamic. If a hardware problem occurs on a swap device, you can remove the device without stopping timesharing. To maintain the same amount of swap space, you can designate a private disk as the swap device.

This section discusses the UTILTY commands that control system file usage. Use the INIT system program commands described in Section 3.1 to implement some of these controls. The commands in the start-up control files allow access to system files at the start of time-sharing operations.

7.1.9.1 Adding and Removing Swap Files – The swap files SWAP0.SYS, SWAP1.SYS and SWAP3.SYS in account [0,1] are standardized names for the three optional swap files. These names are optional but highly recommended because they tell the characteristics of the files. You can, however, assign any file name to these three swap files. The system restricts the file type to .SYS regardless of the file specification you include in the UTILTY commands.

The ADD SWAPFILE command causes the system to access a specific file (if a file-structured device is involved) or a specific device (if a non-file-structured disk is involved). A typical command sequence is:

```
#ADD SWAPFILE 0 DS0:
#ADD SWAPFILE 1 DS1:
#ADD SWAPFILE 3 DB1:SWAP3
#
```

The lowest numbered swap files are added (or installed) on the fastest devices, units 0 and 1 of the RS03 or RS04 disk. The highest numbered file is added on the slowest device, an RP04, RP05, or RP06 unit 1. Note that when you specify a non-file-structured device in the ADD SWAPFILE command, any existing file structure on that device is destroyed.

The /SI:n switch of the ADD SWAPFILE command allows on-line creation of a swap file during timesharing. The switch is most useful when you want to add a swap file to a private disk that contains sufficient contiguous space for the file. The value n in the /SI:n switch represents the number of blocks in the file. UTILTY attempts to access the file with a file type of .SYS in account [0,1] on the associated device. If the file size differs from the size spec ified in the /SI:n switch, UTILTY prints a warning message in the format:

```
NOTE  --  THE SIZE OF file.SYS IS x NOT n.
```

The value x is the actual size and n is the value in the /SI:n switch.

If the file does not exist, the program attempts to create a contiguous file of the size specified. If there is not enough contiguous space available to create the file, UTILTY prints:

```
?No room for user on device
```

After you type the ADD SWAPFILE command, the system tries to install the particular file or device as the swap file. If the system encounters any errors, UTILTY prints an error message in the following format:

```
text  --  in ADD SWAPFILE
```

Table 7–5 lists the possible errors. If no errors occur, the system has installed the swap file successfully.

Table 7–5: ADD SWAPFILE Command Errors

Text and Meaning
?Account or device in use A non-file-structured disk is being added as a swap file, but the disk is currently mounted (that is, it is being used as a file-structured device).
?Can't find file or account A file with the name specified and with a .SYS file type does not exist in account [0,1] on the associated device.
?Device not available A non-file-structured disk is being added as a swap file, but the disk unit or its controller has been disabled. Use an initialization option to enable the unit or its controller.
?Device not file structured The device specified is not a disk.
?Disk pack is not mounted A file-structured disk is specified, but that disk is not currently mounted. Use the MOUNT command to logically mount the disk before adding the file.
?Illegal filename The file name given in the command contains characters other than alphabetic or numeric characters.
?Illegal value - n The swap file number specified is not 0, 1, or 3. (The swap file 2 already exists on the system disk and need not be added.)
?Missing device or file name The command must contain either a device specification or a device and file name specification. If you use the /SI:n switch, a file name must be present.
?No room for user on device A file (on a file-structured device) is being added as a swap file, but the file is not large enough to store even one job at the current SWAP MAX.
?Not a valid device The device specified is a disk but is not configured on this system.
?Protection violation A file is being added as a swap file. Either the disk is mounted read-only or the file is bad (is not contiguous or is currently open). The system must have write-access to the device and file to add the file successfully.

You can then activate the added swap file with the LOGINS command. The LOGINS command forces the system to examine all swap file space and to calculate the number of jobs the system is capable of running.

During timesharing, the number of jobs that can run on the system depends on:

1. The job maximum (JOB MAX) set at the start of timesharing
2. The amount of swap space added
3. The number of logins allowed through the LOGINS, SET LOGINS, and NO LOGINS commands

At the start of timesharing, the number of logins allowed is one. One job is necessary for INIT to run. You can increase this number by using the LOGINS command, but the number is restricted by the swap space available and by JOB MAX. When you add swap space with ADD SWAPFILE commands, the capacity to handle logins is increased. The LOGINS command sets the number of logins to the maximum allowed. This maximum cannot exceed JOB MAX.

UTILTY lets you dynamically remove as well as add swap files 0, 1, and 3. To remove a swap file or device, first decrease the number of logins using the NO LOGINS or SET LOGINS command. The number of logins to be decreased depends on the capacity of the swap file to be removed. By decreasing the number of logins, you allow the system to safely remove swap space.

The sequence of commands to remove a swap file is the reverse of the sequence to add swap space. For example, assume that a system is running with a JOB MAX of 63 and is using a fixed-head disk (non-file-structured) as swap file 1. The disk is capable of swapping 16 jobs with a swap size of 16K words. You must, therefore, decrease the number of logins by 16 jobs before removing the device. The following sequence shows this procedure:

```
*SET LOGINS 47
*REMOVE SWAPFILE 1
*
```

In practice, it may take some time for the system to adjust operations. If the number of jobs currently running is greater than 47 or if the number of logins allowed is too large, the program prints the error message:

```
?Protection violation
```

You either must wait until users log off the system or you can decrease the number of users that can log in to the system (SET LOGINS command). The SYSTAT or VT50PY program prints the number of jobs currently on the system in the free buffer status report.

When you use the REMOVE SWAPFILE command and one or more swapped out jobs are in the swap space, UTILTY sends the error condition to the system and prints the warning message:

```
?Account or device in use
```

The system then locks the file and begins swapping jobs to other space. You must use the SYSTAT program at this point to make sure there are no jobs occupying the swap file. When you are satisfied the file is empty, use the REMOVE SWAPFILE command again to remove the swap file. The command should remove the swap file successfully on this second attempt. (The SHUTUP program automatically removes swap files.)

It is possible to have more swap space added than the system can use. In this case, you do not need to reduce logins to remove a swap file. However, enough room must remain to swap all jobs currently logged in or to swap all jobs that can log in, whichever is greater.

The LIST SWAPFILE command provides a way for you to verify that the following files exist on your system:

1. Swap files
2. Overlay file
3. Error file
4. DECnet/E Network Services Protocol (NSP) system file

The command is most useful when you want to verify the addition or removal of any of these files. After adding a file, it is reassuring to use the LIST SWAPFILE command to see, for example, that you gave the file the correct file specification or that the file was added properly. If you add a swap file, such as DR3:[0,1]SWAP0.SYS, run UTILTY and use the LIST SWAPFILE command as follows:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#LIST SWAPFILE

Swapfile 0:      DR3:[0,1]SWAP0 .SYS
Swapfile 1:      DR1:[0,1]SWAP1 .SYS
Swapfile 2:      DR1:[0,1]SWAP  .SYS
Swapfile 3:      DR4:[0,1]SWAP3 .SYS

Overlay file:    None
Error file:      None

DECnet/E file:   SY:[0,1]NSP0   .SYS
#
```

UTILTY prints NONE beside the files that do not exist. Note that UTILTY cannot add or remove the DECnet/E file but can only verify that the file exists on your system. There is no UTILTY command to add or remove this file. You must refer to the *DECnet/E System Manager's Guide* for further information.

7.1.9.2 Adding and Removing Overlay and Error Files – You can add and remove the overlay and error message files during timesharing. OVR and ERR are standardized names to denote the separate file for the overlay code and the alternate file for the default error message file. Although you can

use other file names, it is highly recommended that you keep the standardized ones. Regardless of the names you give the files, however, the system requires you to use a file type of .SYS for them.

The ADD OVERLAY command directs the system to use the specified file for the system overlay code. Similarly, the ADD ERROR command causes the system to use the specified file when it reads error messages. To add these system files, specify the commands as in the following example:

```
#ADD OVERLAY DS0:OVR
#ADD ERROR DB0:ERR
```

For the overlay file, the system copies the overlay code from the monitor SIL to the file OVR.SYS on RS03 or RS04 unit 0. For the error message file, the system copies into ERR.SYS on RP04, RP05, or RP06 unit 0 the contents of the error message file established by the DEFAULT initialization option. Note that you must format, initialize, and mount the disk on which an overlay or error file resides.

To add these files, the system ensures that the files exist and are properly formed. If any errors are encountered, UTILTY prints them in the format:

```
text   --   in ADD OVERLAY
text   --   In ADD ERROR
```

Table 7–6 lists the text of these errors. If no errors occur, the system has successfully added the file and is accessing it. You do not need to enter any further commands.

Remove these files at any time with the REMOVE OVERLAY and REMOVE ERROR commands. The system thereafter accesses the original data. The SHUTUP program automatically removes both files.

7.1.9.3 Using the SNAP Command – The SNAP command allows privileged users to take an on-line dump of the current monitor image executing in memory. After you execute the SNAP command, UTILTY copies the memory to the crash dump file CRASH.SYS in account [0,1]. Use the command as follows:

```
$ RUN $UTILTY
UTILTY V8 RSTS V8 TIMESHARING
#SNAP
#
```

Once the copy operation is complete, you can use the ANALYS program described in Chapter 6 to analyze the contents of the file.

The program may return the following error message if crash dump was not enabled:

```
?Can't find file or account
```

To enable crash dump, you must answer YES to the CRASH DUMP question during system generation.

Table 7–6: ADD OVERLAY and ADD ERROR Command Errors

Text and Meaning
`?Can't find file or account` The file with the name given and with a file type of .SYS does not exist in account [0,1] on the related device.
`?Device not file structured` The device specified is not a disk.
`?Disk pack is not mounted` The disk specified is not mounted.
`?Illegal filename` A name for the file is required and has not been specified, or the name specified contains illegal characters.
`?Missing device or file name` The command must contain either a device specification or a device and file name specification.
`?Name or account now exists` The file being added is already installed and operational.
`?No room for user on device` The file being added is not long enough. (The overlay file must be at least 32 blocks and the error file must be at least 16 blocks.)
`?Not a valid device` The device specified is a disk but is not configured on this system.
`?Protection violation` Either the disk specified is mounted read-only or the file is bad (is not contiguous or is currently open). The system must have write access to the device and file to successfully add it as a system file.

If a nonprivileged user attempts to dump memory with SNAP, the program prints:

```
?Protection violation
```

Only privileged users have access to UTILTY commands.

Use of the SNAP command may also return various device dependent errors, such as "?Device hung" or "?Disk pack is not mounted".

7.2 Recording System Activities — EMT Logging

An EMT is a PDP–11 assembly language instruction through which a program requests services from the monitor (such as opening and closing a file, doing I/O, and logging in and out). EMT logging is an optional feature that provides a "window" on the process by which timesharing jobs request, and receive, services from the RSTS/E monitor. Thus, EMT logging lets you gather information about the activity on your system.

For example, you might want to know the number of logins on a particular terminal, how many files are accessed on a certain drive, or which nonresident FIP overlays get the heaviest use. Such information can help you "tune" a system for improved performance, identify bottlenecks, establish charging algorithms, and watch for potential security problems.

This section describes the EMT logging feature, how it works, and how to include it in your system. Because of the wide variety of RSTS/E systems, however, you must decide which data are most useful in your environment, and how to collect and use the data the EMT logger provides.

EMT logging includes optional code in the monitor, uses XBUF to pass information, and requires an EMT logging program, which you must write, running as a timesharing job. You should be aware, therefore, that the use of EMT logging can affect performance. This effect depends on which EMTs you decide to log, for which jobs you log them, and how much processing your logging program does for each EMT.

EMT logging provides information on timesharing activity in terms of what the monitor sees. The data returned to your logging program is in terms of FIRQB and XRB contents, regardless of the programming environment of the job that issues the directives. See the *RSTS/E System Directives Manual* for information on the FIRQB and XRB, as well as descriptions of the MACRO form of system directives.

7.2.1 Programming for the EMT Logger

To use EMT logging, you must:

1. Include optional code in your monitor at system generation time.
2. Write a program to process the data extracted by the monitor code. This program retrieves extracted data by send/receive calls.

One question you are asked during the system generation dialogue is "EMT Logging?". You respond with "YES" to create a monitor that can extract EMT information and pass it to your program.

Your logging program will be a normal time-sharing job. You can design the program to do such things as select pertinent data, maintain a log file, signal events on one or more terminals, or control your system while it is running.

EMTs are never logged for certain jobs, including the Error Logger (ERRCPY) and your EMT logging program itself. In addition, the monitor does not allow certain EMTs to be logged. See the *RSTS/E Release Notes* for more information.

7.2.2 How EMT Logging Works

If EMT logging is available and active, the monitor inspects each EMT as it is received. ("Available" means generated into the system; "active" means that your EMT logging program has properly declared itself as a receiver.) If

the newly-received EMT is to be logged, the monitor builds a "packet" of information about the directive and stores the packet in XBUF. (EMT logging does not use small buffers for message transfer.)

The packet contains:

1. Context data, such as date and time, user job number, and keyboard number
2. Directive data, such as information from the job's FIRQB and/or XRB

When the directive is completed, additional information (such as the RSTS/E "error code") is added to the packet. The packet is then made available to your program. Your program can inspect the parameters passed from the user's job to the monitor, determine the relevance of the directive being logged from the packet, and take the appropriate action.

Your program retrieves EMT logging packets by issuing message receive calls. Parameters in your program's declare receiver call specify how many packets constitute a message, how many packets can be outstanding at any time, and how much XBUF the EMT logger can use. Each message received by your logging program consists of the packets – one per selected EMT – plus control information. This control information includes a count of EMTs that may have been "missed" because one or more of the limits you set up at declare time has been exceeded.

The *RSTS/E Programming Manual* describes the use of message send/receive calls and contains specific information about the send/receive calls used for EMT logging. The *RSTS/E Release Notes* contain additional information on the implementation of EMT logging in the current release of RSTS/E.

7.2.3 Data Returned by EMT Logging

The meaning of the data returned by the logging program you write depends on the internal functioning of the monitor. For this reason, both the format and meaning of data returned are subject to change in future releases of RSTS/E. DIGITAL reserves the right to change internal mechanisms, and therefore makes no commitment to continue providing any specific part of the data described for this release of RSTS/E.

7.2.4 EMT Logging and System Security

The information that EMT logging provides may be helpful in checking and maintaining system security. However, the use of EMT logging as a system security tool depends on many site-specific factors: the types of events that can be watched for, the degree of security required, the way that EMT logging is set up, the experience and judgment of the system manager who sets up EMT logging and interprets its data, and the site's general security practices.

In summary, although EMT logging does not guarantee improved system security, it can provide a very useful tracking mechanism.

7.3 Monitoring System Status — SYSTAT

During normal timesharing, there are many occasions on which you need to monitor the status of your system. RSTS/E provides two system programs for this purpose:

VT50PY — Automatically displays system status information on your terminal. By default, the program displays the system status every 15 seconds but you can specify a different interval. Refer to Section 7.3 for a description of the VT50PY program.

SYSTAT — Displays the status of different parts of the system when you use various program switches. See the *RSTS/E System User's Guide* for a complete description of SYSTAT and the switches associated with it.

The description of SYSTAT in this section gives a set of guidelines for using the SYSTAT program for more effective system management. Use SYSTAT to:

- Prepare for system shutdown to learn what jobs are active and which disk devices and assignable devices are in use.
- Determine when the number of general small buffers becomes too low. You can then decide whether to use the UTILTY commands NO LOGINS and SET LOGINS to restrict the number of users that can log in to the system.
- Check on the amount of free disk space. The disk status report in SYSTAT reflects the apparent number of free blocks on each disk on the system. For practical purposes, however, such as for allocating a file on the device, all free blocks that SYSTAT reports may not be usable. You may get the "?No room for user on device" message when SYSTAT reports there are free blocks available.

  The file cluster size or the number of clusters you need can prevent a file from fitting on a device. For example, a file whose cluster size is 16 and whose length is 10 blocks may not fit on a device that SYSTAT reports to have 50 free blocks of file space remaining. The cluster size of 16 demands that 16 contiguous blocks of free space exists on the device before the file can be allocated to the device. In some cases, 16 contiguous blocks do not exist on a device and RSTS/E does not allow a file to extend to another physical device even though SYSTAT can give you some indication of the availability of free disk space.
- Follow the progress of user jobs. You can determine if a job is stalled, waiting for resources on your system. If you notice that a RUN-TIME value of a job is not increasing (the value is printed in a job status report), it indicates that the job is stalled, waiting for an I/O device.

One user job can assign (ASSIGN) a device or keep an assignable device locked by having one file open. You can determine who the user is by examining the device status report that associates the busy device with the job number of the user controlling that device. You can then ask the user to free the device or, if that is not possible, you can use UTILTY commands to force the job off the system or seize the device with the SEIZE command.

- Follow the progress of detached jobs. When SYSTAT reports a detached job in the HB (hibernate) state, you know the detached job is attempting to communicate with its terminal. In this case, log in to the system at a free terminal, and then use the attach facility of LOGIN to attach the job to the terminal. Once you attach to the detached job, messages from the job can be printed.
- Check on the number of jobs on the system. The free status report lists the number of jobs currently logged in to the system and the maximum number of jobs allowed. These numbers are useful when you are adding and removing swap files. After adding the swap files and raising the logins allowed, you can check the free status report to confirm the success of the procedure. Before removing a swap file, check this report to make sure that the number of logins allowed is low enough to enable you to remove the swap file at some later time.

7.4 Dynamic Display of System Status — VT50PY

The VT50PY system program displays the system status on VT52 and VT100 terminals. After you start the program and decide how frequently you want it to update the system status on the screen, the program runs until you interrupt its execution. The information VT50PY prints is similar to that of the SYSTAT program. Commands you type during the execution of the program can alter the information the program displays on the screen. This lets you choose only those portions of the system status you need to see.

The program resides in the system library account [1,2] with the protection code of <232>. This allows users with non-privileged accounts to use the program. If you want to restrict the program to privileged users only, change the protection code to <124>. Generally, it is good practice to limit its use. Besides requiring a large amount of CPU time, it needs a job size of 16K words and thus should be run only on systems with sufficient memory.

7.4.1 Running the VT50PY Program

The program prints an identification line and a question after you type the command line RUN $VT50PY. They appear as follows:

```
$ RUN $VT50PY
VT50PY V8 RSTS V8 TIMESHARING
Interval?
```

If you want updates to occur every 15 seconds, press the RETURN key to accept the default. Otherwise, type the number of seconds you want to elapse between screen updates. You can include in your response any combination of the switches in Table 7–7. If you select a program switch but do not specify an interval, the program again assumes the default and updates the screen every 15 seconds.

Table 7–7: Display Program Switches

Program Switches and Meaning
/DET Detaches the job from this terminal or from the terminal specified. /KBn: Prints the output at keyboard unit n if it is available. If you specify the /DET switch, the program runs detached. /PRIORITY Runs the program at a special priority rather than at the normal –8 priority. /ECHO* Performs output on the terminal by using echo control features. Thus, the program disables echoing while the program updates the screen and spurious characters do not ruin the screen display.
*Echo control is an optional feature of the RSTS/E monitor and may not be available on your system.

When you include the /DET switch in response to the INTERVAL question to run the program detached, you interrupt execution so that you can use the terminal for other work. As soon as you type CTRL/C to interrupt the display, the program prints a message telling you the terminal is available. When you release the terminal by logging off, the program automatically displays the status information on the screen again, as if it had not been interrupted.

You stop the execution of the VT50PY program with CTRL/Z. The procedure you follow after stopping the program depends on whether you were running the program attached or detached. If you were running the program attached, typing CTRL/C to end the program returns control to your keyboard monitor. When running the program detached, typing CTRL/Z causes the program to print the following message before terminating the program:

```
THE TERMINAL IS ALL YOURS NOW
```

You must then attach the job running the program to the terminal as follows:

```
^C

                THE TERMINAL IS ALL YOURS NOW
```

(continued on next page)

```
HELLO 1,2 (RET)
PASSWORD:
Job 9 is detached under this account.
Job number to attach to? (RET)

$
```

When the program becomes attached to the terminal as a result of the ATT command, it prints the CONTINUE question. Finally, type NO or any other string not beginning with Y and press RETURN to end the program.

The program displays on your terminal an identification line at the top line of the screen, skips a line, and fills the left portion of the screen with job status information and the right side with information about:

- Busy devices
- Disk structure
- Run-time systems
- Message receiver statistics
- Free buffer status
- Resident library statistics

After the screen is full, the program moves the cursor to the first character on the second line of the screen. The program is then idle, waiting to cycle through the display again.

At the specified interval, the program checks the system tables and updates the status information on the screen with any changed data. While executing routines to extract update information, the program prints the message:

```
WORKING...
```

It leaves the cursor to the right of the message. After the update is complete, the cursor returns to its idle position.

While the cursor is in the idle position, you can type commands to modify the contents and arrangement of items on the screen. End any command with the ESCAPE key. Although any line terminator works, ESCAPE leaves the cursor positioned on the blank line. The program prints no message or takes no action on invalid statements.

Some commands that add items to the screen take the minus (–) sign as a prefix. A command with a minus sign preceeding it causes VT50PY to delete rather than add a display of the command that would occur without the minus sign. That is, the minus sign negates the effect of the command. Table 7–8 contains the commands you enter on the screen (usually while the cursor is in its idle position) to delete or add system status information.

Table 7–8: Display Program Commands

Command Type	Format and Description
General	*C* Clears the screen and displays new status. *Sn* Displays memory status in place of job status. Starts with the 8K-word section less than or equal to n. If n is not given, starts at the beginning of memory. *J* Displays job status in the standard manner. *Jn* Displays job status starting with active job n + 1. Overcomes physical limitation of the screen. *Xn* Changes the interval to n seconds. *X0* Updates the display with an interval of 0 seconds (that is, runs continuously) but lowers the priority so that other jobs are not stalled.
Job Status	*O* Displays the account number of operator jobs as [OPR]. An operator job has a project number 1 and a programmer number less than 200. *–O* Replaces OPR in operator account designations with the actual project and programmer numbers. *T* Displays total CPU time each job has used. The time is displayed as number of hours, minutes, seconds, and tenths of seconds under the RUN–TIME column. + Displays the increment of CPU time each job has used since the display program last updated the screen. User can return to total CPU time by typing T. % Displays the amount of CPU time each job has used as a percent of the total CPU time expended. The user can return to total or increment of CPU time by typing, respectively, T or +. *J–O* Does not display operator jobs. Operator jobs are those running under project number 1 and programmer number less than 200. *JO* Displays only operator jobs. *J+O* Includes operator jobs in display. *J–D* Does not display detached jobs. *J+D* Includes detached jobs in display.

(continued on next page)

Table 7–8: Display Program Commands (Cont.)

Command Type	Format and Description
Job Status (Cont.)	*JD* Displays only detached jobs. *J–S* Does not display sleeping jobs. *J+S* Includes sleeping jobs in display. *N* Indicates program name in the WHAT column. *–N* Removes program name from the WHAT column and replaces it with name of the RTS under which job is running. *P* Indicates priority of jobs more exactly than plus (+) and minus (-) characters. *–P* Indicates the priority of jobs with the plus (+) character for higher than normal priority and the minus (-) character for lower than normal priority. *W* Indicates under the STATE column, the last WAIT state rather than actual state. *–W* Removes last WAIT state and indicates actual state of each job. *K* Displays, under the SIZE column, the amount of memory occupied by each job. *–K* Displays, under the SIZE column, the amount of memory remaining to each job.
Disk Structure	*D, Dn, –D* Displays disk structure statistics. If n is 1, places item first on the screen. A preceding minus sign removes the disk structure statistics from the screen. *L, –L* Displays, under the COMMENTS column, the logical name of each device. The preceding minus sign replaces logical names with standard PUB, PRI, NFS, or LCK notations.
Busy Devices	*B, Bn, –B* Displays busy device statistics. If n is 1, the program places the statistics first on the screen. A preceding minus sign removes busy device statistics from the screen.
Free Buffer	*F, Fn, –F* Displays free buffer statistics. If n is 1, the program places the statistics first on the screen. A preceding minus sign removes free buffer statistics from the screen.

(continued on next page)

Table 7–8: Display Program Commands (Cont.)

Command Type	Format and Description
Message	*M, Mn, –M* Displays message receiver statistics. If n is 1, the program places the statistics first on the screen. A preceding minus sign removes message receiver statistics from screen.
Run-Time System	*R, Rn, –R* Displays run-time system data. If n is 1, the program places the data first on the screen. A preceding minus sign removes run-time data from the screen.
Resident Libraries	*H, Hn, –H* Displays resident library data. If n is 1, the program places the data first on the screen. A preceeding by a minus sign removes same from the screen.

7.4.2 Screen Layout

The program partitions the screen into four major sections:

Header line	Contains the RSTS/E version number, the name of the system, the current date and time of day, and the number of hours, minutes, and seconds since the start of time-sharing operations (termed "up-time"). The header line appears on the first line of the display.
Total statistics line	Tells the percentage of time that is expended by users (User), input and output processing (I/O), and the monitor (Exec) as well as the amount of idle time (Idle) and lost time (Lost). The statistics print on the line below the header line but are replaced at the interval you specify by the WORKING... message.
Lefthand portion	Contains either the job status statistics or, if you use the L command, the memory usage status.
Righthand portion	Contains statistics for busy devices, the disk structure, run-time systems, message receivers, system buffers, and resident libraries.

All four partitions are displayed in the following example. While the example does not represent an actual log of the VT50PY program, it does give you a picture of how information is displayed. When you run the program on your system, not all of the information that is illustrated can fit on your terminal screen. You will often need to use commands described in Table 7–8 to delete reports so that unseen portions of the display become visible. This example allows you to see an entire display, unhindered by screen limitations (usually 24 rows).

```
RSTS V8     Timesharing     Status on 15-Aug-82 05:31 PM  Up: 41:55:19
    78.0%User      5.4%I/O        6.0%Exec,     10.0%Idle,        1.6%Lost
 Job  Who   Where  What Size State Run-time Pr       Busy  devices
  1  [OPR]  Det    ERRCPY  5 SR              +  Dev Job  Why   Dev Job Why
  2  [OPR]  Det    OPSRUN 16 SL           ,8     PK0  11  A+Op PK1   8 Open
  3  [OPR]  Det    QUMRUN 16 SL          3.4  +
  4  [OPR]  Det   SPLIDL 16 SL                        Disk Structure
  5  [OPR]  Det   BATIDL 13 SL D25          DK0  0   1524  4   0 Pri,R-O
  6  [OPR]  Det   BATIDL 13 SL D26          DR0  0  19772  4   0 Pri,DLW
  7  [OPR]  Det   EVTLOG 22 SL              DR1 43   1720  4   2 Pub,DLW
  8  [OPR]  Det   NPKDVR  9 SL              DR2  0  12496  4   0 Pri,R-O
  9  [OPR]  Det   SYSMAN  8 SL              DR3  9   9720  8   3 Pri,DLW
 11   1,210 KB32  ATPK    8 SL              DR4  1   3204  4   1 Pri,DLW
 12   1,210 POJ11 VT50PY 16 RN Lck
 13   1,223 KB51  SYSTAT 12 ^C        1,8
 15  [OPR]  KB31  ...EDT  7 KB                  Run-time systems
 16   1,226 KB17  NONAME  2 ^C          BAS4F   15K   9 Prm,KBM,CSZ
 20   1,253 KB42  VTEDIT 14 KB          RT11     4K   1 Tmp,KBM,CZR
 21   7,214 KB21  VTEDIT 24 RN          TECO     8K   0 Tmp
 22   1,248 KB29  PIP    16 KB          BAS4AL  16K   0 Non-res,KBM
 23 226,0   P1J8  NONAME  2 ^C A09      BASIC   15K   0 Tmp,KBM,CSZ
 24   1,250 KB33  ONLPAT 15 KB
 26   1,243 KB43  NONAME  2 ^C
 27   1,247 KB30  NET    16 SL              Message receivers
 28   8,254 KB72  ...EDT  7 HB          ERRLOG(Prv)     1  0     0/40
                                         OPSER (Loc)     2  0     0/30
                                         QUEMAN(Loc)     3  0     0/60
                                         LPOSPL(Prv)     4  0     0/5
                                         BA1SPL(Prv)     5  0     0/5
                                         EVTLSN(Prv,Nt)  7  0     0/16
                                         EVTLOG(Prv)     7  1     0/32
                                         NWPK08(Nt)      8  0     0/16
                                         NWTT12(Nt)     12  0     0/5

                                          Gen FIP  Jobs  TTY  Err
                                          315  62 14/50    0    3

                                            Resident Libraries
                                         BASICS   8K   0 Non-res,Rem
                                         EDT     21K   2 Tmp,Rem
                                         RMSRES   4K   1 Non-res,Rem
                                         RMSLBA   4K   0 Non-res,Rem
                                         RMSLBB   4K   1 Non-res,Rem
                                         RMSLBC   3K   0 Non-res,Rem
                                         RMSLBD   2K   0 Non-res,Rem
                                         RMSLBE   4K   0 Non-res,Rem
                                         RMSLBF   4K   0 Non-res,Rem
                                         DAPRES  10K   0 Non-res,Rem
```

The VT50PY program prints information similar to the information SYSTAT includes in its status reports. The description of the major sections of the dynamic status report includes information about how the VT50PY program reports differ from that of the SYSTAT program. If you need a more complete description of SYSTAT, refer to the *RSTS/E System User's Guide.*

7.4.2.1 Job Status Statistics – The job status report displays information in eight separate columns on the lefthand portion of your terminal. The report is similar to the information the SYSTAT program prints in its display of job status. Unlike the SYSTAT program, the job status report does not include the name of the run-time system under which the job is running but does include abbreviations indicating the priority of the running job. The titles

that the VT50PY program prints at the top of a job status report have the following meaning:

Job	Job number that the system assigns when the job starts timesharing activities.
Who	Account number under which each job runs.
Where	Keyboard number of the job. DET appears in place of the keyboard number for jobs that run detached from the keyboard. The abbreviation PxJy can appear for a job running on a pseudo keyboard. The value Px identifies pseudo keyboard unit x; and the value Jy denotes job number y, under which the controlling job is running.
What	Program name that the job is executing.
Size	Current size in K words of the job.
State	Current state of the job indicated by the set of abbreviations in Table 7–9.
Run-time	Hours, minutes, seconds, and tenths of seconds of central processor (CPU) time the job has consumed.

The job status report includes a PR column, which is not displayed by SYSTAT, that identifies the priority of the running job. The PR column can display the following abbreviations:

If –P is in effect:

+	Higher than normal priority
–	Lower than normal priority
S	Special run priority
^	CTRL/C temporary priority
K	Keyboard delimiter temporary priority

If P is in effect:

+n	Positive priority n * 8
0	Zero priority
–n	Negative priority n * 8

Table 7–9 contains a brief description of the abbreviations that you may see in the STATE column of the job status report.

The following status descriptions may appear after one or more of the job status abbreviations in Table 7–9:

Lck	Job is locked in memory for the current operation.
Nsw	Job has requested that it not be swapped from memory and cannot be swapped unless it requests additional memory.
Swi	Job is currently being swapped into memory.
Swo	Job is currently being swapped out of memory.
Xnn	Job is swapped out and occupies slot nn in swap file X; file is denoted A, B, C, D to represent files 0 through 3 of the swap structure.

Table 7–9: STATUS Column Abbreviations

Abbreviation	Meaning
,	Job is not logged in to the system.
??	Job's state cannot be determined.
BF	Job is waiting for buffers (no space is available for I/O buffers).
^C	Job is in CTRL/C state, awaiting keyboard monitor input.
CR	Job is waiting for card reader input.
DET	Job is detached from all terminals.
DK,DM,DB,DS, DP,DL,DF,DR, DU	Job is waiting to perform disk I/O.
DT or DD	Job is waiting for DECtape I/O.
DX	Job is waiting for floppy diskette I/O.
FP	Job is waiting for file processing action by the system (opening or closing a file, file search).
HB	Job is detached and waiting to perform I/O to or from a terminal.
KB	Job is waiting for input from a terminal.
LP	Job is waiting to perform line printer output.
MT,MM, or MS	Job is waiting for magnetic tape I/O.
OPR	Job runs under a system operator account.
PP	Job is waiting to perform output on the high-speed paper tape punch.
PR	Job is waiting for input from the high-speed paper tape reader.
RJ	Job is waiting for RJ2780 I/O.
RN	Job is running or waiting to run.
RS	Job is waiting for residency.
SELF	Job runs under your account.
SL	Job is sleeping.
SR	Job is sleeping and is a message receiver.
TT	Job is waiting to perform output to a terminal.

7.4.2.2 Busy Devices – The busy device report lists the devices that are assigned or opened by a specific user. Items reported are the device specification, the job owning that device, and the condition of the device. The disk status information reports assigned disk units. The busy device report prints on the righthand side of the screen and looks as follows:

```
Dev Job Why  Dev Job Why
PK0  11 Open PK1   8 Open
```

The VT50PY program reports the same information as SYSTAT reports in its busy device report, except that VT50PY prints two reports side-by-side to accommodate the limited screen space. Table 7–10 contains the abbreviations that can appear in the WHY column of the busy device report.

Table 7–10: Busy Device Status Abbreviations – WHY Column

Abbreviations	Meaning
AS	Device is explicitly assigned to a job.
INIT	Device is open on a channel.
DOS	Magnetic tape is assigned with DOS labeling format.
ANSI	Magnetic tape is assigned with ANSI standard labeling format.

7.4.2.3 Disk Structure – The disk structure report describes each disk in use on the system. The report contains the same information displayed by the /D switch of SYSTAT. The VT50PY program prints the following information but because of screen limitations does not print column headings as SYSTAT does:

- Disk device specification
- Number of open files
- Number of free 512-byte blocks
- Pack cluster size
- Disk hardware error count
- Comments on the status of the disk
- Pack identification name or system logical names (if any) assigned for the devices in use on the system

The disk structure report appears on the righthand side of your terminal and looks as follows:

```
      Disk Structure
DR0   0   17252   4    0 Pri,DLW
DR1  42    3044   4    2 Pub,DLW
DR2   0   12496   4    0 Pri,R-O
DR3   5   32144   8    3 Pri,DLW
DR4   1    3272   4    1 Pri,DLW
DR5   0 532032  16    6 Pri,R-O
```

To display the pack identification names or system logical names, you must use the L command described in Table 7–8. The command displays the disk label information in place of the comments, which the program normally displays. For example:

```
      Disk Structure
DR0   0   17252   4    0 R
DR1  42    3044   4    2 SYS
DR2   0   12496   4    0 H
DR3   5   32144   8    3 W
DR4   1    3272   4    1 M
DR5   0 532032  16    6 ...DR5
```

If you want to display the comments again, use the -L command. The limited size of the screen makes using the L and -L commands necessary. Refer to Table 7–11 when you have questions about the abbreviations listed in comment column of the disk structure report.

Table 7–11: Disk Status Abbreviations

Abbreviations	Meaning
Pub	Cartridge or pack is public.
Pri	Cartridge or pack is private.
NFS	Disk is open as a non-file-structured device.
R-O	Disk unit is read-only (write-locked)
DLW	Date of last write (modify), rather than date of last access, is stored in file accounting entries.
Lck	Disk is in a locked state.
NFF	New files on this disk are put at the beginning of the directory.

7.4.2.4 Message Receiver Statistics – The message receiver report:

- Provides the job name of the receiving job
- Provides the job number of the receiving job
- Includes the number of messages queued for the job
- Gives the declared maximum number of messages the job can queue
- Tells whether local and network senders are allowed
- Indicates whether local senders must be privileged

The message receiver report appears on the righthand portion of the screen and often does not appear until you delete the disk and run-time system reports with the -D and -R commands, respectively. A sample of the message receiver report is:

```
Message receivers
ERRLOG(Prv)      1   0      0/40
OPSER (Loc)      2   0      0/30
QUEMAN(Loc)      3   0      0/60
LPOSPL(Prv)      4   0      0/5
BA0SPL(Prv)      5   0      0/5
BA1SPL(Prv)      6   0      0/5
EVTLSN(Prv,Nt)   7   0      0/16
EVTLOG(Prv)      7   1      0/32
```

Due to limited space on the screen, some of the abbreviations are shorter than those printed by the /M switch of SYSTAT. The receiver report does not include the:

- Receiver identification block
- Object type
- Number of links used and the maximum links allowed.

Table 7–12 describes the abbreviations found in the message receiver status report.

Table 7–12: Message Receiver Abbreviations

Abbreviations	Meaning
Loc	Local senders are allowed for this receiver ID.
Prv	Local senders must be privileged to send to this receiver ID.
Nt	Network senders are allowed for this receiver ID.
1S	Receiver can handle one and only one link.
N1	Both of the above.

7.4.2.5 Free Buffer Status – The free buffer statistics provide the following information:

- Number of general small (16-word) buffers not currently in use
- Number of FIP buffers not currently in use
- Number of jobs currently running
- Maximum number of jobs allowed to run
- Number of disabled terminals
- Total number of errors logged on the system

The report prints the same information the F switch of the SYSTAT program produces. An example of a free buffer report is:

```
Gen FIP  Jobs  TTY  Err
183   1 14/50    0  259
```

7.4.2.6 Run-Time System Statistics – The statistics for run-time systems are displayed on the righthand portion of the screen. The report includes the same information as printed by the /R switch of the SYSTAT program, except:

- The maximum size in K words that a job running under the run-time system can take
- Some information in the comments column because of limited screen space

The run-time system report prints:

- Name of each run-time system
- Size of the run-time system in K words
- Number of user jobs currently executing under the control of the run-time system
- Comments regarding the status of the run-time system

An example of the run-time system report follows:

```
   Run-time systems
BAS4F    15K    9 Prm,KBM,CSZ
RT11      4K    1 Tmp,KBM,CZR
TECO      8K    0 Tmp
BAS4AL   16K    0 Non-res,KBM
BASIC    15K    0 Tmp,KBM,CSZ
```

Table 7–13 contains a list and description of each abbreviation that VT50PY prints in the comments column.

Table 7–13: Run-Time System and Resident Library Report Abbreviations

Abbreviations	Meaning
Non-res	Run-time system or library is non-resident.
Loading	Run-time system or library is being loaded into memory.
Tmp	Run-time system or library is removed from memory when not being used.
Prm	Run-time system or library stays in memory when not being used.
Addr:xxx	The value of xxx denotes the starting address of the run-time system or library.
KBM	Run-time system or library can serve as a keyboard monitor.
1US	Run-time system or library can serve only one user.
R/W	Run-time system or library allows read/write access.
NER	Errors occurring within the run-time system or library are not sent to the system error log.
Rem	Run-time system or library is removed from memory as soon as all its jobs switch to another run-time system or library.
CSZ	Proper job image size (in K words) to run a program can be computed as K-size = (filesize + 3)/4.
EMT:yyy	Denotes the EMT code for special EMT prefix.

7.4.2.7 Resident Library Statistics – The resident library information includes:

- Name of the resident library
- Protection code of the resident library
- Size of the resident library
- Number of user jobs currently executing under its control
- Comments regarding the status of the resident library

The statistics for resident libraries are displayed on the righthand portion of the screen. They print the same information as displayed by the /L switch of

SYSTAT. Some information in the comments column may be omitted due to limited space on the screen. Refer to Table 7–13 for a description of the abbreviations used in the comment column.

7.4.2.8 Memory Status – The S command causes the program to print a table that shows the use of each 1K-word portion of memory. The memory status report replaces the job status report on the lefthand half of the screen. Use the J command to display the job information again.

Type the S command to have VT50PY print the memory status report as in this example:

```
  Memory usage (Starting at 0K)
   0 MON   MON   MON   MON   MON   MON   MON
   8 MON   MON   MON   MON   MON   MON   MON
  16 MON   MON   MON   MON   MON   MON   MON
  24 MON   MON   MON   MON   MON   MON   MON
  32 MON   MON   MON   MON   MON   MON   MON
  40 MON   MON   MON   MON   MON   MON   MON
  48 MON   MON   MON   MON   MON   MON   MON
  56 *1*   *1*   *1*   *1*   *1*   *1*   *1*
  64 *1*   *1*   *1*   *1*   *1*   *1*   *1*
  72 EBP   EBP   EBP   EBP   EBP   EBP   EBP
  80 EBP   EBP   EBP   EBP   EBP   EBP   EBP
  88 EBP   EBP   EBP   EBP   EBP   EBP   EBP
  96 EBP   EBP   EBP   EBP   EBP   EBP   EBP
 104 EBP   EBP   EBP   EBP   EBP   EBP   EBP
 112 EBP   EBP   EBP   EBP   EBP   EBP   EBP
 120 EBP   EBP   EBP   EBP   EBP   EBP   EBP
 128 EBP   EBP   EBP   EBP   EBP   EBP   EBP
 136 EBP   EBP   EBP   EBP   EBP   EBP   EBP
 144 EBP   EBP   EBP   EBP   EBP   EBP     7
 152   7     7     7     7     7     7     7
 160   7     7     7     7     7     7     7
```

Due to limited screen space, the program may simultaneously display all memory status information. The Sn command allows the user to determine the starting 8K section. The display program prints the starting section number for the row and 8 abbreviations per row. Each of the abbreviations relates to the status of a 1K section of memory. The number of rows the program prints and the extent of memory covered is limited by the terminal. The program indicates the starting 1K section by printing a header line in the following format:

```
MEMORY USAGE (STARTING AT nK)
```

Table 7–14 contains a description of the abbreviations the memory status report uses.

7.5 Terminal and Remote Line Characteristics – TTYSET

The RSTS/E system can operate with a variety of terminals. During the generation of your system, you specify the number and types of terminal interfaces that are to be part of your hardware configuration. While you do

specify the terminal interfaces, you do not assign specific terminal characteristics. Because many terminals operate in teletype mode, the RSTS/E system automatically sets the default characteristics of all interfaces to certain standard values:

- Can produce hard-copy output
- Prints data up to 72 columns wide
- Receives data at a baud rate of 110

You must use the TTYSET program to set the characteristics of the terminals that do not have these standard default values.

Table 7–14: Memory Status Report Abbreviations

Abbreviations	Meaning
MON	Occupied by the RSTS/E monitor.
n	Occupied by run-time system n, where n is the position in the run-time system list. (A question mark is printed if the run-time system name cannot be determined.)
n	Occupied by job number n.
nLK	Job number is locked in this 1K portion.
nS	Job number n is being swapped out of memory.
nSI	Job number n is being swapped into memory.
NXM	Memory space is nonexistent.
END	End of physical memory for user jobs.
LCK	Memory is locked.
EBP	Memory is reserved for the extended buffer pool.

7.5.1 Introdution to TTYSET

The TTYSET program sets characteristics for terminals attached locally to a RSTS/E system or for terminals connected by remote lines. Users can run the program to set characteristics for their own terminals. Only privileged users have access to the TTYSET KBn: command to define characteristics for remote and local terminal lines. As system manager, you would generally place TTYSET commands in the START.CTL file to set the characteristics of system terminals. This standard procedure sets up local lines automatically at the start of each time-sharing session.

There are two methods for setting the characteristics of remote lines:

- Nonprivileged user sets characteristics. All remote lines start out with the standard default settings. Consequently, the user of a remote line must log in to the system at the standard speed of 110 baud, and then run TTYSET to set the characteristics of the terminal. If the

remote line is connected to a DH11 or DZ11 multiplexer line, the user can type the TTYSET SPEED command to change the baud rate. The terminal reverts to the standard settings when the user logs off.

- Privileged user (system manager) sets ring characteristics. The system manager can run TTYSET to set the ring characteristics. This causes the system to use automatically the characteristics the system manager sets, instead of the standard values, each time a user logs in on a particular remote line. By setting ring characteristics for a remote line, the system manager can establish certain lines for alphanumeric display terminals running at 1200 baud or for other types of terminals. The ring characteristics remain in effect for the current time-sharing session, unless the system manager issues a new TTYSET command.

7.5.2 Terminal Line Speed Characteristics File — TTYSET.SPD

There are two circumstances that require you to create a special terminal line speed characteristics file:

1. You have decided to restrict the valid line speeds at certain terminals to a subset of the line speeds allowed by the terminal interface.
2. You installed an interface that has been modified so that nonstandard line speeds have been substituted for one or more standard interface line speeds.

To begin, you must gather the information on the restrictions or modifications and the keyboard numbers of the terminals affected. Then, create a file named TTYSET.SPD with a text editor. After you include the information in the file, you must place the file in the system library account [1,2].

Each line of information you include in the TTYSET.SPD file is a series of ASCII entries, separated by commas. The entries specify the keyboard number and the line speeds for the terminals affected. You must place the entries in certain positions on the line. That is, the first entry must be the keyboard number, the second entry must be the first programmable baud rate allowed, the third entry must be the next baud rate allowed, and so on, to a maximum of 16 specified baud rates. Sixteen is the maximum number RSTS/E allows.

To restrict the valid line speeds of a terminal to a subset of the line speeds allowed by the interface:

1. Determine the line speeds allowed by the interface by referring to the hardware documentation for the interface.
2. Type the keyboard number, and then type the speeds you are to allow or –1 (for speeds you disallow) in the exact order presented in the hardware documentation.

As an example, assume a DH11 controls keyboard 11 (among others). The DH11 interface allows 14 baud rates plus two external inputs as shown in the following partial listing of the DH11 Speed Table:

Table 7–15: Speed Table for Receiver and Transmitter Speeds

Transmitter Receiver	Bits 13 9	12 8	11 7	10 6	
	0	0	0	0	Zero Baud
	0	0	0	1	50 Baud
	0	0	1	0	75 Baud
	0	0	1	1	110 Baud
	0	1	0	0	134.5 Baud
	0	1	0	1	150 Buud
	0	1	1	0	200 Baud
	0	1	1	1	300 Baud
	1	0	0	0	600 Baud
	1	0	0	1	1200 Baud
	1	0	1	0	1800 Baud
	1	0	1	1	2400 Baud
	1	1	0	0	4800 Baud
	1	1	0	1	9600 Baud
	1	1	1	0	External Input A
	1	1	1	1	External Input B

If you want to restrict valid line speeds for KB29 to 110, 300, and 1200 baud, the line in the TTYSET.SPD file must be:

29,–1,–1,–1,110,–1,–1,–1,300,–1,1200,–1,–1,–1,–1,–1,–1

You must place each entry in the correct position.

To specify the line speed modifications you have substituted:

1. Determine the positions in the speed table for the interface line(s) that have been affected by the nonstandard line speeds.
2. Type the keyboard number, and then the line speeds allowed by the interface, substituting the strings specifying the new speeds in the positions of the replaced line speeds.

Assume, for example, the use of a DH11 multiplexer and KB29. You have a nonstandard line speed set on the DH11 that affects the 2400 baud position. Replace the old 2400 baud rate with the new line speed of 2350. You create an entry in TTYSET.SPD for KB29 (and any other terminal on the DH11):

```
29,0,50,75,110,134.5,150,200,300,600,1200,1800,2350,4800,9600,-1,-1
```

7.5.3 TTYSET Privileged Feature — KBn: Command

You set the characteristics of other terminals in the RSTS/E system with the KBn: command. While logged in to the system under a privileged account, run the TTYSET program:

```
$ RUN $TTYSET
TTYSET V8 RSTS V8 TIMESHARING
Terminal characteristics program
?
```

The program prints a header line and a question mark (?) prompt. The program is now ready to accept commands. If you want to set the characteristics of a VT100 terminal at KB32, type the following:

```
? KB32:
FOR KB32:? VT100
FOR KB32:?
```

After you type the keyboard number and press the RETURN key, the program prints the FOR KB32:? prompt. All commands you type in response apply to the KB32 keyboard. In this case, the VT100 command immediately sets the characteristics of the line to those of a VT100 alphanumeric display terminal. TTYSET prints the prompt again to let you enter a KBn: command for another keyboard.

You can also change specific characteristics of a terminal. For instance, to limit the line length of the terminal at KB40, type:

```
FOR KB32:? KB40:
FOR KB40:? WIDTH 60
FOR KB40:? EXIT

$
```

Each time 60 characters print on KB40, the system performs a carriage return and line feed. EXIT ends the TTYSET program and returns you to your keyboard monitor.

Use the /RING switch to specify characteristics on a disabled terminal. For example, you can use /RING to set an initial WIDTH characteristic for a pseudo keyboard.

7.5.4 Automatic Setting of Terminal Characteristics

The setting of terminal characteristics on a RSTS/E system applies to the current time-sharing session. This occurs because, at system start-up, the system automatically sets the characteristics of all keyboard lines (except line number 0) to the standard values: hardcopy, 72 column, 110 baud. To establish characteristics that match the local hardware, you must run the

TTYSET program each time the system is initialized. Instead of setting the characteristics each time by hand, it is better to use the INIT system program to automatically set both local and remote terminal characteristics by commands in the START.CTL and CRASH.CTL files.

Assume, for example, that keyboard 1 is an LA36 terminal running at 300 baud and keyboard 2 is a VT100 running at a split speed of 300/9600. You create an indirect command file to properly set the characteristics by including the following commands:

```
DETACH
LOGIN KB0: [1,2]
FORCE KB0: RUN $TTYSET
FORCE KB0: KB1:;LA36;300
FORCE KB0: KB2:;VT52;SPLIT SPEED 300/9600
FORCE KB0: EXIT
FORCE KB0: BYE/F
ATTACH
```

The indirect command file detaches, INIT logs the system console terminal in to the system under a privileged account, forces the necessary sequence of TTYSET commands, logs the terminal off the system, and reattaches to INIT.

7.5.5 Setting Terminal Characteristics of Remote Lines — /RING

The /RING switch keeps you from having to set the characteristics each time you log in to the system on certain remote lines. For example, to set the characteristics of the remote line on keyboard 14 for the current time-sharing session, you run TTYSET under a privileged account and type the commands:

```
? KB14:/RING
FOR KB14:(RING)? LA36
FOR KB14:(RING)?
```

When you include the /RING switch with the KBn: command, TTYSET prints the KBn:(RING) message to prompt you for a valid command. The command takes effect immediately and TTYSET prints the message again to let you enter other commands. If you are not logged in to a privileged account or the characteristics of the line conflict with the command, TTYSET prints an error message and reprints the keyboard prompt. The program sets the characteristics of the line for the duration of the current time-sharing session if it does not detect any errors. Thereafter, whenever you dial the particular line, the system uses the characteristics that you set for that terminal.

NOTE

The DL11E-type interface and the individual local interfaces (KL11 and DL11A through DL11D) and the DJ11 multiplexer do not have programmable baud rates. For this reason, you cannot execute commands to change baud rates on a keyboard line having any of those interfaces.

7.5.6 Using the GAG and NOGAG Commands

The GAG command prevents your terminal or any terminal on the system from receiving broadcasts sent by the SEND command of UTILTY. This is useful when you need to print a document on a letter-quality printer and do not want messages corrupting the printout. It is equally helpful when you are using a text editor in screen mode at a remote terminal set at a low baud rate. Repainting the screen each time someone sends a message can be time consuming, especially at 300 baud. The NOGAG command reverses the gag state to again allow your terminal to receive messages.

Use the GAG command to keep messages from printing on your own terminal:

```
$ RUN $TTYSET
TTYSET V8 RSTS V8 TIMESHARING
Terminal characteristics program
? GAG
? LIST
Current settings:
                Tab             Form            LC output
                No XON          Local echo      No scope
                LC input        No stall        Uparrow
                No ESC seq      ESC             CTRL/R
                Resume any      Break           Gag
                No parity       No fill         Width 132
                No delimiter    Speed not settable
?
```

The LIST command lets you verify that the command worked and that the rest of the terminal settings are set properly. The program returns you to the question mark prompt to allow further input. If you decide to reverse this condition, use the NOGAG command:

```
$ RUN $TTYSET
TTYSET V8 RSTS V8 TIMESHARING
Terminal characteristics program
? NOGAG
? LIST
Current settings:
                Tab             No form         LC output
                XON             Full duplex     Scope
                LC input        Stall           Uparrow
                No ESC seq      ESC             CTRL/R
                Resume CTRL/C   Break           No Gag
                No parity       No fill         Width 132
                No delimiter    Speed 9600
?
```

Again, the program returns to the question mark prompt after executing the command. Your terminal can now receive SEND messages.

The TTYSET program also lets you use the GAG and NOGAG commands on other terminals on your system. This you do with the KBn: command as follows:

```
$ RUN $TTYSET
TTYSET V8 RSTS V8 TIMESHARING
Terminal characteristics program
? KB32:;GAG
For KB32:? LIST
Current settings:
                Tab             No form         LC output
                XON             Full duplex     Scope
                LC input        Stall           Uparrow
                No ESC seq      ESC             CTRL/R
                Resume CTRL/C   Break           Gag
                No parity       No fill         Width 80
                Delimiter "A", CHR$(1)          Speed 9600
For KB32:?
```

For proper use of the KBn: command, you must type KB, the keyboard number, a colon (:), a semicolon (;), the command (GAG), followed by the RETURN key. Keyboard number 32 can no longer receive SEND messages. If you have other terminals that you must set to the gag state, type the next keyboard number with the KBn: command in response to the FOR KB32:? prompt as follows:

```
For KB32:? KB40:;GAG
For KB40:? LIST
Current settings:
                Tab             No form         LC output
                XON             Full duplex     Scope
                LC input        Stall           Uparrow
                No ESC seq      ESC             CTRL/R
                Resume CTRL/C   Break           Gag
                No parity       No fill         Width 132
                No delimiter    Speed 9600
For KB40:?
```

You can set KB40 back to its original "nogag" state as follows:

```
For KB40:? NOGAG
For KB40:? LIST
Current settings:
                Tab             No form         LC output
                XON             Full duplex     Scope
                LC input        Stall           Uparrow
                No ESC seq      ESC             CTRL/R
                Resume CTRL/C   Break           No Gag
                No parity       No fill         Width 132
                No delimiter    Speed 9600
For KB40:?
```

As an added note, you use the LIST command with GAG or NOGAG if you separate the commands with a semicolon. The previous example could have been entered as follows:

```
For KB40:? NOGAG;LIST
Current settings:
                Tab             No form         LC output
                XON             Full duplex     Scope
```

(continued on next page)

```
                    LC input        Stall           Uparrow
                    No ESC seq      ESC             CTRL/R
                    Resume CTRL/C   Break           No Gag
                    No parity       No fill         Width 132
                    No delimiter    Speed 9600
For KB40:?
```

You do not need to place the LIST command last on the command line and if you want to list the terminal characteristics after each command in a command line, you need only include the LIST command once.

7.6 Initializing a Disk During Timesharing — DSKINT

There are two ways to initialize a disk during timesharing on a RSTS/E system:

- Running the DSKINT utility
- Using the DCL INITIALIZE command, described in Chapter 11

Both perform the same operation: creating a RSTS/E file structure on a new disk or an old one you want to recycle (by getting rid of any files that are on the old disk).

NOTE

Neither the DSKINT utility nor the INITIALIZE command do disk formatting, required by some new disks. Formatting writes necessary timing and sense marks onto the disk and erases any extraneous information. Disk formatting can be done only with the DSKINT option of INIT.SYS, as described in the *RSTS/E System Generation Manual*. Thus, you must shut the system down to format new RK05, RK05F, RP02, RP03, RP04, RP05, and RP06 disks before using them on a RSTS/E system. Other disks do not require formatting; they have been formatted at the factory.

After these disks have been formatted and initialized with the DSKINT option of INIT.SYS, you can reinitialize them during timesharing with DSKINT or the INITIALIZE command. Disks need to be formatted only once.

7.6.1 Running DSKINT

You run DSKINT by typing:

```
$RUN$DSKINT<RET>
```

DSKINT then presents a series of questions.

Table 7–16 presents the questions and a detailed description of appropriate responses. In addition to these responses, you can also answer the prompts with three special characters:

1. Press the LINE FEED key to accept the default response.

2. Press the RETURN key to request an explanation of the question.
3. Type CTRL/Z to back up to the previous question. (However, a CTRL/Z response to the "Disk?" prompt exits the program.)

The following example shows the entire sequence of DSKINT questions.

```
$ RUN $DSKINT (RET)

Disk? DL (RET)
Unit? 0 (RET)
Pack ID? TESTS (RET)
Pack cluster size <1> ? 8 (RET)
MFD cluster size <16> ? 8 (RET)
SATT.SYS base <10221>? (LF)
Pre-extend directories <NO>? Y (RET)
PUB, PRI, or SYS <PRI>? (LF)
Create account [1,1] <NO>? YES (RET)
[1,1] password <*>? (LF)
[1,1] cluster size <16>? (LF)
Create account [1,2] <NO>? YES (RET)
[1,2] password <*>? THELIB (RET)
[1,2] cluster size <16>? (LF)
[1,1] and [1,2] account base <10221>? (LF)
Date last modified <YES>?  (LF)
New files first <NO>? N (RET)
Read-only <NO>? Y (RET)
Use previous bad block info <YES>? N (RET)
Patterns <3> ? 3 (RET)
Proceed (Y or N) ? Y (RET)

Pattern  3
    .
    .
    .
```

After you complete the dialogue and request that initialization proceed, DSKINT begins checking the disk for bad blocks. As the disk is being exercised, DSKINT prints the pattern number currently being used. If it finds a bad block, DSKINT displays information similar to the following:

```
Bad block added to BADB.SYS
 Block Cluster
  9382    9381
  9545    9544
```

Errors identified as "Recoverable" are not added to the bad block file BADB.SYS. If an error is not recoverable, DSKINT prints the block number and cluster number of the bad block and adds the block to the bad block file. These blocks will not be used to store data.

Table 7–16: DSKINT Dialogue Questions and Responses

Disk?

Type two characters to indicate the type of disk being initialized. Acceptable entries are DF, DS, DK, DL, DM, DP, DR, DB, or DU.

Unit?

Type the physical unit number on which the disk resides. Acceptable entries are 0 through 3 for DL disks and 0 through 7 for other disks. For an RK05F disk, use DSKINT twice, once for each half of the disk (that is, once for each of the two units.)

If the disk already has a RSTS/E file structure, DSKINT notifies you and displays the disk's pack label information. For example:

This disk pack appears to be a RSTS/E formatted disk with the following characteristics:

```
Pack ID :                    SAMPLE
Pack Cluster Size :          8
Pack is currently :          Private,
                             Update access date on writes,
                             Read-only,
                             Level: 1,1
```

Pack ID?

Type one to six alphanumeric characters to be used when logically mounting the disk. RSTS/E uses this pack ID as a system-wide logical name.

Pack cluster size <nn> ?

Type the number of 512-byte blocks that each cluster allocated on the disk will contain. Table 7–17 describes acceptable pack cluster sizes for the various types of disks. The default, shown in angle brackets (< >) in the prompt, is the device cluster size for the device you are initializing.

MFD cluster size <16> ?

Declares the cluster size for the MFD (Master File Directory). The MFD cluster size is the number of 512-byte blocks that each cluster allocated to the MFD contains.

The Master File Directory (MFD) is one of the "catalogues" a RSTS/E system maintains for a disk. It is an important part of the RSTS/E file structure. Along with other crucial structures, it is updated each time accounts and files are added to or deleted from the disk. Thus, because you access it often, you increase performance when the MFD cluster size is large. A small MFD cluster size can save some disk space.

The MFD cluster size can be 4, 8, or 16; it must be greater than or equal to the pack cluster size. The default is 16.

SATT.SYS base <nnnn>?

Press LINE FEED to let DSKINT position the storage allocation file (SATT.SYS) near the center of the disk. (The number between angle brackets, <nnnn>, is the default.) SATT.SYS is a bit map that describes which pack clusters have been allocated.

SATT.SYS controls the location of files on the disk. The monitor updates SATT.SYS as data is written to and deleted from disk. It is generally advantageous to place the SATT.SYS file at the midpoint of removable disks.

(continued on next page)

Table 7–16: DSKINT Dialogue Questions and Responses (Cont.)

If you want to position the SATT.SYS file yourself, type the device cluster number where you want the file placed. If your disk will contain a number of large contiguous files, for example, you might want to locate the SATT.SYS near the front of the disk to avoid fragmenting your data files.

A device cluster number can range from 1 to the device size divided by the device cluster size. (Device sizes and device cluster sizes are listed in Table 7–17.)

Pre-extend directories <NO>?

Type Y to have DSKINT preextend the file directories for accounts [1,1] and [1,2] to seven clusters. This guarantees that contiguous space is allocated for these accounts, which can improve disk access times.

PUB, PRI, or SYS <PRI> ?

Press the LINE FEED key to accept the default, which designates the disk as private. A private disk is accessible only to those with accounts on the disk.

You can type PUB to designate the disk as public, although in most cases you probably do not want to do so. A public disk is accessible to anyone with an account on the system. Mounting a public disk requires privilege. Once it is mounted, the disk is a logical extension of the system disk, and any user can create files on it.

You can create a system disk by typing SYS in answer to this question. DSKINT then creates the accounts [1,1] and [1,2] on the disk. Note that you must then transfer the appropriate files to these accounts and use the HOOK utility to create a runnable system disk. (HOOK is a utility normally run by command files during system generation.)

Create account [1,1] <NO> ?

DSKINT skips this question if you are creating a system disk. The account [1,1] is always created when you initialize a system disk.

A nonsystem disk does not need account [1,1], although if you answer "NO" you can still create one later using the REACT program. Accept the default of "NO" if you do not want to create the account with DSKINT.

Both DSKINT and REACT let you preallocate space for the account and position it on the disk. If you answer YES, DSKINT displays the next two questions about [1,1].

[1,1] password <*> ?

Type a one to six character alphanumeric password for account [1,1]. This question appears only if you requested creation of the account.

The default is *; that is, the "no access" password. If you accept the default, or if you assign a password containing at least one question mark (?), no one will be able to log in to account [1,1].

[1,1] cluster size <16> ?

Defines the cluster size for the User File Directory (UFD) for account [1,1]. This question appears only if you requested creation of the account.

The User File Directory for account [1,1] is a "catalogue" of files for this account. It is often accessed, so you can improve disk access time by specifying a large number for this cluster size. To conserve space, you can pick a value of 1, 2, 4, or 8; but the [1,1] cluster size must be greater than or equal to the pack cluster size.

Create account [1,2] <no> ?

DSKINT skips this question if you are creating a system disk. The account [1,2] is always created when you initialize a system disk.

Both DSKINT and REACT let you preallocate space for the account and position it on the disk. If you answer YES, DSKINT displays the next two questions about [1,2].

(continued on next page)

Table 7–16: DSKINT Dialogue Questions and Responses (Cont.)

[1,2] password <*> ?

Type a one to six character alphanumeric password for account [1,2]. This question appears only if you requested creation of the account.

The default is *; that is, the "no access" password. If you accept the default, or if you assign a password containing at least one question mark (?), no one will be able to log in to account [1,2].

[1,2] cluster size <16> ?

Defines the cluster size for the User File Directory (UFD) for account [1,2]. This question appears only if you requested creation of the account.

The User File Directory for account [1,2] is a "catalogue" of files for this account. It is often accessed, so you can improve disk access time by specifying a large number for this cluster size. To conserve space, you can pick a value of 1, 2, 4, or 8; but the [1,2] cluster size must be greater than or equal to the pack cluster size.

[1,1] and [1,2] account base <nnnn>?

Press LINE FEED to let DSKINT position the UFDs for accounts [1,1] and [1,2] near the place where you positioned the storage allocation file (SATT.SYS). (The number between angle brackets, <nnnn>, is the default.) If you did not select a position for SATT.SYS, the default for these UFDs is near the center of the disk.

If you want to position these UFDs somewhere else, type the device cluster number where you want them placed. A device cluster number can range from 1 to the device size divided by the device cluster size. (Device sizes and device cluster sizes are listed in Table 7–17.)

Date last modified <YES>?

One of two dates are kept for files in a disk directory. Type Y or press LINE FEED to keep the date on which the files were last modified. Type N to keep the date on which the files were last accessed.

New files first <NO>?

When you list files in a disk directory, new files can either be listed first or last. Type N or press LINE FEED to cause new files to be placed at the end of the directory for the account in which they were created. DIGITAL recommends this response.

Type Y to cause new files on this disk to be added at the beginning of the directory for the account in which they are created. However, be aware that this requires you to run REORDR more often.

Read-only <NO>?

Type YES to make the disk default to read-only when it is mounted. If you press the LINE FEED key for NO, the disk instead will default to read-write when it is mounted.

A disk initialized as read-only can still be mounted using the DCL MOUNT/WRITE command sequence. Similarly, a disk initialized as read-write can still be mounted read-only using the DCL MOUNT/NOWRITE or UTILTY MOUNT/RO command sequence.

Use previous bad block info <YES>?

Type Y or press LINE FEED to have a new bad block file (BADB.SYS) created using information from the existing bad block file, if any.

Type N to have DSKINT ignore the current bad block file when creating the new one. You might want to use this option if you have a large bad block file on a disk and you suspect the problem may be with the disk drive, rather than the disk pack.

To check the reason for a large number of bad blocks, you can take the pack to another drive, run DSKINT with N for this option, and see if the bad block file is smaller. If so, the problem is probably with the first disk drive.

(continued on next page)

Table 7–16: DSKINT Dialogue Questions and Responses (Cont.)

Patterns <3> ?

DSKINT uses pattern checks to locate bad blocks on the disk. That is, DSKINT will write a pattern to the disk and then read it to check that the data was written correctly. You can specify up to 3 patterns, meaning 3 passes over the disk. Any bad blocks discovered during pattern checking are added to the bad block file (BADB.SYS) so that data will not be written to those blocks.

You can skip the pattern check by specifying 0. In this case, DSKINT writes one pattern to disk (for security, destroying any old data on the disk), but does not read them to verify the patterns.

Proceed (Y or N)?

Type Y to proceed with the disk initialization. Type N to abort the initialization. This question lets you double-check your responses to the dialogue questions and abort the initialization if you have any errors.

Table 7–17: Disk Size and Clustersize

Disk Type	Device Cluster Size	Acceptable Pack Cluster Size (/CLUSTER_SIZE)	Total Device Size (blocks)
RX50	1	1, 2, 4, 8, 16	800
RF11	1	1, 2, 4, 8, 16	1024 times number of platters
RS03	1	1, 2, 4, 8, 16	1024
RS04	1	1, 2, 4, 8, 16	2048
RK05	1	1, 2, 4, 8, 16	4800
RK05F	1	1, 2, 4, 8, 16	4800 for each unit; 2 units per drive
RL01	1	1, 2, 4, 8, 16	10220
RL02	1	1, 2, 4, 8, 16	20460
RD51	1	1, 2, 4, 8, 16	21600
RC25	1	1, 2, 4, 8, 16	50902 for each unit; 2 units per drive
RK06	1	1, 2, 4, 8, 16	27104
RK07	1	1, 2, 4, 8, 16	53768
RP02	2	2, 4, 8, 16	40000
RP03	2	2, 4, 8, 16	80000
RM02	4	4, 8, 16	131648
RM03	4	4, 8, 16	131648
RP04	4	4, 8, 16	171796
RP05	4	4, 8, 16	171796
RA80	4	4, 8, 16	237208
RM80	4	4, 8, 16	242575
RP06	8	8, 16	340664
RA60	8	8, 16	400175
RM05	8	8, 16	500352
RA81	16	16	888012

7.7 Using the ONLCLN Program

ONLCLN is a privileged program that repairs corrupt disk file structures under timesharing. You must first physically but not logically mount the corrupt disk, run the ONLCLN program with the RUN command, and identify the disk you want rebuilt. ONLCLN then performs a number of operations to correct the altered file structure. Section 7.6.2 describes these operations.

DIGITAL recommends that you add commands to the CRASH.CTL command file to run ONLCLN. Include in the command file the specifications for all disks that must be mounted during the crash/startup procedure. DIGITAL also suggests you allocate at least a 28K-word SWAP MAX when using ONLCLN. If you normally use a 16K-word SWAP MAX, you need to increase the size of the system swap files, unless you decrease the JOB MAX.

7.7.1 When to use ONLCLN

Several RSTS/E system errors can be corrected with the ONLCLN program:

- ?Disk pack needs REBUILDing
- ?Bad directory for device
- ?Corrupt file structure

When you physically and logically mount a disk, the monitor sets an indicator on the mounted pack. If by accident you remove the pack without logically dismounting it, the indicator remains set. The monitor notices the indicator is still set and issues a message that the pack needs rebuilding whenever you attempt to remount. Run ONLCLN and when the rebuilding operation is complete, logically mount the disk. The disk is then ready to use.

The ONLCLN program finds and deletes corrupt disk directories as part of its normal operation. When the monitor prints the ?BAD DIRECTORY FOR DEVICE message, you may correct the error condition by running ONLCLN. In addition, try running ONLCLN to correct errors that caused a corrupt file structure condition to occur.

7.7.2 Running the ONLCLN Program

Type RUN $ONLCLN, and press the RETURN key to run the ONLCLN program. ONLCLN prints a header line and the following question:

```
Disk?
```

Type the device mnemonic, the unit number of the drive on which the disk is mounted and a colon (:), and then press RETURN. For example, the proper

response if you want to rebuild an RP06 mounted on unit 0 is DB0:. If you specify only the device mnemonic, ONLCLN prompts you for the unit number of the drive:

```
Unit?
```

Enter the unit number and press the RETURN key. ONLCLN prints a help message when you press RETURN in response to the DISK question:

```
Type the name of the disk to rebuild.
The disk must be physically but not logically mounted.
```

The program then prints the DISK question again. As soon as you specify the correct disk, ONLCLN:

- Builds a new storage allocation table (SATT.SYS) that reflects all files on the disk
- Scans all directories and deletes all files that have the file type .TMP, that are marked for deletion, or that have no accounting entry
- Finds any doubly-allocated blocks and allows you to specify their correct allocation
- Finds and deletes all invalid directories
- Marks the disk as having a rebuilt file structure
- Allows you to delete files that contain bad blocks
- Zeros all blocks that were in the old storage allocation table (SATT.SYS) but not in the new one

When the operation is complete, ONLCLN returns contol to your keyboard monitor.

7.8 Optimizing Disk Directory Structure — REORDR

The REORDR program can restructure the disk directories on your system to improve disk access time. This process begins once you run REORDR and answer the set of questions in its dialogue. The following sections discuss the operations REORDR performs, the dialogue, and restructuring the public disk structure and give an example of the entire process.

7.8.1 Why Use REORDR?

The system catalogues RSTS/E files on disk in user file directories (UFD). As you create, delete, and extend files, the file directory entries become scattered across the surface of the disk. This scattering increases both the time to open files and the time to retrieve data blocks.

The REORDR program can perform three operations to restructure disk directories. Each of these operations can improve the performance of your

system. After you run the program and answer a sequence of dialogue questions, REORDR:

1. Places the list of file names in as few physical disk blocks as possible. This reduces the number of directory accesses the system needs to make for a file open operation.
2. Attempts to place all directory information concerning the physical location of file data (retrieval pointers) for a file into contiguous disk blocks. This reduces the number of disk directory accesses required to perform disk read and write operations.
3. Optionally sorts the file name list in one of four ways depending on how you answer the dialogue questions. This reduces the number of directory accesses the system needs to make for a file open operation.

The retrieval pointer information for the data blocks in a file may become scattered throughout the UFD. Therefore, as file processing proceeds, the monitor cannot get enough information in a single directory file read operation to allow many data block accesses before the next directory file read operation is needed.

When RSTS/E creates a new file in an account, it adds the file name at the end of the current directory list. That is, RSTS/E places files last in the directory if, during system generation, you typed NO to the DSKINT question NEW FILES FIRST. (See the *RSTS/E System Generation Manual.*) This ordering, while not efficient under most situations, may be more efficient for your installation. You should have new files first only if users on your system do not create many new files and their accounts are not large. Otherwise, because of the way RSTS/E retrieves user directory information, it is more efficient to have new files last. With the REORDR program, you can choose to organize files by access or creation date. If you are not creating many new files, it may be advantageous to have new files first.

7.8.2 Dialogue Questions and Responses

The REORDR program prints a set of questions after you type RUN $REORDR:

```
$ RUN $REORDR
REORDR V8 RSTS V8 TIMESHARING

Sort Directory(s) (YES/NO) <NO>? YES

Order by CRE[ATION] or ACC[ESS] Date<CRE>? RET

In FOR[WARD] or REV[ERSE] Order<FOR>? RET

Device and UFD Specification(s)? DB1:[*,*],DB2:[*,*]
```

Table 7–18: REORDR Dialogue Questions

Questions and Responses
SORT DIRECTORY(S) (YES/NO) <NO>?
Type NO or press the RETURN key to keep the existing order of the account(s) you are processing. Either response causes REORDR to skip the next two questions.
Type YES to have REORDR sort the account(s) you designate. REORDR asks the next two questions to determine how you want the account(s) sorted.
ORDER BY CRE[ATION] OR ACC[ESS] DATE<CRE>?
Type CRE or press the RETURN key to have REORDR sort the account(s) by creation date.
Type ACC to have REORDR use ACCESS date to sort the account(s). Depending on how the disk was initialized, the access date can have one of two meanings. It can mean (1) when the file was last accessed or (2) when it was last modified. (The abbreviation DLW in the comments part of the disk status report of SYSTAT or the VT50PY program shows that a disk's access date is set to the date of last modification.)
IN FOR[WARD] OR REV[ERSE] ORDER<FOR>?
Press the RETURN key or type FOR if you want REORDER to sort oldest files first.
Type REV to have REORDR create a sorted file with newest files first.
DEVICE AND UFD SPECIFICATION(S)?
Enter the device mnemonic(s) and unit number(s) of the device(s) you want REORDR to process. Include with the device designator the account(s) you are having reordered. For example, the specification DB0:[1,210],DB1:[*,210] is a valid response. Although SY: (for the entire public structure) is not a valid device, SY0: (for the system disk, the device that was bootstrapped) is acceptable for the device response. An asterisk is a valid character in either or both of the project-programmer account fields. It tells REORDR to process all project or all programmer numbers on the device. You can also include multiple specifications if you separate them with commas. The account designators are: [*,*] Process all user file directories on the disk. [p,*] Process all user file directories with project number p. [*,pn] Process all user file directories with programmer number pn. [p,pn] Process only the user file directory with project number p and programmer number pn. REORDR does not order account [1,1], even though it may be implicit in an account designation.

7.8.3 Reordering Your Disks

To reorder all disks in the public disk structure, you must include the device designator for each disk and the account numbers of the files you want reordered:

```
Device and UFD Specification(s)? DB0:[*,*], DB1:[*,*]
```

REORDR orders all the directories in the UFD on RP04 init 0 and then all the directories in the UFD on RP04 unit 1. Make sure you logically mount the disks with write access before running the program.

To prevent damage to a directory, REORDR requires that all files in a UFD be closed during the entire operation.

NOTE

REORDR verifies that no file is open when it starts to process the UFD. However, it is unable to detect file opening, creation, or deletion between the initial check and the completion of the UFD processing. REORDR can damage the file structure of a disk if a file creation takes place during reordering. For this reason, follow either of these precautions:

1. Disable logins with the NO LOGINS UTILTY command. Make sure that no other jobs (the spooler, for example) are active on the system.
2. Dismount the disk to be reordered and remount it using the /NOSHARE qualifier.

The only safe way to reorder a UFD on the public structure is to disable logins and to make sure that no other user is currently logged in. You should reorder a UFD on a private disk only when you are sure that no other job will access the account(s) being reordered on that disk.

Running the REORDR program from an indirect command file prevents users from logging in to the system and from accessing files while the disk is being reordered. An example of an indirect command file follows:

```
DETACH
LOGIN  KB0: [1,2]
FORCE  KB0: RUN $REORDR
FORCE  KB0: YES
FORCE  KB0: CRE
FORCE  KB0: FOR
FORCE  KB0: DB0:[*,*],DB1:[*,*]
FORCE  KB0: ^Z
FORCE  KB0: BYE/F
ATTACH
```

In this example, you:

1. Run the REORDR program
2. Answer YES to Sort Directory(s) (YES/NO) <NO>?
3. Answer CRE[ATION] to Order by CRE[ATION] or ACC[ESS] Date<CRE>?
4. Answer FOR[WARD] to In FOR[WARD] or REV[ERSE] Order<FOR>?
5. Specify DB0:[*,*],DB1:[*,*]

6. Respond with CTRL/Z in response to the sort question that REORDR prints after completing the current requests for the devices and accounts you have specified

7. Log out of account [1,2] with the BYE/F command

The example shows [1,2] as the account in which the indirect command file resides. It is also the account from which REORDR reorders the files on DB0: and DB1:. Selecting [1,2] is fine if you do not want to reorder this account. However, if you do, you must place the indirect command file in another account because REORDR does not process an account that contains an open file. The indirect command file in account [1,2] is the open file in this case. The system library account [1,2] contains many important system programs. For this reason, you may choose to place the indirect command file in another account.

NOTE

For clarity, use explicit answers instead of defaults in the indirect command file. While you can force a carriage return (^M) as a response to a REORDR question, making changes to the file at some later date becomes more difficult.

7.8.4 Error and Processing Messages

During the reordering of a User File Directory (UFD), the REORDR program holds the reordered UFD in a temporary file. At the conclusion of the ordering process, REORDR copies the temporary file back to the original UFD. As REORDR processes each UFD, it generates messages that tell why an attempted reorder did not work or a message indicating that it did. The messages REORDR prints in the attempt to reorder an account are:

```
Directory dev:[m,n] cannot be Reordered
Directory dev:[m,n] does not exist
Directory dev:[m,n] has been Reordered
Directory dev:[m,n] is not Reordered - File filename.type is OPEN
Directory dev:[m,n] is not Reordered - No write access to UFD
Directory dev:[m,n] is Null
```

```
|<—— format ——>| |<———————— text ————————>|
```

The general format and the example of the format contain a device designator and an account number:

```
Directory dev:[m,n]  |<———text———>|
Directory DR3:[1,223]  has been Reordered
```

You can find an explanation for each message in Table 7–19.

Table 7–19: REORDR Message Text

REORDR Error Messages and Responses
`Cannot be reordered` Account [0,1] contains open files during timesharing and thus REORDR cannot reorder them.
`Does not exist` REORDR cannot find the account on the device.
`Has been reordered` REORDR has successfully reordered the UFD.
`Is not reordered - file filename.typ is open` A user is accessing the UFD and thus REORDER leaves the UFD intact.
`Is not reordered - no write access to UFD` REORDER encounters a UFD to which it cannot gain write access. It prints the error message and continues.
`Is null` No UFD exists for the account. The following two conditions can cause this error: 1. When you create an account with the REACT program, REACT does not create a UFD. The system sets up a UFD only after you create a file in that account. 2. If you use the UTILTY ZERO command or /ZE switch in PIP on an account, you remove the UFD for the account. If REORDR encounters an account without a UFD, it prints the IS NULL error message.

Fatal errors cause REORDR to print a message in the format:

```
?REORDR Fatal Error [code] <----message-----> AT LINE nnnnnn
```

All fatal REORDR errors include the ?REORDR Fatal Error prefix, a program error code, a message, and a program line number. Code represents the number of the program error that caused the fatal condition to occur. (DIGITAL uses the code number to determine the reason for the fatal error.) REORDR prints a brief message to help identify the error. Finally, the error contains the line number at which the program came to an end. Submit a Software Performance Report (SPR) to DIGITAL whenever REORDR encounters a fatal error.

7.8.5 REORDR Example

The following example shows the REORDR dialogue and the messages that result from the reordering process:

```
$ RUN $REORDR
REORDR V8 RSTS V8 TIMESHARING

Sort Directory(s) (YES/NO) <NO>? YES

Order by CRE[ATION] or ACC[ESS] Date<CRE>? CREATION
```

(continued on next page)

```
In FOR[WARD] or REV[ERSE] Order<FOR>? REVERSE

Device and UFD Specification(s)? DB1:[*,*],DB2:[*,*]
Directory DB1:[0,1] cannot be Reordered
Directory DB1:[1,1] cannot be Reordered
Directory DB1:[1,202] has been Reordered
Directory DB1:[1,3] has been Reordered
Directory DB1:[1,4] has been Reordered
Directory DB1:[120,50] has been Reordered
Directory DB1:[1,13] has been Reordered
Directory DB1:[1,200] has been Reordered
Directory DB1:[1,201] has been Reordered
Directory DB1:[1,203] has been Reordered
Directory DB2:[1,28] has been Reordered
Directory DB2:[1,8] has been Reordered
Directory DB2:[1,44] is Null
Directory DB2:[120,54] is Null
Directory DB2:[251,0] has been Reordered
Directory DB2:[100,100] has been Reordered
Directory DB2:[2,240] has been Reordered
Directory DB2:[210,203] has been Reordered
Directory DB2:[232,13] has been Reordered
Directory DB2:[1,77] is Null
Directory DB2:[26,12] has been Reordered
Directory DB2:[232,15] has been Reordered
Directory DB2:[2,227] has been Reordered

Sort Directory(s) (YES/NO) <NO>? CTRL/Z

$
```

7.9 Processing User Comments with GRIPE

The GRIPE program allows users to place comments about system operations in a file where the system manager can read them. This section describes the commands you as the system manager can use to process these user comments. Nonprivileged users should see the discussion of GRIPE in the *RSTS/E System User's Guide*; the discussion here covers privileged operations only.

7.9.1 Comments in GRIPE.TXT

The comments users make reside in a common file named GRIPE.TXT. Each user comment in the GRIPE.TXT file consists of (1) an identification line and (2) the text of the comment. The identification line contains:

- The name of the user
- The project-programmer number [p,pn] of the user entering the comment
- The keyboard number of the user's terminal
- The date and time the comment was entered

To help you identify who entered a comment, the GRIPE program uses a name from the user's account information in the ACCT.SYS file. If

ACCT.SYS does not contain the name for a user, GRIPE prints **NONAME** instead. It does, however, always supply the project-programmer number:

```
Nancy Hartford                [1,210]      KB36     09-Oct-82 02:23 PM

No one seems to show simple courtesy when picking jobs up at the
line printer. They take their printout but do not separate and file other
jobs. I feel they should. Please distribute a "reminder" if you agree.
```

GRIPE prints an identification line, like the following, for users who do not have a name entry in ACCT.SYS:

```
**NONAME**                    [1,210]      KB36     09-Oct-82 02:23 PM
```

The following section describes how to list and delete comments in the GRIPE.TXT file.

7.9.2 GRIPE Commands: *LIST and *RESET

As a privileged user, you run the GRIPE program in the same way as users who have nonprivileged accounts. Type:

```
$ RUN $GRIPE
```

On the next line, the program prints its identification line and a prompt:

```
GRIPE V8 RSTS V8 TIMESHARING
Yes?  (Press the 'ESCAPE' key to end)
```

After GRIPE prints the YES question line, you can type either of the following commands:

- *LIST prints the contents of the GRIPE.TXT file
- *RESET deletes the contents of the GRIPE.TXT file

The asterisk (*) character is a necessary part of the command. GRIPE treats LIST and RESET as text if you do not include the asterisk as a prefix. If you type LIST or RESET without an asterisk as the first character and then press the ESCAPE key, GRIPE prints a "Thank you" message and exits the program.

If at any time you want to exit the program, either press the ESCAPE key or type CTRL/Z.

An example of how to properly use the *LIST command follows:

```
$ RUN $GRIPE
GRIPE V8 RSTS V8 TIMESHARING
Yes?  (Press the 'ESCAPE' key to exit)
*LIST <ESC> OUTPUT? LP:
```

Type the *LIST command and press the ESCAPE key after GRIPE prints a prompting line. If the text file is empty, GRIPE prints the message NO GRIPES FOUND and exits to your keyboard monitor. Otherwise, the GRIPE program requests an output device on which to list the contents of the GRIPE.TXT file. In response to the OUTPUT question, you can press RETURN to have the text displayed on your keyboard or type a device designator, such as LP: as shown in the example. When GRIPE finishes displaying the comments, it exits to your keyboard monitor.

If you want to clear the contents of GRIPE.TXT after you make a copy of the comments for your files, you must again run the program. The following example shows the use of the *RESET command:

```
$ RUN $GRIPE
GRIPE V8 RSTS V8 TIMESHARING
Yes?  (Press the 'ESCAPE' key to end)
*RESET<ESC>

$
```

To delete all comments from the file, you type *RESET and press the ESCAPE key. After deleting the file, GRIPE returns you to your keyboard monitor.

7.10 Communicating with Other Terminals — TALK

The TALK system program allows you to communicate with another terminal on your system. You can send a message to a user's terminal or to the system console. The program has a protection code of <232>, which permits all users to run it. If you want to let only privileged users run TALK, change the protection code to <124>. You need to tell the nonprivileged users on your system about TALK if you do not plan to restrict its use. The *RSTS/E System User's Guide* contains no documentation on TALK. This gives you the option to keep it a privileged program.

7.10.1 Running the TALK Program

TALK enables you to send messages, line-by-line, to another terminal. If the terminal user receiving your message wants to communicate with you, that person must run TALK also. Both the sending and receiving terminals must be on line, but a user does not need to be logged in on the receiving terminal. Run the TALK program as follows:

```
$ RUN $TALK
TALK V8 RSTS V8 TIMESHARING
```

After TALK prints its idenfication line, it prompts you for the keyboard number of the terminal to which you want to send a message.

```
To which Keyboard (KB)?
```

Press the RETURN key after you enter the keyboard number, 12 for example. (TALK accepts a response such as KB12 but the KB prefix is unnecessary.) To ensure that you do not disrupt an elaborate printout, it is a good policy not to send a message to a terminal unless you know the type of activity the terminal operator is performing. An unscheduled message can sometimes destroy hours of work. You can use SYSTAT or VT50PY to determine the status of another user's job. If the job state is displayed as ^C, then the job is at a keyboard monitor prompt and most likely will not be interrupted by your messages.

If you type the number of an off line terminal, TALK does not issue an error message. It does, in the case of a nonexistent terminal, end the program when you press the RETURN key. Messages you send to an off line terminal, however, are sent normally but are never received. In this case, TALK does not end when you press RETURN.

Once you enter a keyboard number and press the RETURN key, TALK prints the instructions:

```
You may proceed - Carriage Return sends the line
'ALTMODE' ('ESCAPE') sends and terminates the program
```

To send a message, type a line and press RETURN. TALK does not send the line until you press the RETURN key. It then sends the entire line. Because TALK sends each line immediately, it does not print a prompt on your terminal after you press RETURN. Instead, TALK positions the cursor at the beginning of the next line, where you can again type another message.

Pressing the ESCAPE key ends the TALK program. You can use ESCAPE instead of the RETURN key if you want to send a message, exit the program, and return to your keyboard monitor. The ESCAPE key prints an ESCAPE character (echoed as $) on the sending terminal. It prints an ESCAPE character on the receiving terminal when you press ESCAPE instead of RETURN to send the line. The cursor on the receiving terminal remains at the end of the message. That is, the system does not perform a carriage return operation on the receiving terminal. On the sending terminal, TALK prints the ESCAPE character. The system then prints the keyboard monitor prompt on the line immediately under the message you just sent and positions keyboard monitor the third line down on the left margin.

7.10.2 Terminal Session — TALK

This session represents a typical dialogue using TALK. The sending terminal has a keyboard number of 9; the keyboard number for the receiving terminal is 12. After running SYSTAT to see if the user at the receiving terminal is not performing a critical operation, use the TALK program as follows:

```
$ RUN $TALK
TALK V8 RSTS V8 TIMESHARING
To which Keyboard (KB)? 12
You may proceed - Carriage Return sends the program
'ALTMODE' ('ESCAPE') sends and terminates the program
Robert,
```

(continued on next page)

```
Would you please take the RP06 disk, marked SYSGEN, from my a cabinet

and place it on line? I will be needing it later. Thanks! Davo

$
$
```

The printout on the receiving terminal automatically identifies the sending device by enclosing it with asterisk (*) characters. The ESCAPE key you typed to end the message does not appear on the receiving terminal. (It prints only when you end the message line with the ESCAPE key.) The message from keyboard number 9 appears on the receiving terminal as follows:

```
** KB12 ** Robert,
** KB12 **
** KB12 ** Would you please take the RP06 disk, marked SYSGEN, from my cabinet
** KB12 **
** KB12 ** and place it on line? I will be needing it later. Thanks! Davo
** KB12 **
```

The operator at keyboard 12 can return a message:

```
$ RUN $TALK
TALK V8 RSTS V8 TIMESHARING
To which keyboard (KB)? 9
You may proceed - Carriage Return  sends the line
'ALT MODE' (' ESCAPE') sends and terminates the program
Davo,

Found it! And, you probably thought it would be easy. Such a cabinet!

(Just kidding.) Your RP06, labeled SYSGEN, is now on line.

Robert$
$
```

Notice that Robert typed the ESCAPE key instead of RETURN. This displays the ESCAPE character on both the sending and receiving terminals. The message sent by Robert would appear on KB9 as follows:

```
** KB9 ** Davo,
** KB9 **
** KB9 ** Found it! And, you probably thought it be would be easy. Such a cabinet!
** KB9 **
** KB9 ** (Just kidding.) Your RP06, labeled SYSGEN, is now on line.
** KB9 **
** KB9 ** Robert$
```

If you place blank lines between lines of text, your messages can be read more easily.

Chapter 8
The BACKUP System Package

The RSTS/E BACKUP programs allow you to create a disk or magnetic tape copy of files for off-line storage and return the files to on-line use when necessary.

As you read this chapter, keep in mind that:

- "BACKUP", printed in all uppercase letters, refers to the entire package; "Backup", with initial capital only, is the name of a particular BACKUP operation.
- The Backup program does not preserve disk files larger than 65,535 blocks. If you try to back up a disk that contains such a file, Backup notifies you of the error, identifies the large file, and then continues the transfer operation. Use the SAVE/RESTORE package, described in Chapter 9, or PIP.SAV to make an off-line copy of a large disk file.
- BACKUP does not preserve the placement of placed files. It saves and restores them, but their placement is lost. BACKUP notifies you that placement has been lost.
- BACKUP saves contiguous files and attempts to restore them to their original contiguous state. If there is not enough contiguous space on the disk, the file is not restored. The only way to restore the file in that case is to make enough contiguous room for it.

8.1 BACKUP and System Management

BACKUP provides four operational modes:

- Backup, to create magnetic tape or disk copies of system files from system disks
- Restore, to rebuild Backup-created files onto a RSTS/E disk
- Loadindex, to copy the primary index file from the last volume of the Backup Set to a RSTS/E formatted disk
- List, to print directory information contained in a Backup index file

Because each installation is unique, the frequency with which you, as the system manager, back up the system varies. Frequency of backup depends on how the system is used. For example, you may back up the files daily on a system that contains large, frequently updated files but may perform only a weekly backup on a system that is less volatile.

By regularly backing up the system, you protect users from losing hours of work if the system crashes, if disk errors occur, or if the users themselves inadvertently destroy information. However, BACKUP does little to preserve files if the backup medium is error prone. You should use disks or tapes of reliable quality for BACKUP operations. You should carefully store backup copies to protect them from unauthorized users and from a catastrophe that might destroy the system itself.

8.2 How BACKUP Works

Important characteristics of the BACKUP package are:

- Dialogue with the user
- Selection of files
- Entering of accounts (optional – Restore only)
- Transfer of files to and from the backup medium
- File comparison and deletion (optional)
- Building the listing file

The following sections outline the functions BACKUP performs within each of these categories.

8.2.1 Dialogue

The dialogue begins when you run BACKUP. The first question asks you which of the four operational modes you want to perform and allows you, if you choose, to create an indirect command file. The dialogue continues by asking you where to create the work file into which it writes directory and error information. It then asks additional questions to find out which files and accounts to transfer. When the dialogue ends, the program writes a summary of your commands into the listing file.

8.2.2 File Selection

Next, BACKUP searches for all the files and accounts you specified during the dialogue phase. In Backup mode, the program searches the source disk's directory structure for accounts. After finding an account, it searches for files in that account. Backup then sorts the chosen accounts into ascending numerical order. Backup does not sort files within accounts but selects them in the order it finds them in the UFD.

In Restore mode, the program looks up files in one of three index files:

- Primary
- Secondary
- Auxiliary

The indexes are directories of all the tape or disk volumes of the Backup Set. Backup writes a primary index file and a secondary index file as the last two files on the Backup Set. The third index remains in your account and is called the auxiliary index file.

The auxiliary index file can be in one of two formats. If the date format (selected during system generation) is alphabetic, then the format is:

Bddmmm.Jnn

In this case:

- B indicates BACKUP
- dd is the date
- mmm is the month
- nn is the job number

For example, a file named B11JUN.J08 results when BACKUP is run under job number 8 on June 11.

If the date format is numeric, the format is:

Bymmdd.Jnn

In this case:

- B indicates BACKUP
- y is the last digit of the year
- mm is the number of the month
- dd is the date
- nn is the job number

For example, a file named B30611.J08 results when BACKUP is run under job number 8 on June 11, 1983.

If you have a multivolume Backup Set, you must load the last volume of the set before you load the volumes in succession for the Restore operation; Restore must have access to the index before it can begin restoring a disk. To avoid the need to load the last volume of the Backup Set each time you restore a disk, use the Loadindex option of BACKUP. Loadindex places the index on another disk where Restore can access it. See Section 8.7 for more information.

As it makes the selections, BACKUP writes information about each selected file and account into the work file. Therefore, the work file can become quite large as the selection phase proceeds. The size of the work file depends on the number and size of files selected for transfer. During the dialogue, you can specify that the work file be placed on a private disk to ensure that enough space is available for it.

It is not recommended that you specify a contiguous work or listing file; both these files are expanded as BACKUP runs, and if BACKUP tries to expand a contiguous file past the file's initial size a ?PROTECTION VIOLATION error occurs. You must make sure enough room is available on the disk you specified for the work and list files. As a reminder, they must also have different names.

To estimate the size of the work file for Backup and Restore, consider the following guidelines:

- The work file has a fixed overhead of two to four blocks.
- Each file you specify requires one-eighth of a block.
- Each exception also requires one-eighth of a block. (Exceptions are described in Section 8.3.1.)
- Errors and file attribute blocks require an additional one-eighth of a block each.

Therefore, for a system of 512 accounts with an average of 16 files each, no exceptions, and 16 files with attributes, you can expect the work file to contain at least 1094 disk blocks. DIGITAL recommends that you add approximately 10% as error space when backing up a large system. In this example, then, the total is 1204 blocks.

8.2.3 Entering Accounts (Optional - Restore Only)

The Backup Set may contain accounts that do not exist on the disk (destination disk) you are restoring. Restore asks a dialogue question that allows you to reproduce these accounts on the destination disk. When you complete the dialogue, Restore first creates the selected accounts that do not exist on the destination disk and then transfers the files, as described in Section 8.2.4. Accounts recreated on the destination disk retain the same password, quota, and cluster size but not the same account names.

Furthermore, the ENTER ACCOUNTS question does not allow you to selectively create accounts. That is, Restore either creates all the selected accounts on the destination disk or creates none of them. If you want all accounts reproduced, accept the default (YES); otherwise, type NO. Note that accounts that did not contain any files during the Backup operation are not entered during a Restore. Finally, Restore allows only users that are logged into a privileged account to enter accounts on restored disks. It does not ask the ENTER ACCOUNTS question when running under a nonprivileged account.

NOTE

Because all account passwords are on the Backup Set, protect your Backup Set from unauthorized personnel.

8.2.4 File Transfer

When the selection process is complete, Backup transfers the selected files and accounts to the disk or magnetic tape. On a system that has multiple disks in the public structure, Backup transfers all files stored under one account from the entire public structure before proceeding to the next account. Restore transfers files in the order in which it finds them on the Backup Set.

If you include a file specification with the SUPERSEDE dialogue question, Restore replaces files on the destination volume with the backup versions. Accept the NONE default to supersede no files. While the default to the SUPERSEDE question is NONE, you must include a file specification to override the default. The words ALL and YES only make BACKUP try to supersede files named ALL and YES. When BACKUP needs a new volume or encounters an error, it prints a prompt on the job's terminal or (if the job is detached) on the Operator Services Console. BACKUP also logs all errors in the listing file.

8.2.5 File Comparison and Deletion (Optional)

After it copies the files and accounts, BACKUP compares the original files with their duplicates on the Backup Set (if you requested a comparison). It records, in the listing file and on the job's console terminal, any differences between the two files.

Backup then deletes files if you requested it but does not delete any files that generated errors during transfer or comparison. Furthermore, it does not delete files from accounts [1,2], [1,1], and [0,1].

While the default to the DELETE FILE(S) and COMPARE FILE(S) questions is NONE, you must include a file specification to override the default, not enter the words ALL or YES.

8.2.6 Building the Listing File

As the last step, Backup and Restore generate the remaining listing file text. BACKUP lists each file it selected for transfer, whether or not it transferred that file. (For example, BACKUP can select a zero-length file for transfer, but it does not transfer the file's data.) The generation of the listing file can be time-consuming because BACKUP copies ASCII data from the work file to the listing file (which can be output on a terminal or written to disk). The listing file can grow quite large for a large system backup or restore. If the listing file consumes all free disk space, you can abort the

BACKUP run. (The ABORT command is described in Section 8.3.4.) Aborting the run while BACKUP is generating the listing file does not harm the Backup Set or the restored files in any way, but it merely leaves you with an incomplete record of the Backup or Restore.

8.3 Running BACKUP

Type RUN $BACKUP to run BACKUP, which prints the BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T] question. If you want to operate in one of the four BACKUP modes from an indirect command file, type one of the following commands in response to the BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T] question. Substitute the name of the indirect command file for filename.type:

BAC @filename.type

RES @filename.type

LOA @filename.type

LIS @filename.type

The defaults for indirect command file names are BACKUP.CMD, RESTOR.CMD, LOADIN.CMD, and LIST.CMD. Note that, to use the assignable account specifier, two at (@) signs are necessary:

```
BAC @@MYBACK.CMD
```

To create an indirect command file, append the /SAVE (or /SA) switch in response to the BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T] question. The following example illustrates the procedure and BACKUP's response:

```
BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T]? BAC/SAVE

INDIRECT FILE NAME<SY:[  1,213]BACKUP.CMD> ?
```

In specifying a file other than the default, you can designate a disk or tape as the output device. After you answer the question, BACKUP continues to prompt normally, writes your responses into the indirect command file, and carries out the commands.

8.3.1 File Specification

Several questions in the dialogue require you to respond with a file specification. The file specification lists criteria that files must meet for BACKUP to process them. The criteria can include file name, account, and dates of creation and last access. In addition, you can specify one or more files as exceptions from a process.

The file specification has the form:

filename/keyword:comparison:date:time/exception (filename . . .)

The file name is a standard RSTS/E file name specification and can contain the asterisk (*) and question mark (?) as wildcard characters. The keyword is CREATION or ACCESS and tells BACKUP whether the date represents file CREATION or ACCESS. The comparison is BEFORE or AFTER. The meanings of keyword and comparison depend on the "milestone" date supplied later in the specification string. The comparison causes BACKUP to process files created or accessed BEFORE or AFTER the date. The date is in DD-MMM-YY format, where the YY element defaults to the current year, if none is specified. Table 8–1 summarizes the file specification string.

Table 8–1: BACKUP File Specification

Element	Full Name and Meaning
filename	[p,pn]filename.type Process the file(s) specified. Standard wild cards are allowed.
keyword	CR[EATION] Interpret date as a milestone creation date.
	AC[CESS] Interpret date as a milestone last access date.
comparison	BEF[ORE] Process only the file(s) created or accessed before the milestone date.
	AFT[ER] Process only the file(s) created or accessed after the milestone date.
date	DD-MMM-YY. Use this date (depending on keyword and comparison) as the milestone before or after which the file(s) must have been created or accessed in order to be processed.
time	HH:MM. Use to refine date, with CR only. (Time is optional, and given in 24-hour format; the default is 00:01 of date.)
exception	EXC[EPT] Exclude the file(s) in the following specification from the process.
	An EXC phrase can include all the previous elements – but not another EXC phrase.
	Multiple file names in an EXC phrase must be separated by commas, and the list of excepted file names must be enclosed in parentheses.

Separate creation and access date comparisons from each other and from the file name (even if it is null) by a slash (/). A file specification applied to the default file name might be as follows:

```
/CR:BEF:01-NOV-82
```

The preceding specification designates any files named by the default that were created before 1-NOV–82 (that is, 31-OCT–82 or earlier).

A colon (:) separates the components of the specification. You can include only one CREATION comparison and one ACCESS comparison in each file specification. For example:

```
*.BAC/AC:BEF:06-JUN-82/CR:BEF:25-APR-82
```

This specification designates all files with the file type .BAC (in the default account) that were last accessed before 06-JUN–82 and created before 25-APR–82.

In addition to specifying a date, you can designate the time of day (in 24-hour format) in the CREATION comparison only. The following specification designates all files with name PLANTZ that have been created since 21:13 (9:13 P.M.) on 07-NOV–82:

```
PLANTZ.*/CR:AFT:07-NOV-82:21:13
```

You can exempt one or more files from a comparison or operation by including an /EXCEPT switch in the file specification. Follow these rules when using the /EXCEPT switch:

1. Any /EXCEPT: switch is associated with the file specification immediately preceding the "/".
2. All /EXCEPT switch file specifications must be enclosed in a single set of parentheses (...).

As you can see, this command line is incorrect:

```
[1,200]*.*,[1,210]*.*/EXCEPT:[1,200]*.SYS
```

The /EXCEPT switch should not be associated with the [1,210] file specification but rather with the file specification for account [1,200]. Instead of placing the /EXCEPT:[1,200]*.SYS specification after the specification for account [1,210], place it with the file specification, [1,200]*.* as follows:

```
[1,200]*.*/EXCEPT:[1,200]*.SYS,[1,210]*.*
```

This then is the correct command line for the previous example. You now have the /EXCEPT switch immediately following the file specification you want it to define.

The next example illustrates the correct way to use the /EXCEPT switch with multiple file specifications:

```
FROM FILES<[1,200]*.*>?[1,200]*.*/EXCEPT:([1,200]*.SYS,-
CONT>[1,200]*.RTS/CR:AFT:21-JAN-82),-
CONT>[1,210]*.*/EXCEPT:([1,210]*.BAK,-
CONT>[1,210]*.BAC),[1,220]*.*
```

The first line of this file specification backs up everything in account [1,200] except files in account [1,200] with a .SYS file type or a .RTS file type with the specified creation date. The command line on the second line, which is a continuation of the first, causes Backup to process all files in account [1,210], except the ones with .BAK and .BAC file types and all files in account [1,220]. For Backup to process this command line properly, you must place parentheses around the file specifications associated with each /EXCEPT switch.

To include a CREATION or ACCESS comparison in an exception, you must, as in the previous example, enclose the files and any comparisons in parentheses:

```
*.*/CR:BEF:28-FEB-82/EXC:(*.RNO/CR:AFT:01-FEB-82,MT?.*/AC:AFT:31-DEC-81)
```

This command line specifies all files on the default account that were created before 28-FEB–82. The exceptions are:

- All files with file type .RNO that were created after 01-FEB–82
- Any file whose name is three characters long, begins with MT, and has been accessed since 31-DEC-81

A CREATION or ACCESS comparison applies to the files in an EXCEPT comparison as well. To exclude an EXCEPT comparison from CREATION and ACCESS comparisons, you must specify a CREATION or ACCESS date for the exception (and, therefore, enclose the exception in parentheses). A file specification can include only one EXCEPT comparison. The EXCEPT comparison must follow any CREATION or ACCESS date comparisons.

A list of file specifications often uses several lines. Therefore, use a hyphen (-) as the last character before a line terminator to indicate that the next line is a continuation of the current line. In response to the hyphen and line terminator, BACKUP prompts with CONT> and a tab. The specification must include all normal punctuation in addition to the hyphen. BACKUP performs no syntax or semantic processing on a response until you have entered all continuation lines. The following example illustrates again the use of a continuation line but this time without the use of the /EXCEPT switch. (Note that the default is [1,213], which we are currently logged into.)

```
FROM FILES<[1,213]*.*>?BACK.RNO,CHAP3.RNO,TEST.BAS,-
CONT>TAPE.BAS,[2,213]PRINT.BAC
```

For example, the following command line designates all files on account [100,250] (except SOCIO.TMP) that were created after 01-JAN-83:

```
[100,250]*.*/CR:AFT:01-JAN-83/EXC:SOCIO.TMP
```

BACKUP ignores blanks as separators in specification lines but considers them significant as terminators. For example, the following line is illegal because the space terminates the scan after the characters EXC:

```
EXC EPT  :  BEE.BAS
```

On the other hand, in EXCEPT: BEE.BAS, BACKUP ignores the blanks and processes the text as:

```
EXCEPT  :  BEE.BAS
```

You can include a comment in a command line by beginning the comment with an exclamation mark, for example:

```
FROM FILES<[1,213]*.*>? W:MANCH8.RNO!THIS FILE CONTAINS CHAPTER 8
```

If you need to continue a command line as well as annotate it, place the hyphen denoting a continuation line before the exclamation mark and the comment:

```
FROM FILES<[1,213]*.*>? CHAP2.RNO,BACK.CMD,TAPE.BAS-!THIS IS A COMMENT
CONT>
```

To avoid confusion, when you use the exclamation mark as the system-wide logical for account [1,3] in a BACKUP command line, make sure you place the exclamation mark in quotation marks ("!" or '!'):

```
FROM FILES<[1,223]*.*>?"!"HELP.TSK!THIS FILE IS IN ACCOUNT [1,3]
```

BACKUP then differentiates the use of an exclamation mark to denote an account from use as a prefix for a comment.

8.3.2 Running BACKUP under BATCH

The BATCH processing program allows you to perform a BACKUP operation without any dialogue interaction. This is possible if you create and execute a batch control file that contains not only the standard BATCH commands but also:

1. The BACKUP run command, RUN $BACKUP.
2. The responses to all dialogue questions, either as an explicit list within the file or as a list in the indirect command file that is specified in the first dialogue question response. The /SA[VE] switch that creates the indirect command file is discussed in the introduction to Section 8.3.
3. The responses to all the mount dialogue questions.

Refer to the *RSTS/E DCL User's Guide* or the *RSTS/E System User's Guide* for a description of BATCH.

Note that you must know the number of devices necessary for a BACKUP operation before submitting the batch job. If you do not know this number, it is best to include too many devices rather than too few. Furthermore, you must allocate a different drive for each device you specify. That is, you need to make sure there are as many drives of the device type you are using as there are volumes in the Backup Set.

In addition, you must include a response for every mount question that might be generated, such as the MOUNT IT ANYWAY question that appears only if a BACKUP volume's expiration date has not passed or if a magnetic tape cannot be read. The rule is: anticipate all mount questions and include an appropriate response. As an added option for the privileged user, you can bypass all magnetic tape label checking by specifying the /SCRATCH switch in the DEVICE question. Otherwise, account for every mount prompt by including an appropriate response in the batch control file.

8.3.3 The BACKUP Dialogue

The BACKUP dialogue lets you specify the files and accounts to back up, restore, compare to their original versions, and delete. This section describes the parts of the dialogue that differ for privileged and nonprivileged users. Tables 8–2 through 8–5 summarize the questions BACKUP asks and the responses the program requires. During the dialogue, Backup asks the same questions of privileged and nonprivileged users. Restore asks the privileged user the following additional question:

```
ENTER ACCOUNTS <YES>?
```

If you answer NO, Restore does not add any accounts to the destination disk.

If you accept the default, Restore creates, on the destination disk, all accounts that are on the Backup Set but not already on the destination disk. Accounts retain all characteristics (password, quota, clustersize) except account name when Restore creates them. If you specify an account in the FROM FILE(S) <[CUR ACT] *.*> question and that account does not exist on the destination disk, Restore transfers the account only if you answer YES to ENTER ACCOUNTS.

Privileged and nonprivileged users achieve slightly different results by accepting the default to the following Restore question:

```
TO DISK <_SY:[*,*]>
```

Because a nonprivileged user can create files only under the current account, the entire Backup Set is restored, protection codes permitting, to the account by accepting the default. For a privileged user, the default restores all files to the accounts from which they were backed up.

Table 8–2: Backup Dialogue Summary

Question, Response and Meaning
1. BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T]? BAC[KUP](/SA[VE]) Backs up files. If you append /SA[VE], Backup creates an indirect command file and asks question 1a.
1a. INDIRECT FILE NAME <_SY:[CUR ACT]BACKUP.CMD>? File specification Creates an indirect command file with the specified name. Valid output devices are disk and tape. Asked only if you attach /SAVE in question 1.
2. WORK-FILE NAME <_SY:[CUR ACT]Bddmmm.Jnn>? File specification Uses the specified file as the work file. If you accept the default, Backup uses either Bddmmm or Bymmdd as the file-name format. If the system date format, selected during system generation, is ALPHABETIC, the default name format is Bddmmm, where dd is the day of the month and mmm is the alphabetic month abbreviation. If the system date format is NUMERIC, Bymmdd is the default name format. The letter y represents the last digit of the current year, mm is the number of the month, and dd is the day of the month. In the file type .Jnn, nn represents the job number under which BACKUP is running. You can substitute any mounted, write-enabled disk for SY:. By renaming the work file in this response, you can create the auxiliary index file.
3. LISTING FILE <_KB:>? KBn: or LPn: Writes the listing file to the keyboard or line printer unit specified. File specification Writes the listing file to the specified file on disk or tape. The default file name is SY:[CUR ACT]backup.LST.
4. FROM DISK <_SY:>? Disk name Backs up from the specified disk. This disk must remain mounted.
5. FROM FILE(S) <[CUR ACT]*.*>? BACKUP File specification(s) Backs up the specified files.
6. TO DEVICE <_MT:>? MT:, MM:, MS:, or other device name Backs up to the specified medium. (Do not specify a unit number here, only a device type. Backup requests the unit number later.)
7. BEGIN AT <[*,*]*.*>? BACKUP File specification Starts the backup with the file specified. The default starts with the first file matching a file specification in question 5. Answer with a single file specification only. No EXCEPT modifiers are allowed.
8. DELETE FILE(S) <NONE>? BACKUP File specification(s) Deletes the specified files after backing up and comparing them.
9. COMPARE FILE(S) <NONE>? BACKUP File specification(s) Compares the specified files to the originals after backing them up.

Table 8–3: Restore Dialogue Summary

Question, Response and Meaning
1. BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T]? RES[TORE] (/SA[VE]) Restores files. If you append /SA[VE], Restore creates an indirect command file and asks question 1a.
1a. INDIRECT FILE NAME<_SY:[CUR ACT]RESTOR.CMD>? File specification Creates an indirect command file with the specified name. Valid output devices are disk and tape. Asked only if the SAVE switch is appended in question 1.
2. WORK-FILE NAME<_SY:[CUR ACT]Bddmmm.Jnn>? File specification Uses the specified file as the work file. If the default is accepted, Restore uses either Bddmmm or Bymmdd as the file name format. If the system date format, selected during system generation, is ALPHABETIC, the default name format is Bddmmm, where dd is the day of the month and mmm is the alphabetic month abbreviation. If the system date format is NUMERIC, Bymmdd is the default name format. The letter y represents the last digit of the current year, mm is the number of the month, and dd is the day of the month. In the .Jnn file type, nn represents the job number BACKUP is running under. You can substitute any mounted, write-enabled disk for SY:.
3. LISTING FILE<_KB:>? KBn: or LPn: Writes the listing file to the specified keyboard or line printer unit. File specification Writes the listing file to the specified file on disk or tape. The default file name is SY:[CUR ACT]RESTOR.LST.
4. INDEX FILE<PRIMARY>? File specification Uses the specified file as the index file. You can specify an auxiliary index file or an index file created during a Loadindex operation. If you accept the default, Restore uses the primary index file that is on the final volume of the Backup Set.
5. FROM DEVICE<_MT:>? MT:, MM:, MS:, or disk name Restores backed up files from the specified medium. (Do not specify a unit number here, only a device type. Restore requests the unit number later.)
6. FROM FILE(S) <[CUR ACT]*.*>? BACKUP File specification(s) Restores the specified files.
7. TO DISK<_SY:[*,*]>? Disk name Restores the specified files to the designated disk.
8. BEGIN AT<[*,*]*.*>? BACKUP File specification Restores starting with the file specified here. The default starts with the first file matching the file specification in question 6. The answer must be a single file specification. No EXCEPT modifiers are allowed.

(continued on next page)

Table 8–3: Restore Dialogue Summary (Cont.)

Question, Response and Meaning
9. (Privileged only) ENTER ACCOUNTS<YES>? YES Creates, on the destination disk, all accounts on the Backup Set that do not already exist on the destination disk. NO Does not create any accounts. You cannot selectively create accounts.
10. SUPERSEDE<NONE>? BACKUP File specification(s) Overwrites the specified files on the destination disk with the backup versions. If you accept the default, Restore supersedes no files. (To supersede files, you must include a file specification. Restore does not supersede all files on the destination volume if you type ALL or YES. Instead of superseding the entire volume, Restore replaces only files that have a file specification of ALL or YES.)
11. COMPARE FILE(S)<NONE>? BACKUP File specification(s) Compares the restored files to the backup versions after transfer. As in the SUPERSEDE question, you must provide a file specification here instead of an answer such as ALL or YES.

Table 8–4: LOADINDEX Dialogue Summary

Question, Response and Meaning
1. BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T]? LOA[DINDEX]? Copies index file from the Backup Set to a RSTS/E formatted disk. If you attach /SA[VE] to the LOA[DINDEX] response, BACKUP creates an indirect command file and asks question 1a.
1a. INDIRECT FILE NAME <_SY:[CUR ACT]LOADIN.CMD>? File specification Creates an indirect command file with the specified name. Valid output devices are disk and tape. This question is asked only if you use the /SA[VE] switch in response to question 1.
2. WORK-FILE NAME<_SY:[CUR ACT] Bddmmm.Jnn>? File specification Uses the specified file as the work file. If you accept the default, Loadindex uses either Bddmmm or Bymmdd as the file name format. The letters dd represent the day of the month, mmm is the three-character month abbreviation, y is the last digit of the current year, and mm is the number of the month. In the file type .Jnn, nn represents the job number BACKUP is running under. You can substitute any mounted, write-enabled disk for SY:.

(continued on next page)

Table 8–4: LOADINDEX Dialogue Summary (Cont.)

Question, Response and Meaning
3. LISTING FILE<_KB:>? KB: or LP: Writes the listing file to the keyboard or the line printer. If you press the RETURN or LINE FEED key, BACKUP sends the listing file to the keyboard <KB:>. File specification Writes the listing file to the specified disk or tape. If you type SY: and then press the RETURN key, the BACKUP program creates the default listing file [CUR ACT]LOADIN.LST.
4. FROM DEVICE<_MT:>? MT:, MM:, MS:, or disk name Copies the index file from the specified Backup Set device.
5. TO FILE<_SY:[CUR ACT][BINDdd.IND]? File specification Loads the Backup Set index file into the specified file. You can copy the index to a disk device only. The format for the default file name is BINDdd.IND, where dd is the numeric day of the month.

Table 8–5: LIST Dialogue Summary

Question, Response and Meaning
1. BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T]? LIS[T]/SA[VE] Runs the List dialogue. When you attach the /SA[VE] switch to the LIST response, BACKUP creates an indirect command file after you respond to question 1a.
1a. INDIRECT FILE NAME<_SY:[CUR ACT][LIST.CMD]>? File specification Creates an indirect command file with the specified name. Question 1a appears only if you use the /SA[VE] switch in response to question 1. Valid output devices are disk and tape.
2. WORK-FILE NAME<_SY:[CUR ACT]Bddmmm.Jnn>? File specification Uses the specified file as the work file. If you accept the default by pressing the RETURN or LINE FEED key, List uses either Bddmmm or Bymmdd as the file name format. The choice depends on the date format of your system. The letters dd represent the day of the month, mmm is the three-character abbreviation of the month, y is the last digit of the current year, and mm is the number of the month. In the file type .Jnn, nn represents the job number BACKUP is running under. You can substitute any mounted, write-enabled disk for SY:.
3. LISTING FILE<_KB:>? KB: or LP: Writes the backup directory information to the specified device. If you press the RETURN or LINE FEED key, the backup directory is written to your keyboard. Type LP: and press RETURN for a line printer listing.

(continued on next page)

Table 8–5: LIST Dialogue Summary (Cont.)

Question, Response and Meaning
File specification Writes the backup directory information to the specified file. Type SY:, and then press RETURN if you want BACKUP to create the default listing file, [CUR ACT]LIST.LST.
4. INDEX FILE<PRIMARY>? File specification Extracts the index file from the specified file and writes it to the listing file. If you accept the default by pressing the RETURN or LINE FEED key, List requests the index volume. It then takes the directory information from the index volume and writes the information to the listing file. The program asks the next question, FROM DEVICE<_MT:>, if you accept the default in question 4.
5. FROM DEVICE<_MT:>? File specification Asked only if the default is accepted in the INDEX FILE<PRIMARY> question. The file specification response must identify the device on which the index resides. You can, however, accept the magnetic tape default <_MT:> by pressing the RETURN or LINE FEED key. The List program then begins a mounting dialogue, which asks you to mount the Backup Set index volume and identify the target device.

8.3.4 Interruption Commands

The BACKUP package includes interruption commands for your convenience. These commands enable you to detach, terminate, suspend, and continue processing and to check the status of the run.

BACKUP shows it is ready to accept interruption commands by printing an asterisk on the job's console terminal after the dialogue is complete. You can type an interruption command as a response to a request in the mount procedure or any time after the asterisk appears. Table 8–6 describes the interruption commands.

Table 8–6: Interruption Commands

Command and Meaning
ABO[RT] Terminates processing immediately and returns control to your keyboard monitor.
CON[TINUE] Continues processing after a PAUSE command.
DET[ACH] Detaches job from KB:. BACKUP detaches if four conditions are met: (1) OPSER is running, (2) the job has permanent privilege, (3) the listing file is not on KB:, and (4) no messages are currently pending for this copy of BACKUP. BACKUP is given a receiver identification of BACKnn, where nn is its job number. All interaction with BACKUP is done through OPSER. (See Chapter 5 for the description of OPSER.)

(continued on next page)

Table 8–6: Interruption Commands (Cont.)

Command and Meaning
END Terminates after processing the current file.
LAS[T] Reprints, on KB:, the last text printed. If the program is awaiting a response, reprints the question or prompt. If not, reprints the last message.
LEG[AL] Prints, on KB:, a list of the commands and responses that are legal now. In the list, **OTHER** means that a request for information from you is pending, for instance, Backup Set Name. The response to such a request is any appropriate string. You do not need to use quotation marks unless the response can be construed as an interruption command. For example, you can legally name the Backup Set PAUSE, but you must type the name in quotes ("PAUSE" or 'PAUSE').
NOT[ICE] Writes, in the listing file, the specified text. The format of the notice command is: `NOT[ICE] message` Message represents one line of text. No line terminators are allowed in the message.
OFF[LINE] Same function as ABORT.
PAU[SE] Suspends execution until you type CONTINUE. Any legal command typed during a PAUSE is executed immediately.
STA[TUS] Prints, on KB:, BACKUP status information. The information printed varies by phase and appears in the examples in Sections 8.5, 8.6, 8.7, and 8.8.
TER[MINATE] Closes the current volume at the end of the current file or at the end of the current output volume, whichever comes first.

8.3.5 Mounting and Dismounting Volumes

After BACKUP selects all files and accounts for transfer, it requests Backup Set labeling information and the device unit number for the first backup volume. It also prints a volume identification summary.

The labeling information is the name and expiration date of the Backup Set. The name acts as an identifier for the Backup Set. BACKUP interprets the expiration date as the date after which it can automatically write over the data on the volume. The default expiration date is one year from the current date. If you mount a volume for backup before its expiration date, BACKUP asks for confirmation before writing on the volume. For magnetic tapes, BACKUP also requests density (800 or 1600 BPI) and, for 800 BPI tapes, parity.

When backing up the system onto magnetic tape, you have the option to bypass all tape label checking. Backup mounts the magnetic tape volume with no label checking when you specify the /SCRATCH switch in response to the DEVICE question:

```
DEVICE? MM1:/SCRATCH (RET)
```

After answering the labeling questions, mount the volume and either write-enable it for a Backup operation or write-lock it for a Restore operation.* In response to the DEVICE question, type the full device specification (device mnemonic and unit number followed by a colon) of the volume that is mounted.

NOTE

The BACKUP interruption commands and the BACKUP requests RET[RY], SKI[P], and IGN[ORE] are executed during the mount dialogue if any of your responses to any of the questions have as its first three characters the first three characters of a command or request. To avoid this, either make sure the response does not have the same first three characters as the commands and requests, or, enclose the response in quotes.

The following example shows the mount procedure printout. Press the LINE FEED key to select a default response:

```
PLEASE ENTER BACKUP SET NAME<BACK28>  --  (LF)
PLEASE ENTER EXPIRATION DATE<08-Jul-82>  --  (LF)
PLEASE ENTER DENSITY IN BPI<800>  --  800(RET)
PLEASE ENTER THE PARITY<ODD>  --  (LF)
MOUNT    DEVICE:        _MT        :
            ID:         BACK28
          SEQ#:         1
       DENSITY:         800 BPI
        PARITY:         ODD
      IDENTIFICATION WILL BE FINAL UPON SUCCESSFUL MOUNT
DEVICE? MM1(RET)
THIS VOLUME HAS NO BACKUP LABEL!
MOUNT IT ANYWAY<NO>? Y
```

BACKUP attempts to read the mounted tape at all valid density and parity settings (unless you have specified /SCRATCH) to determine whether there is a backup label on the tape. If no information is on the tape (that is, it is a new tape), this process can take several minutes and causes magnetic tape errors to be written to the system error log. DIGITAL therefore recommends that you initialize new tapes with PIP, using the /ZE switch, before you use them with BACKUP. The tape is always written at the requested density and parity settings.

*You must keep the disks from which BACKUP restores write-enabled because bad block files on disk must be updated as required.

When BACKUP has finished using a volume, it prints a dismount message, such as:

```
DISMOUNT DEVICE:        _MM1:
              ID:       BACK28
            SEQ#:       1
         DENSITY:       800 BPI
          PARITY:       ODD
    PLEASE LABEL THIS VOLUME!
```

When the dismount message appears, physically dismount the volume (if necessary) and ready the next volume for processing. BACKUP requests name, expiration date, parity, and density for only the first volume in each Backup Set. Use the information in the dismount message to label the tape. For subsequent volumes, BACKUP requires only the device specification.

NOTE

BACKUP logically mounts disks you use during any of its operations. Do not attempt to logically mount or dismount them with the UTILTY MOUNT or DISMOUNT commands. UTILTY expects a RSTS/E file-structured disk. A BACKUP volume has its own structure and causes UTILTY to generate an error when it tries to logically mount or dismount a BACKUP volume.

8.3.6 Writing the BACKUP Structure on Disks

Each backup disk must contain the special BACKUP structure before the BACKUP package can copy files to that disk. The BACDSK program, which is part of the BACKUP package, writes this format on initialized RSTS/E disks and rewrites this format on BACKUP disks to update bad block information. BACDSK runs automatically as part of the Backup process for privileged users. However, you (or another privileged user) must run the BACDSK program once to write the initial BACKUP format on disks that are used by nonprivileged users. For example:

```
$ RUN $BACDSK
BACDSK V8 RSTS V8 TIMESHARING
DEVICE? DB0:
PROJ,PROG? 1,227

$
```

To the DEVICE question, type the specification of the nonprivileged user's disk. Answer the PROJ,PROG question with the user's project-programmer number. BACDSK then writes the BACKUP format on the disk. The BACKUP structure remains on the disk through subsequent Backup and Restore operations. Note, however, that you must initialize all BACKUP disks with the RSTS/E file structure before you can return them to normal RSTS/E file-structured use.

8.4 BACKUP Error Handling

Four types of errors are possible with BACKUP:

1. Dialogue command errors that occur in Backup, Restore, Loadindex, or List dialogue. BACKUP diagnoses these errors immediately. Usually, you can respond to these errors by typing the correct dialogue answer.
2. Interruption command errors that occur when you type an interruption command. BACKUP responds immediately to these errors, also. Type a valid command to correct the error condition.
3. Volume mount errors that occur when BACKUP mounts tape and disk backup volumes. Some volume mount errors are user errors (for example, specifying an illegal density setting for tape); others involve hardware (for example, tape errors that prevent labeling).
4. Backup processing errors that occur any time during the BACKUP run except during the dialogue. The hardware on which BACKUP is running or logic errors can cause processing errors.

The following four sections describe each of these error types.

8.4.1 Dialogue Command Errors

When BACKUP encounters a dialogue command error (usually a syntax error), it prints the ?COMMAND ERROR message on your terminal. The error message is followed by either a BACKUP-specific error message or a RSTS/E error message generated during the syntax processing of the command. After printing the error message, BACKUP prints a question mark (?) and repeats its prompt. If you type a question mark and press the RETURN key, BACKUP prints the command line up to the position of the error, and then prints the prompt again. Table 8–7 describes the BACKUP dialogue error messages.

Table 8–7: BACKUP Dialogue Error Messages

Message and Meaning
BAD DIRECTORY FOR DEVICE The directory structure on the device you specified is corrupt. Try to place the file on a different device.
CAN'T FIND FILE OR ACCOUNT BACKUP cannot find the file or account you specified. Make sure you typed the file name and account correctly and that they exist.
DEVICE HUNG OR WRITE-LOCKED The device you specified is hung or write-locked or has generated a read or write error. Make sure the device is ready, write-enable the device (if necessary), or specify a different device.

(continued on next page)

Table 8–7: BACKUP Dialogue Error Messages (Cont.)

Message and Meaning
DEVICE NOT AVAILABLE The device you specified is disabled or assigned to another user. Specify a different device.
DUPLICATE SWITCH The file specification contains multiple ACCESS or CREATION comparisons. BACKUP accepts only one of each comparison in a file specification.
ILLEGAL EXCEPT NESTING The file specification includes more than one EXCEPT comparison. BACKUP accepts only one EXCEPT in a file specification.
ILLEGAL FIELD The response contains a field that is not permitted. For example, you cannot specify a device name in the response to the FROM FILES question.
ILLEGAL FILENAME The file name you specified contains invalid characters or has an incorrect format.
ILLEGAL KEYWORD The response contains a misspelled or incorrectly abbreviated keyword or has incorrect punctuation marks.
ILLEGAL OPERAND The EXCEPT comparison in the file specification contains an unmatched parenthesis, or the day or year number in a date is out of range.
INCOMPLETE COMMAND FILE You typed CTRL/Z or the indirect command file ended before the end-of-file. BACKUP returns to the BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T] question.
NO DEFAULT The current dialogue question does not have a default answer. You must type an explicit response.
NOT A VALID DEVICE The device you specified is not on this system.
NOT A VALID DEVICE (PROHIBITED) The device you specified in response to this question is illegal. Refer to Table 8–2, 8–3, 8–4, or 8–5 to determine the valid devices for each question.
PROTECTION VIOLATION The file or account you specified is protected.
TOO MANY FILES OPEN ON UNIT Only one file at a time can be open on a magnetic tape unit. Specify the second file on another device.
TOO MUCH DATA The response contains more data than BACKUP accepts for this question.
UNIT NUMBER NOT VALID BACKUP does not accept a device unit number in the response to this question.

8.4.2 Interruption Command Errors

An invalid interruption command generates one of three error messages. Table 8–8 summarizes these messages and their meanings.

Table 8–8: Interruption Command Error Messages

Message and Meaning
CAN'T DETACH The DETACH command is invalid because one or more of the four conditions described in Table 8–6 is not true.
UNRECOGNIZED COMMAND You typed a string that is not an interruption command.
ILLEGAL COMMAND You typed an interruption command that is illegal (for example, typing CONT when no PAUSE is in effect). Type LEGAL for a list of interruption commands that are currently legal.

8.4.3 Volume Mount Errors

Several errors can occur during the volume mount process. BACKUP prints an error message and then repeats the current prompt. Table 8–9 summarizes these error messages.

Table 8–9: BACKUP Volume Mount Error Messages

Message and Meaning
?DENSITY/PARITY CANNOT BE SET ON THIS DRIVE Specified parity or density is illegal on this type of hardware.
?DISK MUST BE INITIALIZED A disk has been mounted but lacks BACKUP file structure format. You must run DSKINT to initialize any disk that BACKUP uses. The disk must not be logically mounted or opened.
?INVALID BACKUP SET NAME The set name contains illegal characters.
?INVALID DATE Expiration date is not in dd-mmm-yy format or has passed.
?INVALID DENSITY SETTING Valid density settings are 800 and 1600 BPI.
?INVALID DEVICE OR NO UNIT NUMBER The input device specification refers to a device not configured on the current system, or the device unit number is not specified.
?INVALID PARITY SETTING Valid parity settings are ODD and EVEN.
?INVALID SWITCH SPECIFICATION The switch does not contain (as its first three characters) SCR.
?ILLEGAL SWITCH USAGE The /SCRATCH switch was specified during a Restore, was specified for a disk device, or was specified by a nonprivileged user.
?MAGTAPE NOT WRITE LOCKED During a Restore operation, the magnetic tape must be mounted write-locked.

(continued on next page)

Table 8–9: BACKUP Volume Mount Error Messages (Cont.)

Message and Meaning
?MAGTAPE SELECT ERROR The magnetic tape unit specified is not READY or is OFFLINE. Ready the unit or specify another unit.
?MAGTAPE WRITE LOCKED During a Backup operation, the magnetic tape must be mounted write-enabled.
?THIS IS NOT THE RIGHT VOLUME 1. Some volume data does not match with what BACKUP expected. 2. The user is not privileged or is not the owner of the volume.
?YOU ARE NOT PRIVILEGED TO USE THIS VOLUME The volume's expiration date has not yet passed, and the user is nonprivileged (and not the volume's owner).
?UNABLE TO WRITE ON THIS TAPE BACKUP cannot write a label on the magnetic tape because of tape errors. Try another magnetic tape.

8.4.4 BACKUP Processing Errors

BACKUP processing errors can occur as the package selects, transfers, compares and deletes files, and generates the listing file. A processing error can be a hardware error, which indicates a hardware problem (for example, ?DEVICE HUNG OR WRITE-LOCKED), or a logic error, which indicates an attempt at an illogical or prohibited action (for example, ?PROTECTION VIOLATION). The BACKUP package contains error handling routines for common hardware and logic errors.

8.4.4.1 Selection Errors — If a logic error occurs during the selection process, BACKUP skips over the file or account causing the error. BACKUP prints an error message and then selects the next file or account, according to the following rules:

1. If an error occurs while BACKUP is looking up a file, BACKUP skips that file in that account and looks for the next sequential file.
2. If an error occurs during the search for an account, BACKUP terminates the search on that input volume and continues the search for that account on the next input volume.
3. If an error occurs during the search for an account on the final input volume, BACKUP returns to the first input volume and searches for the next sequential account.
4. If an error occurs in the final account on the final input volume, the selection process ends and BACKUP begins to transfer the selected files and accounts.

BACKUP contains error handling routines that allow you to correct or avoid certain problems that cause the ?DEVICE HUNG OR WRITE-LOCKED hardware error. When this error occurs, BACKUP prints the error message, followed by one or more valid request commands (RETRY or SKIP) set in parentheses. After BACKUP issues an asterisk prompt, you can respond with one of the suggested commands. A RETRY response causes BACKUP to attempt the failed operation again. For example, device hung errors may simply indicate:

- A transient hardware problem
- A device has gone off line
- A file or account contains bad blocks

To correct a transient hardware problem may only require you to use the RETRY command. Problems of this type may not occur during a subsequent retry attempt. Also, when a device goes off line you can usually place the device back on line and respond to the asterisk prompt by typing RETRY. Finally, if bad blocks in a file or account cause a device hung error, you can attempt to bypass the bad block area by using the SKIP command. In reply, BACKUP ends its search of the file or account containing the bad block, proceeds to look for the the next file or account, and then attempts to resume normal processing. Using the RETRY and SKIP request commands may correct otherwise irretrievable situations and thereby save you hours of work.

When BACKUP discovers a disk file larger than 65,535 blocks, it displays the following message and then continues the operation:

LARGE FILE <filespec> – WILL NOT BE BACKED UP

The filespec field includes the device, account, file name, and type.

The Backup operation transfers placed files to the Backup Set but does not preserve placement. If BACKUP encounters any placed files during processing, it prints a warning message, transfers the file, and ignores the file's placement. The warning message has the form:

PLACED FILE <filespec> SELECTED PLACEMENT WILL BE LOST UPON RESTORATION!

During a Restore, the message takes the form:

PLACED FILE <filespec> SELECTED – PLACEMENT HAS BEEN LOST!

The filespec field includes the device type, account number, file name and type. Both placed file messages are sent to the keyboard as well as to the listing file.

If a bad block error (?DATA ERROR ON DEVICE) occurs during the selection phase of a Backup operation, Backup prints an error message, stops searching for the current file or account, and looks for the next file or account according to the same rules it uses for logic errors.

During a Restore operation, a bad block error at index load time usually means that the index file is corrupt. If you specified that the primary index file was to be used, the following message is issued:

```
PRIMARY INDEX LOAD FAILED! STARTING SECONDARY LOAD
```

BACKUP attempts to load the secondary index file (which may require you to mount a different volume).

If an error occurs during the secondary index load, BACKUP prints the following message and aborts:

```
SECONDARY INDEX LOAD FAILED. RUN ABORTED!!!
```

You must rerun BACKUP, using the auxiliary index file.

If you specified that the auxiliary index file was to be used and a load error occurs, BACKUP prints the following message and aborts:

AUXILIARY INDEX LOAD FAILED – filename.type DUE TO <text> ERROR.
RUN ABORTED!!!

The <text> is the RSTS/E error that caused the load failure.

A bad block error ?DATA ERROR ON DEVICE that occurs in the Backup or Restore work file is fatal. BACKUP prints the following error message and aborts the run:

```
?UNEXPECTED ERROR --  DATA ERROR ON DEVICE
```

8.4.4.2 Transfer, Deletion, and Listing Errors — Logic errors during the transfer, deletion, and listing processes usually result from failure to open a file that BACKUP must transfer or delete. For example, the ?PROTECTION VIOLATION logic error means you are prohibited by the protection code from deleting or transferring a certain file. The logic error, SUPERSEDE FAILURE, indicates a failure to restore a file that already exists on the destination disk. This error can occur even if you did not request a supersede operation. The BACKUP package automatically skips over the file or account causing a logic error during transfer, deletion, or listing processes.

BACKUP error handling routines allow you to retry (RETRY) or skip (SKIP) over a ?DEVICE HUNG OR WRITE-LOCKED error that occurs during the transfer, deletion, or listing process. RETRY and SKIP work in the same way here as they do during the selection process.

In addition to device hung errors, BACKUP can be interrupted by an error such as ?CAN'T FIND FILE OR ACCOUNT or ?NO ROOM FOR USER ON DEVICE. If one of these errors occurs while BACKUP is processing a listing file, type the IGNORE request command in response to the asterisk prompt BACKUP provides you. This procedure forces BACKUP to disable future I/O to the listing file and to continue with the current operation.

The SEQUENCING PROBLEM ON MAGTAPE error can occur during a Restore from magnetic tape. This error means that Restore read a magnetic tape that did not have the expected record number. When the sequencing error occurs, Restore usually prints several bad block error messages (as it tries and fails to space to the correct magnetic tape record) and then recovers automatically. You cannot restore the file in which the sequencing error occurred, but you can restore the remainder of the files on the tape.

If BACKUP encounters a bad block during a transfer operation that involves a RSTS/E (not a BACKUP) disk, BACKUP prints an error message. It copies the rest of the current file although the file is corrupt. If the bad block is in the work file, BACKUP aborts.

The BACKUP package specially structures each backup disk when you mount it for a Backup operation. BACKUP creates a bad block file on the disk and allocates to this file all bad blocks it finds during formatting. The BACKUP package does not permit any disk that has an excessive number of bad blocks to be used as a backup disk. When a Restore operation discovers a bad block on a backup disk, it allocates the bad block to the bad block file. This procedure prevents future use of the block for backup data. (You should be aware that BACKUP bad block files are not equivalent to the RSTS/E file BADB.SYS. You must reinitialize a backup disk before using it as a RSTS/E file-structured disk. Section 7.5 describes disk initialization procedures.)

8.4.4.3 Informational Messages — During the transfer process, BACKUP often prints informational messages. These messages usually do not indicate errors but inform you about files that are open (and subject to change) and files that have changed in length since selection. BACKUP transfers these files, but the copies may not be accurate. Other messages list files that BACKUP cannot transfer: those deleted since selection and those whose length is zero.

The BACKUP program displays a timed reminder after you type the PAUSE command and when it is waiting for a user response. Two minutes after you type the PAUSE command, which suspends the execution of a BACKUP operation, BACKUP prints the message IN PAUSE. The reminder appears only once. Similarly, BACKUP reminds you, after a two minute interval, that it is waiting for a user response. The FURTHER RESPONSE NECESSARY message does not appear during any of the initial dialogue questions. Like the PAUSE reminder, this message is not repeated.

8.5 Backing Up System Files – Example

In Backup mode, you can create magnetic tape or disk copies of system files. The destination medium, to which you transfer the files, is called the Backup Set. The example in Section 8.5.1 shows how to create a magnetic tape Backup Set from RK06 disk files. Responses to the dialogue questions are explained after the Backup example. The listing file that results from the Backup operation is shown in Section 8.5.2. A description of the characteristics of the Backup listing file follows that example.

8.5.1 Terminal Printout – Backup

To duplicate the example that follows, log on to your system, run the BACKUP program, and then answer the dialogue questions as shown. The letters are keyed to the explanation that follows:

```
   $ RUN $BACKUP
   BACKUP V8 RSTS V8 TIMESHARING

a  BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? BAC

b  WORK FILE NAME<_SY:[ 1,110]B08SEP.J16> ? BACKUP.WKF/MO:256

c  LISTING FILE<_KB:> ? BACKUP.LST

d  FROM DISK<_SY:> ? DM1:

e  FROM FILES<[ 1,110]*.*> ? [*,110]*.*/EXC:[1,110]*.BAK

f  TO DEVICE<_MT:> ? (RET)

g  BEGIN AT<[*,*]*.*> ? (RET)

h  DELETE FILES<NONE> ? [2,110]*.DAT

i  COMPARE FILES<NONE> ? [2,110]*.DAT
        *

j  PLEASE ENTER BACKUP SET NAME<BACKUP> - (RET)
   PLEASE ENTER EXPIRATION DATE<08-Sep-83> - (RET)
   PLEASE ENTER DENSITY IN BPI<800> - (RET)
   PLEASE ENTER THE PARITY<ODD> - (RET)
   MOUNT    DEVICE:        _MT     :
               ID:         BACKUP
             SEQ#:         1
          DENSITY:         800 BPI
           PARITY:         ODD
          IDENTIFICATION WILL BE FINAL UPON SUCCESSFUL MOUNT

k  DEVICE? MM0:(RET)
l  EXPIRATION DATE HAS NOT YET ARRIVED!
               ID:         BACKUP
             SEQ#:         1
m         DENSITY:         800 BPI
           PARITY:         ODD
   EXPIRATION DATE:        08-Sep-83
n       MOUNT IT ANYWAY<NO>?Y


   *(RET)
   DATA UNRELIABLE - FILE OPENED BY ANOTHER USER IN FILE

o  _DM1 :[1,110]ADDR.DAT
   (ON TRANSFER)

p  *STATUS
```

(continued on next page)

```
q  PHASE      : TRANSFER
   VOLUME #   : 1
   ACCOUNTS   : 1
   FILES      : 2
   BLOCKS     : 2010
   ERRORS     : 1

   CURRENT VOLUME  : _MM0    :
r  CURRENT ACCOUNT : _DM1    :[1,110]
   CURRENT FILE    : _DM1    :[1,110]RT11   .RTS<60>

   ELAPSED TIME : 83 SECONDS
s  CPU TIME     : 1.8 SECONDS
   KCTS         : 284


   *
t  NO MORE ROOM ON VOLUME - TERMINATING.
   *


   DISMOUNT DEVICE:         _MM0    :
                ID:         BACKUP
             SEQ#:          1
           DENSITY:         800 BPI
            PARITY:         ODD
   EXPIRATION DATE:         08-Sep-83
           PLEASE LABEL THIS VOLUME!

   MOUNT    DEVICE:         _MM     :
                ID:         BACKUP
             SEQ#:          2
u          DENSITY:         800 BPI
            PARITY:         ODD
           IDENTIFICATION WILL BE FINAL UPON SUCCESSFUL MOUNT
v  DEVICE? MM1:/SCR
   *
w  DISMOUNT DEVICE:         _MM1    :
                ID:         BACKUP
             SEQ#:          2(INDEX)
           DENSITY:         800 BPI
            PARITY:         ODD
   EXPIRATION DATE:         08-Sep-83
           PLEASE LABEL THIS VOLUME!
   *
   $
```

a Select the Backup mode by typing BAC, and press the RETURN key. You must type at least the first three letters of the mode. In the initial BACKUP prompt, the square brackets ([ ]) printed with each mode show this requirement.

b Create the work file BACKUP.WKF. The /MO:256 switch, attached to the work file name, enables data caching for the BACKUP.WRF file. For an explanation of data caching modes, see the *RSTS/E Programming Manual.*

c Specify BACKUP.LST as the file location from which Backup prints the listing file. You usually print the listing file after completing a Backup operation to check the integrity of the run. Section 8.5.2 reproduces and explains the information contained in the listing file for this Backup example.

d Type DM1: to have Backup copy files from an RK06. If the files you want to back up are on a system disk (SY:), you can accept the default.

e Specify the backup of all files in programmer account 110 except those in account [1,110] that have a .BAK file type.

f Transfer all files to a magnetic tape Backup Set by accepting the MT: default. Thus, all the files specified in the FROM DISK question are transferred to magnetic tape.

g Begin the transfer operation with the first file on the input device (RK06) by pressing the RETURN key.

h Delete all files in account [2,110] that have a .DAT file type. The wildcard designator, represented by the asterisk symbol, causes Backup to delete all files in account [2,110] that have a .DAT file type.

i Compare the specified input files to the same files that have been transferred to the magnetic tape Backup Set. Note that BACKUP compares the [2,110]*.DAT files before it deletes them from the input volume. BACKUP then prints an asterisk that indicates the beginning of the selection process and Backup's readiness to accept interruption commands. After printing the asterisk, BACKUP selects files for transfer according to your specifications. When the selection process ends, Backup asks additional questions at j.

j Accept the defaults to the next four questions. Backup always asks questions regarding the Backup Set and Expiration Date; however, the Density and Parity questions are printed only when the Backup Set is magnetic tape. After you answer the last question, BACKUP prints a summary of the tape label information.

k Mount a magnetic tape to which Backup can copy files; type the mnemonic name and unit number of the device on which the tape is mounted, and press the RETURN key. Backup then searches the magnetic tape for a valid label and finds one.

l Notice that, as a result of the magnetic tape label inspection, Backup discovers the expiration date has not arrived. The message means that the date you last placed data on the tape has not expired and thus the tape may still contain valuable information.

m Notice that Backup prints a summary of the label information it found. In this summary, Backup includes the expiration date.

n Override the default and use the magnetic tape despite Backup's warning. The data on the magnetic tape is no longer needed.

o Notice the error message. As it encounters errors in the transfer process, Backup prints messages at the job's console terminal. This error message warns you of open files that may have changed between selection and transfer. Any errors that occur during a Backup run also appear in the listing file.

p Type the STATUS interruption command in response to the asterisk prompt. Backup responds by printing the current status of the operation.

q Note the information that the status report contains. Backup:

- Is in the Backup transfer PHASE
- Continues to work on the first volume of the Backup Set
- Transferred 1 account
- Transferred 1 file
- Copied 2010 blocks
- Encountered 1 error

r Notice that the summary identifies the Backup Set medium, the account, and the file that is currently being transferred.

s Observe that BACKUP prints the elapsed time, the CPU time, and the number of kilo-core-ticks used so far in the current phase. After printing the status report, Backup displays the asterisk prompt to indicate that it is ready to continue with the selection process.

t Note that Backup now requires the use of a second volume because the first Backup Set volume is full. After printing another asterisk prompt, Backup displays label information for the magnetic tape medium you need to dismount.

u Notice that Backup assumes the same label characteristics for the second volume as it assumed for the first. Immediately after the label summary, Backup asks you to mount the second magnetic tape volume.

v Specify the mnemonic name and unit number of the device on which the tape is mounted. Attach the /SCR switch so that Backup will avoid all label checking. Backup resumes transfer processing, indicated by the appearance of the asterisk prompt.

w Notice that Backup issues a summary of the label information on the second volume of the Backup Set after the Backup operation is complete. Control returns to your keyboard monitor.

8.5.2 Listing File – Backup

The Backup listing file summarizes the Backup dialogue, provides statistics on each of the Backup phases, and lists the operations performed on each selected file. The following is the listing file from the Backup example in the preceding section. The letters are keyed to the explanation that follows.

```
                                        Backup Run Listing

   Backup from '_DM1    :' to '_MT     :

   Run started on 08-Sep-82 at 11:33 AM

   Work-File is _SY     :[1,110]BACKUP.WKF/MO: 256

a  Transfer   : [*,110]??????.???EXC:([1,110]??????.BAK)

   Begin at   : [*,*]*.*

   Delete     : [2,110]??????.DAT

   Compare    : [2,110]??????.DAT

   PHASE    : LIST COMPLETE
   ERRORS   : 0

b  ELAPSED TIME : 2 SECONDS
   CPU TIME     : .6 SECONDS
   KCTS         : 87

c  NO FILES FOR ACCOUNT [3,110] ON DISK _DM1    :

   PHASE    : LOOKUPSELECT COMPLETE
   VOLUME # : 1
   ACCOUNTS : 2
   FILES    : 11
d  BLOCKS   : 11589
   ERRORS   : 0

   ELAPSED TIME : 11 SECONDS
   CPU TIME     : 1.3 SECONDS
   KCTS         : 208

   PHASE    : MOUNT /  DISMOUNT COMPLETE
   ERRORS   : 0
e  ELAPSED TIME : 82 SECONDS
   CPU TIME     : 1.4 SECONDS
   KCTS         : 224

   ZERO LENGTH FILE - _DM1    :[1,110]OPEN  .TMP<60> - ONLY
   DIRECTORY ENTRY TRANSFERRED.
f  DATA UNRELIABLE - FILE OPENED BY ANOTHER USER IN FILE
   _DM1 :[1,110]ADDR .DAT
   (ON TRANSFER)
g  NO MORE ROOM ON VOLUME - TERMINATING.

   PHASE    : TRANSFER COMPLETE
   VOLUME # : 1
   ACCOUNTS : 2
   FILES    : 10
   BLOCKS   : 9493
   ERRORS   : 1

h  CURRENT VOLUME  : _MM0    :
   CURRENT ACCOUNT : _DM1    :[2,110]
   CURRENT FILE    : _DM1    :[2,110]MASTER.DAT<60>
```

(continued on next page)

```
    ELAPSED TIME : 290 SECONDS
    CPU TIME     : 11.9 SECONDS
    KCTS         : 1916

    PHASE    : COMPARE COMPLETE
    VOLUME # : 1
    ACCOUNTS : 1
    FILES    : 1
    BLOCKS   : 6904
    ERRORS   : 0

i   CURRENT VOLUME  : _MM0    :
    CURRENT FILE    : _DM1    :[1,1]

    ELAPSED TIME : 368 SECONDS
    CPU TIME     : 27.4 SECONDS
    KCTS         : 4377

    PHASE    : MOUNT / DISMOUNT COMPLETE
    ERRORS   : 0

j   ELAPSED TIME : 84 SECONDS
    CPU TIME     : 1.3 SECONDS
    KCTS         : 240

    PHASE    : TRANSFER COMPLETE
    VOLUME # : 2
    ACCOUNTS : 1
    FILES    : 1
    BLOCKS   : 2096
    ERRORS   : 0

k   CURRENT VOLUME  : _MM1    :

    ELAPSED TIME : 48 SECONDS
    CPU TIME     : 2.2 SECONDS
    KCTS         : 348

    PHASE    : COMPARE COMPLETE
    VOLUME # : 2
    ACCOUNTS : 1
    FILES    : 1
    BLOCKS   : 2096
    ERRORS   : 0

l   CURRENT VOLUME  : _MM1    :
    CURRENT FILE    : _DM1    :[1,1]

    ELAPSED TIME    : 75 SECONDS
    CPU TIME        : 8.5 SECONDS
    KCTS            : 1356

    PHASE    : INDEX DUMP COMPLETE
    ERRORS   : 0

    CURRENT VOLUME  : _MM1    :

m   ELAPSED TIME : 3 SECONDS
    CPU TIME     : .7 SECONDS
    KCTS         : 108
```

(continued on next page)

```
   PHASE    : MOUNT / DISMOUNT COMPLETE
   ERRORS   : 0

n  ELAPSED TIME : 13 SECONDS
   CPU TIME     : .3 SECONDS
   KCTS         : 48

   PHASE    : DELETION COMPLETE
   VOLUME #  : 1
   ACCOUNTS   : 1
   FILES      : 1
   BLOCKS     : 9000
o  ERRORS     : 0

   ELAPSED TIME : 1 SECONDS
   CPU TIME     : .5 SECONDS
   KCTS         : 60

   Backup Set Name : BACKUP                 Backup Device : _MM    :
   Volume Sequence # : 1                    Owner : [1,110]
p  Creation Date : 08-Sep-82                Expiration Date : 08-Sep-83
   Density :  800 BPI                       Parity : ODD


   Account : [1,110]                        Quota : 0
q  Clustersize : 4


                                      Creation          Access
    Name  Ext    Size     Prot      Date      Time       Date      Clu  RTS     TSCD
r   SAVE  .DOC     10       60   10-Jul-82  05:01 PM  10-Jul-82    8 RSX     *T
s   OPEN  .TMP      0       60   08-Sep-82  10:46 AM  08-Sep-82    4 BAS4F
    ADDR  .DAT   2000       60   08-Sep-82  10:46 AM  08-Sep-82    4 BAS4F   T

              Errors :

        DATA UNRELIABLE - FILE OPENED BY ANOTHER USER on FILE
            Total of 1 error encountered on FILE
    RT11  .RTS     20    60  31-Aug-82   04:39 PM   31-Aug-82     4 RSTS   T
    SAVE  .RNO      9    60  10-Jul-82   04:59 PM   10-Jul-82     8 TECO   T
    UTILTY.BAS      4    60  23-Nov-81   10:44 AM   23-Nov-81     8 BASIC  T
    DIRECT.BAS     56    60  21-Jul-82   11:24 AM   21-Jul-82     8 BAS4F  T

       * Attributes associated with this file

     Account total of 2142 blocks in 7 files on account [1,110]
         Total of 1 error encountered on ACCOUNT

    Account : [2,110]                       Quota : 0
    Clustersize : 4


                                      Creation          Access
    Name  Ext    Size     Prot      Date      Time       Date      Clu  RTS     TSCD
    BACKUP.BAS    121       42   23-Aug-82 09:56 AM  23-Aug-82    8 BASIC   T
    BACFRM.BAS    158       42   15-Aug-82 04:49 PM  15-Aug-82    8 BASIC   T
    BACKTO.BAS    168       42   15-Aug-82 04:36 PM  15-Aug-82    8 BASIC   T
t   MASTER.DAT   6904       60   08-Sep-82 10:55 AM  08-Sep-82    4 BAS4F   T C
                 (of 9000)
```

(continued on next page)

```
u     ***** File continued on next Volume *****

      Account total of 7351 blocks in 4 files on account [2,110]
      ***** Account continued on next Volume *****


v   Volume total of 9493 blocks in 11 files in 2 accounts on volume MM    :
       Total of 1 error encountered on VOLUME


         Backup Set Name : BACKUP              Backup Device : _MM     :
         Volume Sequence # : 2(Index)          Owner : [1,110]
w        Creation Date : 08-Sep-82             Expiration Date : 08-Sep-83
         Density :  800 BPI                    Parity : ODD


           Account : [2,110]                   Quota : 0
           Clustersize : 4
                                      Creation          Access
    Name  Ext    Size     Prot      Date      Time       Date        Clu  RTS  TSCD
   MASTER.DAT    2096      60    08-Sep-82 10:55 AM  08-Sep-82       4 BAS4F T CD
                 (of 9000)


           Account total of 2096 blocks in 1 files on account [2,110]


         Volume total of 2096 blocks in 1 files in 1 accounts on volume _MM    :


         RUN total of 11589 blocks in 11 files on 2 accounts on 2 Volumes
x        RUN total of 1 errors


         PHASE     : LIST COMPLETE
         VOLUME # : 2
         ACCOUNTS : 2
         FILES     : 11
         BLOCKS    : 11589
y        ERRORS    : 0


         ELAPSED TIME : 2 SECONDS
         CPU TIME     : 1.7 SECONDS
         KCTS         : 252
```

The Backup listing file:

a Summarizes the Backup dialogue, indicating which operations are to be performed on each selected file. Backup writes this summary into the listing file immediately after executing the dialogue.

b Displays accounting information about the listing process that recorded the dialogue summary. After the dialogue summary is written, the selection begins.

c Indicates that Backup found no files for account [3,110] and therefore has not accepted that account for transfer.

d Summarizes the selection phase that lists the total number of files, accounts, and disk blocks selected for transfer. The total number of errors that occurred during the selection phase (zero for this Backup) is also listed. The final part of the selection summary lists accounting information.

e Records the magnetic tape mounting process and its statistics. When the magnetic tape is mounted, the transfer begins.

f Lists messages from the transfer phase. The ZERO LENGTH FILE message tells you that [1,110]OPEN.TMP is a zero-length file that Backup does not transfer. However, Backup does transfer the directories file. One DATA UNRELIABLE error occurred for the file [1,110]ADDR.DAT. The error was printed at the job's terminal during the run, as noted in Section 8.5.1, and is listed in the summary at h.

g Prints the NO MORE ROOM ON VOLUME message, which appeared during the BACKUP operation and indicates volume 1 of the Backup Set is full.

h Summarizes the transfer phase.

i Summarizes the compare phase.

j Displays information concerning the dismount process for volume 1. Because there are two volumes in the Backup Set, the listing file also summarizes in the next four steps the transfer, compare, the index dump, the mount/dismount process, and the deletion process for the second volume.

k Summarizes the transfer phase for the second volume of the Backup Set.

l Summarizes the compare phase.

m Displays statistics for the index dump phase.

n Indicates that the mount/dismount phase for volume 2 is complete.

o Summarizes the deletion phase. All of the summaries, h through o, have time statistics.

p Displays the label information for volume 1 and then reports directory information for each selected file and account for that volume.

q Lists the account number, quota, and the cluster size of [1,110], the selected account. For each selected file in the account, the listing file displays the file name, size in decimal blocks, protection code, creation date and time, last access date, the file cluster size, and the associated runtime system. The final column, labeled TSCD, contains a T if the file was transferred, an S if it was superseded (possible only in a Restore operation), a C if it was compared, and D if it was deleted (possible only in a Backup operation).

r Transfers file SAVE.DOC and notifies you of the transfer by placing a T in the TSCD column. The asterisk before the T means there are attributes associated with that file.

s Does not process OPEN.TMP, a zero-length file, and shows this by leaving the TSCD column blank.

t Transfers and compares MASTER.DAT. Note that the MASTER.DAT file is continued onto the second volume of the Backup Set.

u Notifies you that the MASTER.DAT file is a split file and that it is continued on the next volume.

v Provides a summary of blocks, files, accounts, and errors for that volume after listing directory information for the first volume of the Backup Set.

w Gives you directory information for each selected file and account for volume 2. The directory information for volume 2 is in the same format as it is for volume 1.

x Prints the total statistics for the Backup run after listing each volume.

y Ends with a summary of the statistics for the final listing phase.

8.6 Restoring Files – Example

Restore recreates a RSTS/E formatted disk from a magnetic tape or disk Backup Set. The Restore example in Section 8.6.1 shows how to restore an RK06 disk from a magnetic tape Backup Set. The listing file that results from the Restore run in Section 8.6.1 is reproduced and described in Section 8.6.2.

8.6.1 Terminal Printout – Restore

To duplicate the following example, run the BACKUP program, and respond to the resulting questions. The letters in the printout are keyed to the description that follows:

```
   $ RUN $BACKUP
   BACKUP V8 RSTS V8 TIMESHARING

a  BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? RES

b  WORK FILE NAME <_SY:[  1,110]B08SEP.J16> ? RESTOR.WKF

c  LISTING FILE <_KB:> ? RESTOR.LST

d  INDEX FILE<PRIMARY> ? RET

e  FROM DEVICE<_MT:> ? RET

f  FROM FILES<[  1,110]*.*> ? [2,110]BACKUP.BAS,BACKTO.BAS

g  TO DISK<_SY:[*,*]> ? DM1:

h  BEGIN AT<[*,*]*.*> ? RET

i  ENTER ACCOUNTS<YES> ? NO
```

(continued on next page)

```
j   SUPERSEDE FILES<NONE> ? RET

k   COMPARE FILE<NONE> ? RET

l   *
m   PLEASE ENTER BACKUP SET NAME<RESTOR> - BACKUP
    PLEASE ENTER DENSITY IN BPI<800> - RET
    PLEASE ENTER THE PARITY<ODD> - RET
    MOUNT    DEVICE:          _MT     :
                 ID:          BACKUP
               SEQ#:          INDEX
            DENSITY:          800 BPI
             PARITY:          ODD
n           PLEASE MOUNT VOLUME WRITE LOCKED!
o   DEVICE? MM1:
p   *STATUS

    PHASE    : SELECT
    VOLUME # : 2
    ERRORS   : 0

    CURRENT VOLUME  : _MM     :
    CURRENT ACCOUNT : _DM1    :[2,110]

    ELAPSED TIME    : 1 SECONDS
    CPU TIME        : .4 SECONDS
    KCTS            : 64


    *
q   DISMOUNT DEVICE:          _MM1    :
    MOUNT    DEVICE:          _MM     :
                 ID:          BACKUP
               SEQx:          1
            DENSITY:          800 BPI
             PARITY:          ODD
r           PLEASE MOUNT VOLUME WRITE LOCKED!
s   DEVICE? MM0:
    *
t   DISMOUNT DEVICE:          _MM0    :
    *
    $
```

a Type RES in response to the initial BACKUP prompt, and then press the RETURN key. This starts the Restore program which prints a set of dialogue questions.

b Specify a work file name that does not have a .TMP file type such as RESTORE.WKF. BACKUP deletes any work file created during a Restore run that has a temporary file type of .TMP.

c Create the RESTOR.LST file to store the listing file generated from this Restore operation. Restore prints the listing file at your terminal (KB:) if you accept the default.

d Press the LINE FEED key to have Restore transfer the primary index file, located on the last volume of the Backup Set, to the destination disk.

e Accept the MT: default because the files you are restoring reside on magnetic tape.

f Specify the files for Restore to recreate on the output disk. In this case, Restore returns the two files [2,110]BACKUP.BAS and BACKTO.BAS to the RK06 disk.

g Identify the disk to which you plan to copy the files contained on the magnetic tape Backup Set. Type DM1:, which is the device specification of the RK06 disk.

h Have Restore begin processing the first file it encounters on the Backup Set by accepting the default.

i Type NO to make sure the files that exist on the Backup Set but do not exist on the output disk are not reproduced on the output disk.

j Press the LINE FEED key to have files superseded. That is, you do not want any files on the Backup Set to overwrite the same files on the output disk.

k Accept the default; you do not want any files compared.

l Notice that Restore prints an asterisk prompt after you complete the dialogue. This indicates the program is ready to accept interruption commands. Ignore the prompt at this point; you have no reason to interrupt processing.

m Type BACKUP to specify the correct name of the Backup Set. During the Backup operation (Section 8.5.1), you chose the default to the PLEASE ENTER BACKUP SET NAME question. The name you entered in response to that question is the name you must enter at this point in the Restore operation. (Whatever name you enter in response to the WORK FILE NAME question, Restore and Backup use for the default response to PLEASE ENTER BACKUP SET NAME.) After you press the LINE FEED key in response to the next two questions, Restore prints a summary of the label information contained on the Backup Set.

n Note that the message asks you to mount the last volume of the Backup Set, the one containing the primary index file. Mount it without the write-enable ring. If you have not created an index file with Loadindex, created an auxiliary index file by copying the work file using PIP.SAV, or have not given the work file a file type other than .TMP during the Backup operation, you must repeatedly load the last backup volume for each Restore operation. You can use Loadindex to create an easily accessible index and thus avoid this inconvenience.

o Enter the device specification of the last volume of the Backup Set. Restore needs the index that is on the last volume to begin the Restore operation. (Note that SEQ #: tells you which volume you should load; the index volume is the correct volume to load in this case.) When you press the RETURN key, Restore prints the asterisk prompt again.

p Respond to the asterisk prompt by typing STATUS, which summarizes the operational status of the Restore run. The summary indicates that Restore:

- Is in the select PHASE
- Is processing the last volume of the Backup Set
- Has encountered no errors
- Is being performed from a magnetic tape Backup Set
- Is processing account [2,110]
- Has consumed 1 second of elapsed time, .4 seconds of CPU time, and 64 kilo-core-ticks

After printing the summary, Restore returns to normal processing, as the asterisk prompt indicates.

q Note the dismount summary that Restore prints when it finishes pro cessing the Backup Set volume that contains the primary index file.

r Notice that Restore asks you to mount the remaining volume in the Backup Set.

s Type the device specification of the device on which you have mounted the second volume of the Backup Set. After you press RETURN, Restore selects the specified files from the mounted volume and transfers them to the output disk.

t Note that after Restore prints the dismount message for the second volume, it returns control to your keyboard monitor.

8.6.2 Listing File – Restore

The listing file summarizes the phases of the Restore operation and provides directory information on the restored files. The letters in the example are keyed to the description that follows:

```
Backup Run Listing

   Restore from '_MT    :' to '_DM     :[*,*]

   Run started on 08-Sep-82 at 11:57 AM

   Work-File is _SY     :[1,110]RESTOR.WKF/MO: 256

a  Transfer    : [2,110]BACKUP.BAS,[2,110]BACKTO.BAS

   Begin at    : [*,*]*.*

   Superseded  : <none>

   Compare     : <none>

   PHASE    : LIST COMPLETE
   ERRORS   : 0
```

(continued on next page)

```
b  ELAPSED TIME : 1 SECONDS
   CPU TIME     : .5 SECONDS
   KCTS         : 72


   PHASE    : MOUNT / DISMOUNT COMPLETE
   ERRORS   :0

c  ELAPSED TIME : 61 SECONDS
   CPU TIME     : .9 SECONDS
   KCTS         : 160

   PHASE    : INDEX LOAD COMPLETE
   ERRORS   : 0

   CURRENT VOLUME : _MM1  :

d  ELAPSED TIME : 1 SECONDS
   CPU TIME     : .7 SECONDS
   KCTS         : 107

   PHASE    :  SELECT COMPLETE
   VOLUME # : 2
   ERRORS   : 0

e  ELAPSED TIME : 2 SECONDS
   CPU TIME     : .8 SECONDS
   KCTS         : 128


   PHASE    : MOUNT / DISMOUNT COMPLETE
   ERRORS   : 0

f  ELAPSED TIME : 24 SECONDS
   CPU TIME     : .8 SECONDS
   KCTS         : 144


   PHASE    : TRANSFER COMPLETE
   VOLUME # : 1
   ACCOUNTS : 1
   FILES    : 2
   BLOCKS   : 289
   ERRORS   : 0

g  CURRENT VOLUME  : _MM0   :
   CURRENT FILE    : _MM0   :

   ELAPSED TIME : 50 SECONDS
   CPU TIME     : 1.5 SECONDS
   KCTS         : 254


   PHASE    : MOUNT / DISMOUNT COMPLETE
   ERRORS   : 0

h  ELAPSED TIME : 15 SECONDS
   CPU TIME     : .2 SECONDS
   KCTS         : 32
```

(continued on next page)

```
   Backup Set Name : BACKUP                Backup Device : _MM     :
   Volume Sequence # : 1                   Owner : [1,110]
i  Creation Date : 08-Sep-82               Expiration Date : 08-Sep-83
   Density : 800 BPI                       Parity : ODD


     Account : [2,110]                     Quota : 0
     Clustersize : 4



                                  Creation          Access
j    Name  Ext  Size  Prot     Date     Time        Date      Clu  RTS      TSCD
     BACKUP.BAS   121   42  23-Aug-82 09:56 AM  23-Aug-82     8 BASIC      T
     BACKTO.BAS   168   42  15-Aug-82 04:36 PM  15-Aug-82     8 BASIC      T

k    Account total of 289 blocks in 2 files on account [2,110]
l    ***** Account continued on next Volume *****


m  Volume total of 289 blocks in 2 files in 1 accounts on volume _MM    :

   Backup Set Name : BACKUP                Backup Device : _MM     :
   Volume Sequence # : 2(Index)            Owner : [1,110]
n  Creation Date : 08-Sep-82               Expiration Date : 08-Sep-83
   Density : 800 BPI                       Parity : ODD


o  <NO TRANSFER SELECTED>

p  Volume total of 0 blocks in 0 files in 0 accounts on volume _MM   :


   RUN total of 289 blocks in 2 files on 0 accounts on 2 Volumes
q  RUN total of 0 errors

   PHASE    : LIST COMPLETE
   VOLUME #    : 1
   ERRORS      : 0

r  ELAPSED TIME : 3 SECONDS
   CPU TIME     : 1.1 SECONDS
   KCTS         : 162
```

The Restore listing file:

a Summarizes the Restore dialogue that indicates you are restoring two files in account [2,110], BACKUP.BAS and BACKTO.BAS. The summary also shows that Restore starts processing at the beginning of the file, does not supersede any files, and does not compare any files.

b Supplies accounting statistics about writing the dialogue summary into the listing file.

c Summarizes the mount/dismount phase, during which you mounted the backup index volume.

d Describes the index load phase.

e Summarizes the selection phase statistics.

f Indicates that Restore finished selecting data from the backup index volume and requested that it be dismounted. The summary also describes the various times required to complete the dismount operation and mount the other volume in the Backup Set.

g Describes the transfer phase which shows that two files selected for transfer were on volume 1.

h Displays the mount/dismount summary, which is caused by the dismount request that occurs at the end of the transfer.

i Lists the label information for volume 1 after displaying each operational phase.

j Prints the directory information for each file and account for that volume. This section, like that in the Backup listing, includes the account, quota, and cluster size for each affected account and lists each restored file by name. The block size, protection code, date and time of creation, date of last access, cluster size, and associated run-time system are listed for each file. The TSCD column contains a T for each restored file because you specified only that they be transferred, not superseded (S), compared (C), or deleted (D).

k Indicates that a total of 289 blocks in 2 files were transferred for account [2,110].

l States that account [2,110] is continued on the second volume.

m Displays the block, file, and account summary for the entire volume.

n Lists label information for volume 2.

o Notifies you that no files or accounts were transferred. The account message, displayed at k, does not appear for volume 2 because no accounts were transferred from volume 2.

p Always displays the volume summary information for each volume of the Backup Set even when the totals are zero.

q Reports the total statistics for the run.

r Supplies statistics that describe the listing phase that generated the listing file.

8.7 Loading the Index File – Example

Loadindex allows you to copy a primary index file from a Backup Set to a RSTS/E formatted disk. To perform the transfer, run the BACKUP program, type Loadindex in response to the initial BACKUP prompt, and answer the resulting questions. During the dialogue, you specify the name and location of the Backup Set and the file into which the program is to copy

the index. BACKUP then prompts you to mount the Backup Set index file, copies the index when the device is ready, and requests the dismount of the index file volume. Finally, BACKUP prints a listing file that summarizes the Loadindex operation.

When you perform a series of restores, you must have access to the index file located on the last volume of the Backup Set. If you have not created an index file with Loadindex, created an auxiliary index file by copying the work file using PIP.SAV, or have given the work file a file type other than .TMP during the Backup operation, you must repeatedly load the last backup volume for each Restore operation. You can use Loadindex to create an easily accessible index and eliminate this inconvenience.

A discussion follows both the Loadindex example in Section 8.7.1 and the Loadindex listing file example in Section 8.7.2.

8.7.1 Terminal Printout – Loadindex

Log on to the system under account [1,110] and run BACKUP. The letters in the sample printout are keyed to the description that follows:

```
   $ RUN $BACKUP
   BACKUP V8 RSTS V8 TIMESHARING


a  BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? LOA

b  WORK FILE NAME<_SY:[  1,110]B08SEP.TMP> ? LOADIN.WKF/MO:256

c  LISTING FILE<_KB:> ? LOADIN.LST

d  FROM DEVICE<_MT:> ? (RET)

e  TO FILE<_SY:[  1,110]BIND08.IND> ? BACIND.IND
f  *
g  PLEASE ENTER BACKUP SET NAME<LOADIN> - BACKUP
h  PLEASE ENTER DENSITY IN BPI<800> - (RET)
i  PLEASE ENTER THE PARITY <ODD> - (RET)
   MOUNT    DEVICE:          _MT     :
               ID:            BACKUP
             SEQ#:            INDEX
j          DENSITY:           800 BPI
            PARITY:           ODD
           PLEASE MOUNT VOLUME WRITE LOCKED!
l  DEVICE? MM1:
m  *
n  DISMOUNT DEVICE: _MM1:
   *
o  $
```

a Run the Loadindex dialogue by typing LOA or LOADINDEX in response to the initial BACKUP question. BACKUP understands only the first three letters of the option.

b Type LOADIN.WKF/MO:256 to store the work file on the public disk structure. Giving the specified file a file type other than .TMP allows you to use the work file as an auxiliary index file at some later date. However, if you accept the default or specify a work file with a .TMP file type, BACKUP deletes the file at the end of the Loadindex run. The MODE 256% switch enables random data caching, which facilitates data transfers within the LOADIN.WKF work file. For an explanation of the MODE 256% switch, refer to the *RSTS/E Programming Manual.*

c Specify LOADIN.LST as the file into which BACKUP prints the listing file. The listing file is displayed on your terminal if you accept the default.

d Accept the MT: default by pressing the RETURN key because the index resides on magnetic tape.

e Have BACKUP copy the backup index to the file BACIND.IND.

f Ignore the asterisk prompt, which indicates that BACKUP is ready to accept interruption commands.

g Respond by typing the Backup Set Name, BACKUP.

h Accept the default density of 800 BPI because that is the density of the magnetic tape.

i Press the RETURN key to accept the ODD parity default setting for the magnetic tape.

j Check the magnetic tape label information summary.

k Make sure the magnetic tape is mounted write-locked and is at its load point.

l Type MM1:, which is the mnemonic name and unit number of device on which the magnetic tape is mounted.

m Type a BACKUP interrupt command if you want to temporarily suspend processing at this point. Because the operation is running smoothly, ignore the asterisk prompt. Before BACKUP issues the dismount message at n, it copies the index file to the specified area.

n Dismount the Backup index volume by typing the mnemonic name and unit number (followed by a colon) of the device on which the magnetic tape is mounted.

o Notice that the dollar prompt ($) indicates that the Loadindex operation has terminated and you are now under the control of the DCL keyboard monitor.

8.7.2 Listing File – Loadindex

The Loadindex listing file summarizes the Loadindex dialogue, provides statistics on each phase, and lists the operations performed on each selected file. The letters in the file are keyed to the descriptions that follow:

```
Backup Run Listing

   Loadindex from '_MT     :

   Run started on 08-Sep-82 at 12:03 PM

a  Work-File is _SY     :[1,110]LOADIN.WKFMO: 256

   To file  : _SY     :[1,110]BACIND.IND


   PHASE    : LIST COMPLETE
   ERRORS   : 0

b  ELAPSED TIME : 1 SECONDS
   CPU TIME     : .5 SECONDS
   KCTS         : 72


   PHASE    : MOUNT / DISMOUNT COMPLETE
   ERRORS   : 0

c  ELAPSED TIME : 60 SECONDS
   CPU TIME     : .9 SECONDS
   KCTS         : 160


   PHASE    : INDEX LOAD COMPLETE
   ERRORS   : 0

d  CURRENT VOLUME  : _MM1    :

   ELAPSED TIME : 1 SECONDS
   CPU TIME     : .3 SECONDS
   KCTS         : 46


   PHASE    : MOUNT / DISMOUNT COMPLETE
   ERRORS   : 0

e  ELAPSED TIME : 12 SECONDS
   CPU TIME     : .2 SECONDS
   KCTS         : 32
```

The Loadindex listing file:

a Summarizes the Loadindex dialogue described in Section 8.7.1. The summary shows the date and time of the Loadindex operation, the work file name, the name of the output file, and that the index resides on magnetic tape.

b Provides time and error statistics about writing the dialogue summary into the listing file.

c Lists the statistical details concerning the mount phase of the Backup Set index volume.

d Describes which volume contains the index, the times required to complete the load, and the errors encountered during the index load phase.

e Summarizes the dismount of the Backup Set volume.

8.8 Listing the Index File – Example

List initiates a set of dialogue questions like the other BACKUP modes. After you answer these questions, BACKUP creates a copy of the Backup Set index file, an index created by Loadindex, or, if specified during the Backup operation, an auxiliary index file. These files contain a directory listing of the Backup Set and a log of any errors encountered during the Backup operation. Before it can copy the index file, BACKUP asks for the index name and location and then asks where you want the index listed. As options, you can either list the index file at your terminal or place it in another file. Like the other BACKUP modes, a summary of the dialogue is written to the listing file.

The example in Section 8.8.1 describes how to generate a list of an auxiliary index file.

8.8.1 Terminal Printout – List

Log on to your system, run BACKUP, and then respond to the dialogue questions as shown. The letters in the printout are keyed to the description that follows:

```
   $ RUN $BACKUP
   BACKUP V8 RSTS V8 TIMESHARING


a  BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? LIS
b  WORK FILE NAME_SY:[  1,110]B08SEP.TMP> ? LIST.WKF/MO:256
c  LISTING FILE<_KB:> ? LIST.LST
d  INDEX FILE<PRIMARY> ? BACKUP.WKF
e  *
f  $
```

a Type LIS and press RETURN in response to the first BACKUP question. Because BACKUP recognizes only the first three characters of each option, you can type either LIS or LIST to begin the List dialogue.

b Specify LIST.WRK as the work file and attach the MODE 256% switch to optimize file transfer. Refer to the *RSTS/E Programming Manual* for a description of MODE 256%.

c Have BACKUP copy the listing file to LIST.LST. If you accept the default (KB:), BACKUP prints the index file at your terminal

d Type BACK.WRK to indicate which index file you want BACKUP to copy. In this case, specify the BACK.WRK file, which you created as an auxiliary index file during the Backup operation in Section 8.5.1. Because you use a file type other than .TMP, namely .WRK, when you specified the work file name, BACKUP created BACK.WRK as a permanent auxiliary index file. After you complete the dialogue, BACKUP copied the BACK.WRK file to the listing file, LIST.LST. If, however, you accept the default, BACKUP prints the FROM DEVICE question. In response, specify the device type on which the primary index volume is mounted. Because BACK.WRK is not the primary index file, you accepted the default and the question did not appear.

e Respond to the asterisk prompt by pressing the RETURN key. The asterisk appears after you complete the dialogue and indicates that BACKUP is ready to accept interruption commands. After your response, BACKUP copies BACK.WRK to the LIST.LST listing file. The Ready prompt appears when the List operation ends.

f The dollar prompt ($) appears when the List operation ends.

8.8.2 Listing File – List

The List listing file summarizes the List dialogue, provides statistics on the each phase of the operation, and prints a copy of the Backup index file. The example that follows is the listing file for the List operation described in Section 8.8.1. The letters in this listing file are keyed to the description that follows:

```
Backup Run Listing

   List

a  Run started on 08-Sep-82 at 12:08 PM

   Work-File is _SY    :[1,110]LIST  .WKF/MO: 256

   PHASE    : LIST COMPLETE
   ERRORS   : 0

b  ELAPSED TIME : 1 SECONDS
   CPU TIME     : .5 SECONDS
   KCTS         : 72

   PHASE:   : INDEX LOAD COMPLETE
   ERRORS   : 0

c  ELAPSED TIME : 2 SECONDS
   CPU TIME     : .4 SECONDS
   KCTS         : 55
```

(continued on next page)

```
   Backup Set Name : BACKUP                Backup Device : _MM     :
   Volume Sequence # : 1                   Owner : [1,110]
d  Creation Date : 08-Sep-82               Expiration Date : 08-Sep-83
   Density :  800 BPI                      Parity : ODD


     Account : [1,110]                     Quota : 0
     Clustersize : 4


                                    Creation              Access
    Name  Ext    Size  Prot        Date     Time          Date       Clu  RTS    TSCD
   SAVE  .DOC     10    60   10-Jul-82 05:01 PM    10-Jul-82        8 RSX    *T
   OPEN  .TMP      0    60   08-Sep-82 10:46 AM    08-Sep-82        4 BAS4F
   ADDR  .DAT   2000    60   08-Sep-82 10:46 AM    08-Sep-82        4 BAS4F   T


                  Errors :
        DATA UNRELIABLE - FILE OPENED BY ANOTHER USER on FILE
e                  Total of 1 error encountered on FILE
   RT11  .RTS     20C   60   31-Aug-82 04:39 PM    31-Aug-82        4 RSTS    T
   SAVE  .RNO       9   60   10-Jul-82 04:59 PM    10-Jul-82        8 TECO    T
   UTILTY.BAS      47   60   23-Nov-81 10:44 AM    23-Nov-81        8 BASIC   T
   DIRECT.BAS      56   60   21-Jul-82 11:24 AM    21-Jul-82        8 BAS4F   T

     * Attributes associated with this file

    Account total of 2142 blocks in 7 files on account [1,110]
        Total of 1 error encountered on ACCOUNT

    Account : [2,110]                      Quota : 0
    Clustersize : 4


                                    Creation              Access
    Name  Ext   Size   Prot        Date     Time          Date      Clu   RTS     TSCD
   BACKUP.BAS    121    42   23-Aug-82 09:56 AM    23-Aug-82        8  BASIC    T
   BACFRM.BAS    158    42   15-Aug-82 04:49 PM    15-Aug-82        8  BASIC    T
   BACKTO.BAS    168    42   15-Aug-82 04:36 PM    15-Aug-82        8  BASIC    T
f  MASTER.DAT   6904    60   08-Sep-82 10:55 AM    08-Sep-82        4  BAS4F    T C
               (of 9000)

       ***** File continued on next Volume *****


     Account total of 7351 blocks in 4 files on account [2,110]
     ***** Account continued on next Volume *****


g  Volume total of 9493 blocks in 11 files in 2 accounts on volume MM    :
     Total of 1 error encountered on VOLUME


   Backup Set Name : BACKUP                Backup Device : _MM     :
   Volume Sequence # : 2(Index)            Owner : [1,110]
h  Creation Date : 08-Sep-82               Expiration Date : 08-Sep-83
   Density :  800 BPI                      Parity : ODD


   Account : [2,110]          Quota : 0
   Clustersize : 4
```

(continued on next page)

```
                                    Creation              Access
     Name  Ext    Size  Prot      Date      Time          Date      Clu  RTS    TSCD
i    MASTER.DAT   2096   60    08-Sep-82 10:55 AM   08-Sep-82        4 BAS4F  T CD
                  (of 9000)

      Account total of 2096 blocks in 1 files on account [2,110]

j   Volume total of 2096 blocks in 1 files in 1 accounts on volume MM    :

    RUN total of 11589 blocks in 11 files on 2 accounts on 2 Volumes
k   RUN total of 1 errors


    PHASE          : LIST COMPLETE
    VOLUME #       : 2
    ACCOUNTS       : 2
l   FILES          : 11
    BLOCKS         : 11589
    ERRORS         : 0

    ELAPSED TIME : 2 SECONDS
m   CPU TIME      : 1.7 SECONDS
    KCTS           : 252
```

The List listing file:

a Summarizes the List dialogue, which includes the name of the operation, the run date and time, and the work file name.

b Provides statistics for the listing phase about writing the List dialogue summary into the listing file. Notice that Backup rounds off elapsed time to whole seconds and CPU time to tenths of seconds. It is possible, therefore, for the elapsed time to read zero while there is CPU time indicated.

c Summarizes the index load phase, which prints the elapsed time, CPU time, and the kilo-core ticks needed to load the index into the listing file.

d Prints the Backup directory information extracted from the index file after it completes the List dialogue summary and reports on the various phases of the List operation. The directory report begins by listing the label information for the first volume of the Backup Set. It then displays a list of the files for each account on the directory. At the end of each account report, BACKUP prints the total number of blocks used by the files listed and the number of errors encountered on the account. In addition to supplying the file name, BACKUP includes in the account report the file block size, protection code, date and time of creation, date of last access, the cluster size, and the associated run-time system. The TSCD column contains a T for each restored file, an S for each superseded file, a C for a compared file, and a D for each deleted file. An asterisk beside a letter in the TSCD column indicates that the file has attributes associated with it. Zero-length files, such as OPEN.TMP in account [1,110], are not processed and thus no letter is assigned to them in the TSCD column. Note that any errors in the dialogue sequence appear in the index.

e Prints the DATA UNRELIABLE message immediately after the file that caused the error. After enumerating the error messages, BACKUP prints the total number of errors detected in the file.

f Summarizes the files contained in account [2,110].

g Prints at the end of the account report the total number of blocks, files, and accounts on the entire volume.

h Prints label information found on the second volume of the Backup Set.

i Prints the account summary for volume 2. If there is more than one volume in the Backup Set, as there is in this example, BACKUP prints account and file information in the same format for the other volumes in the set.

j Summarizes the account that was transferred for the second volume of the set. Because there is only one account report, the volume and account totals are equal.

k Displays a summary of the blocks consumed by all the files on both volumes of the Backup Set. This completes the account and file statistics.

l Informs you that there are two volumes in the Backup Set, two accounts in the directory, and 11 files in the two accounts that require 11,589 blocks of disk space. In addition, an item in the list phase report indicates that there were no errors encountered during the printing of the list phase summary.

m Prints the time required to complete the list phase.

Chapter 9
SAVE/RESTORE System Program

The RSTS/E SAVE/RESTORE system program is a disk backup and copy utility that has four operational functions:

- SAVE backs up a disk to tape or disk.
- RESTORE recreates a disk from tape or disk.
- IMAGE makes a copy of a disk.
- IDENTIFY extracts label information from a SAVE/RESTORE volume or RSTS/E disk.

9.1 When to Use SAVE/RESTORE

SAVE/RESTORE and BACKUP perform similar system functions but differ in their objectives. SAVE/RESTORE provides a nonselective, fast-volume backup and copy capability that requires few operator responses during the operational dialogue. Unlike BACKUP, SAVE/RESTORE processes entire volumes only and does not allow selective file transfers or file deletions. When SAVE/RESTORE encounters bad blocks, it does not require any operator intervention. For these reasons, use SAVE/RESTORE when you need to:

1. Create a fast, reliable copy of an entire RSTS/E disk
2. Back up files that are larger than 65,535 blocks
3. Make a fast copy to a disk containing bad blocks that may not be reflected in the BADB.SYS file
4. Make a recovery medium (bootable) for a system disk in case of catastrophic errors

9.2 Definitions of SAVE/RESTORE Terms

To understand this chapter, you should be familiar with these terms:

- SAVE format: The format of the output written by a SAVE operation and read by a RESTORE operation.
- SAVE Set: The set of magnetic tapes or disks created by a SAVE operation. A SAVE Set must be composed entirely of disks or tapes, not a combination of the two device types. However, you can mix different drive types within one set, for example, two RK05s and one RK06.
- SAVE volume: One of the magnetic tapes or disks of a SAVE Set.
- SAVE Set Name: One to six alphanumeric characters used to identify a SAVE set. By default, the SAVE set name is the same as the Pack ID from which it was created. However, you can specify another name for the SAVE Set.
- LIKE Disks: Like disks are units of the same device size (that is, the same number of data blocks). SAVE/RESTORE considers the following devices alike:
  - Two RP06 disks
  - An RM02 and an RM03
  - An RP04 and an RP05
  - An RK05 (RK05J) and one unit of an RK05F

  Therefore, if you copy an RM02 to a SAVE Set, that set can later be restored to either an RM03 or another RM02.

9.3 Running SAVE/RESTORE

You use the SAVE/RESTORE program off line while running the system initialization code INIT.SYS. In reply to the INIT OPTION prompt, type SAVRES and press the RETURN key. SAVE/RESTORE asks for the current date and time and then responds with the SAV/RES FUNCTION prompt. For example:

```
Option: SAVRES

DD-MMM-YY? 14-MAR-83
HH:MM? 12:15

SAV/RES Function:
```

NOTE

You must run SAVE/RESTORE off line to copy (SAVE or IMAGE) or restore (RESTORE) the system disk with which you are currently running on line.

You can also use SAVE/RESTORE on line, during timesharing. When you do, you must not logically mount the RSTS/E disk(s) on which

SAVE/RESTORE operates. This ensures the integrity of the data on the disk(s). To run SAVE/RESTORE on line, type RUN $SAVRES, and press the RETURN key. SAVE/RESTORE then prints its program prompt:

```
RUN $SAVRES(RET)
SAV/RES Function:
```

After the SAV/RES FUNCTION prompt appears, as a result of running SAVE/RESTORE either off line or on line, you are under the control of SAVE/RESTORE and are ready for backup processing.

Next, select from Table 9–1 one of the four operational functions:

- SA[VE]
- RE[STORE]
- IM[AGE]
- ID[ENTIFY]

Unless you need to exit SAVE/RESTORE (by pressing LINE FEED or typing CTRL/Z) or need further help (by typing HE[LP] or pressing the RETURN key), type one of the four functions on your terminal in response to the SAV/RES FUNCTION prompt and press the RETURN key. For example:

```
SAV/RES Function: SAVE(RET)
```

This starts the SAVE/RESTORE program, which immediately prints the first of several dialogue questions. Each of the four sections 9.5.5 through 9.5.8 contains a description of one of the operational functions.

Table 9–1: SAVE/RESTORE Functions

Function	Description
SA[VE]	Creates a copy of a RSTS/E file-structured disk. The SAVE function backs up to disk or tape.
RE[STORE]	Recreates a RSTS/E file-structured disk from a SAVE Set.
IM[AGE]	Copies a RSTS/E file-structured disk to a like disk.
ID[ENTIFY]	Prints label information and other volume characteristics of a SAVE volume or a RSTS/E file-structured disk.
HE[LP]	Prints a table of SAVE/RESTORE functions.
(LF)	Exits the SAVE/RESTORE program. If you run SAVE/RESTORE with a RUN command, pressing LINE FEED returns you to your keyboard monitor. If you are running SAVE/RESTORE off line with INIT.SYS, pressing LINE FEED returns you to the INIT OPTION prompt.
(RET)	Displays the "Type HELP for help" message on your terminal.
(CTRL/C)	Produces the same result as pressing the LINE FEED key.
(CTRL/Z)	Produces the same result as pressing the LINE FEED key.
other	Produces the same result as pressing the RETURN key.

Your response to the SAV/RES FUNCTION prompt determines the type of dialogue SAVE/RESTORE performs. You can respond to the prompt in several ways:

1. With the function name, such as SAVE or RESTORE, or by pressing a key such as RETURN, as shown in Table 9–1. If you enter only a function, the full dialogue follows. See Sections 9.5.5 through 9.5.8.
2. With the function name and various switches. For example:

   ```
   SAV/RES Function: SAVE/EXP:10-JUL-83
   ```

   If you include switches with the function, the questions follow only for the devices to be used and the unspecified switches. See Section 9.4.
3. With a full function command line, such as:

   ```
   SAV/RES Function: IMAGE DM1:=DM0:MYPAK/VER/NOERR
   ```

   If you enter a complete command line, SAVE/RESTORE does not ask any dialogue questions. See Section 9.5.9.

SAVE/RESTORE prints messages on your terminal to tell you its status. Some of the messages are intended for your information; others are error conditions that may require your attention during the dialogue, mount, transfer, or verification phases of SAVE/RESTORE.

When the program encounters an error during a dialogue or mount operation, it asks you to correct the problem and then to retype the appropriate command. Such errors never affect the integrity of data being transferred.

Errors that occur while SAVE/RESTORE is transferring or verifying data can, however, jeopardize data. In this case, SAVE/RESTORE resolves the problem, aborts the run, or asks you to continue or to abort the program. The program gives you information to help you protect your data by printing informational messages and error messages on your terminal.

Finally, SAVE/RESTORE prints an optional summary report about SAVE, RESTORE, or IMAGE operations at the end of the run. This report describes:

- The operation performed
- Device information
- Start date and time statistics
- Run total statistics

The run statistics include the number of errors encountered during the operation.

9.4 SAVE/RESTORE Switches

You can include SAVE/RESTORE switches in a function response or in a full function command line. There are two types of switches in Table 9–2:

1. The expiration and verification switches correspond to questions in the SAVE/RESTORE dialogue and have default settings. If you do not include the switches in a function response, the dialogue asks you for that information. For example, if you do not include the /EXPIRATION switch, the dialogue asks:

   ```
   Expiration Date<DD-MMM-YY>?
   ```

   If you include one of the switches in a function response, SAVE/RESTORE does not ask the corresponding question but does ask the question for the switch you did not specify. For example, if you include the /EXPIRATION switch but not /VERIFY, SAVE/RESTORE does not prompt you for an expiration date but does ask the verification question.

   If you do not include the switches in the full function command line, SAVE/RESTORE assumes the default settings. For example, if you do not include /VERIFY, SAVE/RESTORE assumes /NOVERIFY. This means if you do not explicitly include the /VERIFY switch in a full function command line, SAVE/RESTORE does not compare the data transfer.

2. The /STATS and /ERROR switches have no corresponding dialogue questions but do have default settings. Therefore, if you do not specify these switches in a function response or command line, SAVE/RESTORE assumes the default settings.

Table 9–2: SAVE/RESTORE Switches

Switch	Default and Description
EX[PIRATION][:date] NOEX[PIRATION]	EXPIRATION SAVE/RESTORE warns you of any attempt to overwrite the volume(s) any time prior to the specified date. The date argument is the date after which you may overwrite the destination volume. You can use this switch only for a SAVE operation. If you specify /EX[PIRATION] with no date argument, SAVE/RESTORE uses one year after the current date as the the expiration date. If you specify /NOEX[PIRATION], SAVE/RESTORE uses the current date as the expiration date.
VE[RIFY] NOVE[RIFY]	NOVERIFY You can use this switch only for a SAVE, RESTORE, or IMAGE operation. If you specify the /VE[RIFY] switch, SAVE/RESTORE compares the volume(s) to ensure the transfer was accurate.

(continued on next page)

Table 9–2: SAVE/RESTORE Switches (Cont.)

Switch	Default and Description
ST[ATS] NOST[ATS]	STATS SAVE/RESTORE automatically prints a summary report at the end of a run, unless you specify the /NOST[ATS] switch with either a function response or a full function command line. Section 9.8 contains a description of the summary report. You can make /NOSTATS the default by applying patches to INIT.SYS and/or to $SAVRES.SAV. This causes SAVE/RESTORE to omit a summary report after completing a transfer. If you want a summary report printed while /NOSTATS is the default, however, append the /STATS switch. For that particular operation, SAVE/RESTORE prints a summary report. Refer to the *RSTS/E Maintenance Notebook* for instructions on how to install the patches.
ER[ROR] NOER[ROR]	ERROR If you do not include the /ERROR switch and a non-fatal error occurs during a SAVE/RESTORE run, SAVE/RESTORE prints an error message and continues processing. This is the default condition. You must explicitly specify the /NOERROR switch if you want SAVE/RESTORE to print an error message and terminate as a result of any of the errors described in Table 9–12. As an alternative, you can apply patches to INIT.SYS to make /NOERROR the default. With the patches installed, however, you can override the /NOERROR default for a particular operation by specifying /ERROR. Refer to the *RSTS/E Maintenance Notebook* for instructions on how to apply the SAVE/RESTORE /NOERROR patches.

9.5 SAVE/RESTORE Dialogue

The SAVE/RESTORE dialogue begins after you run the SAVE/RESTORE program and respond to the SAV/RES FUNCTION prompt. If you type SAVE, RESTORE, IMAGE, or IDENTIFY, the program asks a set of dialogue questions, which have either a short or a long form. The short form appears automatically while the long form prints only if you press the RETURN key in response to the short form question. The long form gives you information to help you answer the questions. Tables 9–5 through 9–8 show the text of each of the long form questions in quotation marks (").

Some of the questions have default values which the program prints in angle brackets. You can select the default value by pressing the LINE FEED key or by typing the proper default response. If you press LINE FEED in response to a question that has no default, SAVE/RESTORE repeats the question.

There are two valid responses to a SAVE/RESTORE program that can change its operating status:

- CTRL/Z
- CTRL/C

If you type CTRL/Z at any point in the dialogue, SAVE/RESTORE returns to the previous question. A CTRL/C response aborts the run immediately and returns you to the INIT OPTION prompt if you are running off line under the control of the INIT.SYS program or to your keyboard monitor prompt if you are running SAVE/RESTORE on line. SAVE/RESTORE automatically returns to the SAV/RES FUNCTION prompt after the execution of a specified function when you are operating either on line or off line. Exit from the INIT version of SAVE/RESTORE to the OPTION prompt by pressing LINE FEED. Type CTRL/Z to the SAV/RES FUNCTION prompt to exit the on-line version of SAVE/RESTORE.

9.5.1 SAVE Volumes

You can perform SAVE/RESTORE operations on disks and tapes. The SAVE Set, a set of tapes or disks created from a SAVE operation, cannot include both disks and tapes. The SAVE Set can, however, include one or more volumes of the same class. In other words, a tape volume SAVE Set could consist of two MM devices and one MT device or one RP04 and two RK05 disks. SAVE/RESTORE does not let you mix disks and tapes in the same SAVE Set.

A legal SAVE disk device is any disk supported by RSTS/E that meets the following requirements:

1. All disks must be formatted (that is, contain sector and track information).
2. All disks used by the on-line version of SAVE/RESTORE must have a valid RSTS/E file structure; that is, the disk device must be formatted and cleaned by DSKINT prior to its use. The one exception to this rule is the disk you have used as a Save Set volume. It does not have the RSTS/E file structure (and cannot be logically mounted) but can be used for SAVE/RESTORE operations.
3. The first 16 blocks plus the Device Cluster Size blocks on a disk must not contain any bad blocks.
4. A RSTS/E output disk can contain no more than 161 bad clusters (the RSTS/E limit).

Magnetic tape is the only valid SAVE/RESTORE tape medium; DECtape and cassette are unacceptable. The following statements define what constitutes a valid tape in the SAVE/RESTORE package:

1. The magnetic tape must be long enough to contain:
   - The SAVE Set label and header information
   - A bootstrap
   - INIT.SYS that is on the first volume only
   - Two copies of the RSTS/E disk file SATT.SYS
   - At least 16 blocks of data from the RSTS/E disk

2. A bad block is considered to be the end of the tape volume during the SAVE operation.

3. The default density is the lowest density allowed by the drive on which the first volume is mounted. Subsequent volumes cannot be written at any other density. Therefore, make sure the density used on the first volume is compatible with all other tape drives you plan to use.

9.5.2 Device Specifications

SAVE/RESTORE requires you to use device specifications as responses to several dialogue questions. The general format is:

<device>[:[<id>]][<switch(es)>]

Device is a two-character device type (for disk either "Dx" or "Rx" is acceptable), followed by a single digit unit number in the range 0 to 7. The identification can be either a SAVE Set Name or a disk pack identification while the switches are any of those from Table 9–3.

Table 9–3: SAVE/RESTORE Device Specification Switches

Switch	Description
/SCR[ATCH]	The /SCRATCH switch is legal only on an output volume device specification. If you use the /SCRATCH switch, SAVE/RESTORE by-passes most volume label checking. The program always checks the label to see if an output volume contains a SAVE Set written at the current density and ensures that it is not a volume from the SAVE Set currently being written. It also checks destination disks for bad block information.
/DEN[SITY]:[800] [1600]	The /DENSITY switch is legal only for magnetic tape. With this switch, you can specify the density at which the tape is to be written or read. If SAVE/RESTORE cannot read the tape at the specified density, it tries to read the tape at the other legal density setting.

9.5.3 Checking the Input Volume

SAVE/RESTORE checks the input volume before transferring any actual data and warns you of any problem it foresees in saving, restoring, or copying the volume. All of the input volume error messages that are printed at this point are warning messages only; you can recover from them by following appropriate procedures. The error messages that result from a problem with an input volume are listed and described separately in Table 9-4.

Table 9–4: SAVE/RESTORE Input Volume Error Messages

Message and Meaning
%%% Input disk has only nn% free clusters. Mount it anyway <NO> The input disk has very few clusters that are not allocated. There must be at least as many good pack clusters on an output RSTS/E disk as there are clusters to be transferred from the original input disk. This message indicates that you may encounter problems copying or restoring the disk. Type NO or press the LINE FEED key to return to the previous device prompt. If you type YES, SAVE/RESTORE proceeds with the SAVE or IMAGE operation. In the case of an IMAGE copy, the program also checks the output volume to see if the transfer can be made (see Section 9.5.4). No further checking takes place for a SAVE operation. At this point, you may want to free up space on the input disk by deleting unnecessary files. This increases your chances of eventually completing a successful RESTORE.
%%% This is not the correct volume. The Pack ID or Save Set Name you included with the input device specification does not match the name already written on the volume. If SAVE/RESTORE finds that the names do not match, it prints the warning message and then issues the input device prompt again. A dismount request follows the warning message and lists useful information from the volume label of a SAVE Set or a RSTS/E file-structured disk.
%%% This volume has no label. The input volume is neither a RSTS/E disk nor a SAVE Set. SAVE/RESTORE prints the input device prompt again after issuing the warning message.
%%% Input disk should be "CLEANed". You removed a disk from a drive without logically dismounting it and then attempted to remount it. Perform the clean operation with the ONLCLN program or the REFRESH option of INIT. For information on ONLCLN, refer to Section 7.6. Refer to the *RSTS/E System Generation Manual* for a description of REFRESH.

9.5.4 Checking the Output Volume

SAVE/RESTORE checks the output volume before saving, restoring, or copying a RSTS/E disk to ensure each operation ends with the least chance of error. After checking the output volume, SAVE/RESTORE displays messages about the expiration date, the label, and the available data space on the destination device. Some of the messages are only for your information; the rest warn you of serious output device problems.

When the expiration date of an output volume, labeled as a SAVE Set, has passed, SAVE/RESTORE prints this series of messages and proceeds to the next dialogue question:

```
*** The volume on dev: is SAVE Set <xxxxxx>
Density          : nnn
Creation date    : day-of-week, dd-mmm-yy
Expiration date  : day-of-week, dd-mmm-yy
```

If you decide not to use the expired date volume, type CTRL/Z to return to the TO DEVICE output question.

If the ouput volume is labeled as a SAVE Set and its expiration date has not passed, SAVE/RESTORE prints this series of messages:

```
*** The volume on dev: is SAVE Set <xxxxxx>
Density                : nnn
Creation date          : day-of-week, dd-mmm-yy
Expiration date: day-of-week, dd-mmm-yy
%%% Its expiration date has not passed
Mount it anyway <NO>?
```

If you type NO or press the LINE FEED key to the mount question, SAVE/RESTORE returns to the previous device prompt. SAVE/RESTORE goes to the next question if you type YES. If you press the RETURN key, SAVE/RESTORE prints:

```
*** This is your last chance to prevent the
*** loss of any data on the output volume.
Mount it anyway <NO>?
```

When the output volume you are using is labeled a RSTS/E disk, SAVE/RESTORE prints the message:

```
*** The volume on dev: is a RSTS disk
*** Pack ID is <xxxxxx>
*** Pack will be reinitialized
Mount it anyway <NO>?
```

If you type NO or press the LINE FEED key, SAVE/RESTORE returns to the previous output prompt. A YES response indicates you want to use the present volume.

If you restore (RESTORE) or copy (IMAGE) a disk, SAVE/RESTORE checks whether the total number of clusters on the output disk minus the known bad blocks is greater than or equal to the number of allocated clusters on the original source disk. If there is not enough space and the transfer cannot be made, it prints the following message and then returns to the previous device prompt:

```
%%% Too many bad blocks on output disk.
```

If, however, the program decides there are very few free blocks, it prints:

```
%%% Only nn% of the output disk clusters are available for relocation.
Mount it anyway <NO>?
```

You must decide whether the number of free clusters is large enough to permit a successful SAVE/RESTORE operation. If you decide it is not, type NO or press LINE FEED to return to the output prompt. Type YES if you want SAVE/RESTORE to continue the operation. Whenever you get this warning message, DIGITAL recommends you use another disk for output. If you do not, there is a greater than normal chance the operation will not succeed. SAVE/RESTORE normally transfers each allocated pack cluster from the original source disk to the same cluster on the output RSTS/E disk. If

SAVE/RESTORE finds the output cluster is bad, it tries to relocate the data. If the cluster size of the item being moved is larger than the pack cluster size, there must be enough contiguous free pack clusters on the disk to accommodate the entire entity cluster. If there is not, SAVE/RESTORE aborts the operation.

If the specified output is not a RSTS/E or a SAVE Set disk, the program prints the new SAVE Set Name and proceeds without further notice. It does not check the output volume label if you specified the /SCRATCH switch, except to determine whether it is a volume of the SAVE Set that is currently being created. It always attempts to recover bad block information from a RSTS/E file-structured and SAVE disk.

9.5.5 Saving a RSTS/E Disk using the SAVE Dialogue

The SAVE function backs up an entire RSTS/E formatted disk to either tape or disk. The output volumes created by this operation are collectively called the SAVE Set and are written by SAVE/RESTORE in the SAVE format. You can recreate the original disk only by using the RESTORE operation. You cannot use other RSTS/E software, such as BACKUP or PIP.SAV to restore a SAVE Set. Note that you must use SAVE if the output medium is different from the input medium.

Once you determine that the disk you are to save and the output SAVE Set device(s) are valid media, type SAVE to start the SAVE dialogue described in Table 9–5.

NOTE

When SAVE/RESTORE creates a SAVE Set, it includes on the output volume an extra set of directory blocks in case some directory blocks are found to be bad during a RESTORE operation. These additional blocks are reflected in the total block count for a SAVE Summary Report. Because the extra blocks are needed only when bad blocks are encountered during a RESTORE operation, it is unnecessary for SAVE/RESTORE to transfer these blocks to the restored volume. Thus, you will notice a discrepancy between the number of blocks transferred for a SAVE operation and the number transferred during a RESTORE. Furthermore, because the extra directory blocks are not transferred to the restored volume, under certain circumstances you may restore your data to one less volume than existed in the SAVE Set.

The Pack ID of the input disk is the default SAVE Set Name of the output volumes. After you answer the FROM RSTS DISK question, SAVE/RESTORE tells you the output SAVE Set Name it intends to use. The general format is:

*** Pack ID/default SAVE Set Name is <xxxxxx>

If you do not want the output volume(s) to have this SAVE Set Name, specify a different Pack ID in the <devspec> response to the TO DEVICE question. After you select the ID and answer the question, SAVE/RESTORE prints:

*** SAVE Set Name is <xxxxxx>

SAVE/RESTORE warns you if the output volume contains a SAVE Set that has an expiration date that has not passed. You can proceed or respecify the output volume (Section 9.5.4).

Table 9–5: SAVE Dialogue Questions

Question	Response and Description
From RSTS Disk?	<devspec> Data is transferred from the specified disk. If you include a Pack ID in the device specification, SAVE/RESTORE checks whether it matches the one on the disk. When it does not, the program prints a warning message (Section 9.5.3) and prints the FROM RSTS DISK prompt again. The disk must be physically but not logically mounted and should be write-locked. Type 2 characters to specify the input device type, followed by a single digit (0–7) to specify the unit number. Include a Pack ID if you want SAV/RES to ensure it matches the one on the volume. This drive should be write-locked.
To Device?	<devspec> All data is transferred from the previously specified input disk to this tape or disk. If you do not specify a SAVE Set Name in the <devspec>, SAVE/RESTORE gives the SAVE Set the same name as the input disk. Type 2 characters to specify the output device type, followed by a single digit (0–7) to specify the unit number. Include a Save Set Name if you wish to override the default. This drive should have a SCRATCH volume mounted, and must be write-enabled.
Expiration Date <DD-MMM-YY>?	<DD-MMM-YY> This is the dialogue form of the /EXPIRATION switch (Section 9.4). If you do not specify a date, SAVE/RESTORE creates an expiration date for the SAVE Set that is one year from the date that it was created. Type the desired expiration date or press the LINE FEED key to accept the date printed.
Verify (Yes or No) <NO>?	Y[ES] or N[O] This is the dialogue form of the /VERIFY switch (Section 9.4). If you respond with YES, the output is then compared to the input at the end of each volume of the SAVE Set. Type 'YES' if you want SAV/RES to compare the input and output volumes at the end of each output volume. Type 'NO' or press the LINE FEED key if you do not want SAVRES to perform this verification.
Proceed (Yes or No)?	Y[ES] or N[O] This question allows you to double check your dialogue responses, to abort the operation if errors have been made, or proceed with the SAVE operation. Type 'YES' to proceed with the operation. Type 'NO' to abort and return to the SAV/RES Function: prompt.

The following SAVE example illustrates the backup of an RM03 disk to tape. The letters are keyed to the explanation the follows this sample run:

1. Physically mount the RM03 source disk on drive DR1:, write-locked.
2. Physically mount the destination tape on drive MM1:, write-enabled.
3. Type RUN $SAVRES and then press the RETURN key.

```
a  $ RUN $SAVRES

b  SAV/RES Function: SA

c  From RSTS disk? DR1:

   *** Pack ID/default SAVE Set Name is JOEM

d  To Device? MM1:SAVSET

   *** SAVE Set Name is SAVSET

e  Expiration Date <1-Jul-82>? 1-Jan-82

f  Verify (Yes or No) <NO>?RET

g  Proceed (Yes or No)? Y

   *** Initializing first SAVE volume

   *** Begin SAVE from DR1: to MM1: at 08:07 PM

h      Dismount Device:   MM1:
              Set Name:   SAVSET
                 Seq #:   1
               Density:   800
         Creation date:   Wednesday, 1-Jul-81
       Expiration date:   Friday, 1-Jan-82

          Please label this volume!

   Mount volume # 2 of SAVE Set SAVSET
i  Device ? MM2

j  Proceed (Yes or No)? Y

   *** Begin SAVE from DR1: to MM2: at 08:14 PM

      Dismount Device:   MM2:
              Set Name:   SAVSET
                 Seq #:   2
               Density:   800
         Creation date:   Wednesday, 1-Jul-81
       Expiration date:   Friday, 1-Jan-82

          Please label this volume!

   --- SAVE operation completed at 08:22 PM
```

(continued on next page)

```
k                Summary Report

    SAVE of DR1:JOEM to SAVE Set SAVSET

          Input Device:   DR1:
               Pack ID:   JOEM
     Pack Clustersize:   8
        Creation date:   Friday, 16-Jan-81
        Output Device:   Mastape
             Set Name:   SAVSET
         # of volumes:   2
              Density:   800
        Creation date:   Wednesday, 1-Jul-81
      Expiration date:   Friday, 1-Jan-82

    SAVE started on Tuesday, 1-Jul-81, at 08:07 PM

l               Run Statistics

    Transfer Totals:

         Total of 55192 blocks transferred

    Error Totals:

         Total of     0 new bad blocks encountered on source.

Timing Totals:

       Total elapsed time:   0 hrs.,  14 mins.,  41 secs.

       Total wait    time:   0 hrs.,   0 mins.,  19 secs.

       Total process time:   0 hrs.,  14 mins.,  22 secs.

m   SAV/RES Function: ^Z

    $
```

a Type RUN $SAVRES, and then press the RETURN key to run the SAVE/RESTORE program.

b After you press RETURN, SAVE/RESTORE prints its initial program prompt. Because you need to perform a backup operation of an RM03, you select the SAVE function of SAVE/RESTORE by typing SA and pressing the RETURN key.

c SAVE/RESTORE asks for the device name and unit number of the disk from which you are extracting data. You respond with DR, the device mnemonic of the RM03 disk, and indicate that it is mounted on drive 1. As an option, you could have included a Pack ID with the device specification. If you had, SAVE/RESTORE would have compared your Pack ID response to the ID currently on the disk. By not specifying a Pack ID, SAVE/RESTORE accepts the current ID. After you terminate your response to this question, the program shows you the name of the Save Set.

d You specify the medium to which SAVE copies the input data. Because you want the SAVE Set to reside on tape, you type MM1:. Rather than allow SAVE to give the SAVE Set the same name as the input disk, you give the SAVE Set its own. If you assign the input and output media separate names, you can better differentiate their identity (with IDENTIFY) at a future date. Once you end your response, SAVE stores the name you specified on the output medium and then prints the SAVE SET NAME IS SAVSET message.

e SAVE creates a default expiration date equal to the current date plus one year. This represents the date after which you can write over the data. Rather than accept the default by pressing LINE FEED, you specify your own date and press the RETURN key.

f You choose not to have SAVE/RESTORE verify the transfer. A YES or Y response causes SAVE to perform a verify pass after it completes the data transfer process. In the verify pass, SAVE compares the input data with the data it transfers to the output medium. As an alternative, if you know that you need to verify the transferred data, you can attach the /VERIFY switch to the SAVE/RESTORE function and thus suppress the VERIFY (YES or NO) question.

g You have made no errors in the dialogue sequence; therefore, you can type Y to have SAVE proceed with the operation. Because no default exists for this question, you must type either a YES or a NO response. Once you terminate your answer, SAVE notifies you that it is initializing the first SAVE volume. Only the first volume of the SAVE Set is initialized. When that operation is complete, SAVE prints a message indicating that the transfer process has begun, which devices are involved, and the time the operation began.

h The dismount report summarizes the label information found on the initial SAVE Set volume. If SAVE/RESTORE requires another volume, it asks you to mount the next volume in the SAVE Set after the first volume is exhausted.

i You respond to the DEVICE question with the device name and unit number of the second SAVE Set volume, and then press the RETURN key.

j Again, you have the option to proceed or to abort the operation. Because there is no reason to abort, you type Y and the RETURN key to proceed. SAVE then notifies you, as it did for the first volume, that the SAVE operation has begun, that the data is being transferred, and that 08:14 was the time the SAVE operation began. As soon as SAVE finishes the transfer, it issues the dismount message for the second volume of the SAVE Set, informs you that the SAVE operation is complete, and the time it finished.

k The Summary Report appears by default and can only be suppressed by attaching the /NOSTATS switch (or by making /NOSTATS the default) to a SAVE/RESTORE function. The report includes the Pack ID and SAVE Set names, label information for both the input and output volumes, and the date of the SAVE operation.

l The Run Statistics report prints the number of blocks transferred, the number of bad blocks encountered, and elapsed times. Note that the blocks transferred includes an extra set of directory blocks stored at the end of the SAVE Set. The additional set of directory blocks are included to ensure that SAVE/RESTORE has a valid directory to read when it attempts to restore from the SAVE Set.

m When the SAVE operation is complete, SAVE/RESTORE returns to its initial SAV/RES FUNCTION prompt. If you want to return to your keyboard monitor prompt (DCL in this example), type CTRL/Z.

9.5.6 Restoring a RSTS/E Disk using the RESTORE Dialogue

To recreate a RSTS/E file-structured disk from a SAVE Set, you must perform the RESTORE operation. Because SAVE/RESTORE writes the SAVE Set in the SAVE format, no other RSTS/E software can perform this rebuild process. Before transferring any data, SAVE/RESTORE tries to extract bad block information from the destination disk and, if it finds any bad blocks, incorporates them into the new BADB.SYS file.

The SAVE/RESTORE program may encounter bad blocks during a RESTORE run. If it does, it scans the SATT.SYS file to determine if there is a place where the block corresponding to a bad block can be relocated. If a large enough area exists on the destination disk, the program simply moves the data to the new location and modifies the directory information accordingly.

The relocation of blocks may require RESTORE to change a file's characteristics. This can occur, for example, if the program encounters a bad cluster while transferring a contiguous file. To relocate the cluster that falls on the bad block, SAVE/RESTORE must make the file noncontiguous. Changes of this type cause RESTORE to preserve the data and to inform you of these adjustments.

The operation ends if there is no way RESTORE can relocate the data cluster. This can occur if:

- No unallocated pack cluster remains on the disk
- The file cluster size is larger than the pack cluster size and there is not an equivalent number of contiguous pack clusters left on the disk

Normally, SAVE/RESTORE uses the name of the input SAVE Set as the Pack ID of the output volume. After you answer the FROM DEVICE question, SAVE/RESTORE prints a message showing you the name it plans to use. The general format is:

*** SAVE Set name/Default Pack ID is xxxxxx

After you successfully respond to the TO RSTS DISK question, SAVE/RESTORE prints the message:

*** Pack ID is xxxxxx

Table 9–6 lists the RESTORE dialogue questions. An example of a RESTORE procedure follows the dialogue description.

Table 9–6: RESTORE Dialogue Questions

Question	Response and Description
From Device?	<devspec> Data is transferred from the specified device. If you include a SAVE Set Name in the device specification (see Section 9.5.2), SAVE/RESTORE checks whether it matches the one on the device. If it does not, the program prints a warning message (Section 9.5.3) and prints the FROM DEVICE question again. The device must be physically mounted and should be write-locked. Type 2 characters to specify the input device type, followed by a single digit (0–7) to specify the unit number. Include a SAVE Set Name if you want SAVRES to ensure it matches the one on the volume. This drive should be write-locked.
To RSTS Dx: Disk?	<devspec> Data from the previously specified SAVE Set is restored to this disk. If you include a Pack ID in the device specification, it becomes the Pack ID of the restored disk. Otherwise, SAVE/RESTORE uses the SAVE Set Name (which may also be the Pack ID of the original source disk). Physically mount and write-enable the disk. Type 2 characters to specify the output device type, followed by a single digit (0–7) to specify the unit number. Include a Pack ID if you wish to override the default. This drive should have a SCRATCH volume mounted, and must be write-enabled.
Verify (Yes or No) <NO>?	Y[ES] or N[O] This is the dialogue form of the /VERIFY switch. If you answer YES, SAVE/RESTORE compares the output with the input at the end of each input volume to ensure the volumes are equivalent. Type 'Yes' if you want SAVRES to compare the input and output volumes at the end of each input volume. Type 'No' or press the LINE FEED key if you do not want SAVRES to perform this verification.
Proceed (Yes or No)?	Y[ES] or N[O] This question allows you to double check your dialogue responses, to abort the operation if errors have been made, or to proceed with the RESTORE operation. Type 'Yes' to proceed with the operation. Type 'No' to abort and return to the SAV/RES Function: prompt.

NOTE

If you are using the output of a RESTORE operation as a system disk, you must use the INIT.SYS INSTALL option to reinstall the desired monitor Save Image Library (SIL). You must also redo any necessary HARDWR suboptions (such as changing Hertz) because the INSTALL option clears various items set up by the HARDWR suboptions. This procedure is necessary because a disk can be saved with SAVE from one system disk and restored with RESTORE on a system with a different hardware configuration.

The following example illustates the restoration of an RM03 disk from the tape SAVE Set created in the previous example. The letters are keyed to the explanation that follows this sample run:

1. Mount the two SAVE Set volumes on drive MM1: and MM2:, write-locked.
2. Mount the destination disk on DR1:, write-enabled.
3. Run SAVE/RESTORE by typing RUN $SAVRES.

```
a  $ RUN $SAVRES

b  SAV/RES Function: RE

c  From device? MM1:

   *** SAVE Set Name/default Pack ID is SAVSET

d  To RSTS DR: Disk? DR1:TSTPAK

   *** The volume on DR1: is a RSTS disk
   *** Pack ID is MYPAK

   *** Pack will be reinitialized
   Mount it anyway <NO>? Y

   *** Pack ID is TSTPAK

e  Verify (Yes or No) <NO>? (RET)

f  Proceed (Yes or No)? Y

   *** Begin RESTORE from MM1: to DR1: at 08:28 PM

      Dismount Device:   MM1:
             Set Name:   SAVSET
               Seq #:    1
             Density:     800
        Creation date:   Wednesday, 1-Jul-81
      Expiration date:   Friday, 1-Jan-82

   Mount volume # 2 of SAVE Set SAVSET
   Device ? MM2:
```

(continued on next page)

```
Proceed (Yes or No)? Y

*** Begin RESTORE from MM2: to DR1: at 08:32 PM

   Dismount Device:    MM2:
          Set Name:    SAVSET
             Seq #:    2
           Density:    800
     Creation date:    Wednesday, 1-Jul-81
   Expiration date:    Friday, 1-Jan-82

   Dismount Device:    DR1:
           Pack ID:    TSTPAK
  Pack Clustersize:    8
     Creation date:    Friday, 15-Jan-81

        Please label this volume!

--- RESTORE operation completed at 08:37 PM

g              Summary Report

RESTORE of SAVE Set SAVSET to DR1:TSTPAK

       Input Device:    Mastape
          Set Name:    SAVSET
       # of volumes:    2
           Density:    800
     Creation date:    Wednesday, 1-Jul-81
   Expiration date:    Friday, 1-Jan-82

       Output Device:    DR1:
           Pack ID:    TSTPAK
  Pack Clustersize:    8
     Creation date:    Friday, 15-Jan-81

RESTORE started on Tuesday, 1-Jul-81, at 08:28 PM

h              Run Statistics

Transfer Totals:

     Total of 54184 blocks transferred

Error Totals:

     Total of     0 new bad blocks encountered on destination.

     0 files structurally altered.

Timing Totals:

      Total elapsed time:    0 hrs.,    9 mins.,   36 secs.

      Total wait    time:    0 hrs.,    0 mins.,   16 secs.

      Total process time:    0 hrs.,    9 mins.,   20 secs.

i  SAV/RES Function: ^Z

   $
```

a You type RUN $SAVRES, and then press the RETURN key to run the SAVE/RESTORE program.

b You begin the RESTORE operation by typing RE and the RETURN key in response to the initial SAVE/RESTORE prompt. Because you need to recreate an RM03 from a tape SAVE Set, you select the RESTORE function.

c The RESTORE dialogue begins when it prints the FROM DEVICE question. In response, you specify the device name and the unit number of the SAVE Set medium and press the RETURN key. Because the input medium is tape, mounted on drive number 1, you enter MM1:. If you had included a SAVE Set Name in the device specification, RESTORE would have checked it against the name already on the set. If they had not matched, RESTORE would have issued a warning message informing you of the discrepancy. After you terminate your response with the RETURN key, RESTORE reads the SAVE Set Name from the input medium and displays it.

d Because in this RESTORE operation you are recreating an RM03 disk, you respond to the TO RSTS DR: DISK question by typing DR1:TSTPAK. By your response, you indicate that the device mnemonic of the RM03 is DR, that it is mounted on drive number 1, and that you wish it to have a Pack ID of TSTPAK. After pressing the RETURN key, RESTORE checks the Pack ID of the output disk. It then prints a message showing you that the disk is a RSTS/E disk, has a Pack ID of MYPAK, and that RESTORE will reinitialize it. RESTORE issues a mount question that gives you the option to mount the disk or not. Because the data on the disk is no longer valuable, answer Y and press the RETURN key. RESTORE then initializes the disk with the new Pack ID.

e The comparison of the input data to the data already transferred to the destination disk is not essential during this particular run; therefore, you press the LINE FEED key to accept the NO default. When you ask RESTORE to verify the transferred data, you are essentially requesting that RESTORE perform two operations rather than one. That is, you make the program perform two passes on the data, once for the transfer and once for the comparison.

f No mistakes were made during the dialogue phase; therefore, RESTORE proceeds with the operation in reply to your Y response. After you press the RETURN key, RESTORE displays a message to indicate it has begun the transfer. After RESTORE exhausts the first volume, it prints input medium information, asks you to mount the next volume, and asks if you want to continue the operation. Again, because there are no errors in the dialogue, RESTORE proceeds. RESTORE moves the remaining data to the second volume, issues the input and output dismount messages, and tells you the operation is completed.

g The Summary Report prints the SAVE Set Name, the Pack ID, and label information for both the input and output medium, in addition to the date and time of the RESTORE operation.

h The Run Statistics Report prints the number of blocks transferred, the number of bad blocks encountered, and the times required to perform various operations. Note that the number of blocks transferred does not include the extra set of directory blocks stored at the end of the SAVE Set.

i RESTORE returns to the SAV/RES FUNCTION prompt when the operation has ended. Pressing CTRL/Z returns you to your default keyboard monitor (DCL in this example).

9.5.7 Copying a RSTS/E Disk using the IMAGE Dialogue

The IMAGE function allows you to make an equivalent copy of a RSTS/E file-structured disk; however, only disk to like disk transfers are legal when using the IMAGE operation. Like disks are units that have exactly the same device size. The SAVE/RESTORE program handles bad block recovery during an IMAGE operation in the same way as RESTORE.

If you choose the IMAGE function, SAVE/RESTORE asks the dialogue questions in Table 9–7. An example of the IMAGE operation follows the Table 9–7 description.

NOTE

If you are using the output of an IMAGE operation as a system disk, you must use the INIT.SYS INSTALL option to reinstall the desired monitor Save Image Library (SIL). You must also redo any necessary HARDWR suboptions (such as changing Hertz) because the INSTALL option clears various items set up by the HARDWR suboptions. This procedure is necessary because you can make a copy of a disk on one system and use the copied disk on another.

Table 9–7: IMAGE Dialogue Questions

Question	Response and Description
From RSTS Disk?	<devspec> SAVE/RESTORE transfers all data from the disk specified in the device specification response. If you include a Pack ID in the device specification, SAVE/RESTORE checks to see if it matches the one actually on the disk. If it does not, the program prints a warning message (Section 9.5.3) and prints the FROM RSTS DISK question again. The device must be physically mounted and for safety reasons should be write-locked. Type 2 characters to specify the input device type, followed by a single digit (0–7) to specify the unit number. Include a Pack ID if you want SAVRES to ensure it matches the one on the volume. This drive should be write-locked.

(continued on next page)

Table 9–7: IMAGE Dialogue Questions (Cont.)

Question	Response and Description
To RSTS Dx: Disk?	<devspec> SAVE/RESTORE transfers all information from the previously specified source disk to this disk. If you include a Pack ID in the device specification, it becomes the Pack ID of the output disk. Otherwise, SAVE/RESTORE uses the Pack ID of the source disk. The disk must be physically but not logically mounted and write-enabled. Type 2 characters to specify the output device type, followed by a single digit (0–7) to specify the unit number. Include a Pack ID if you wish to override the default. This drive should have a SCRATCH volume mounted, and must be write-enabled. Note that the device mnemonic you specify in response to the FROM RSTS DISK question appears in place of the Dx in the TO RSTS Dx: DISK? question. In some cases, you may specify, in your device specification response to the TO RSTS Dx: DISK question, a device mnemonic other than the one that appeared for Dx. See the discussion of like disks in Section 9.2.
Verify (Yes or No) <NO>?	Y[es] or N[o] This is the dialogue form of the VER[IFY] switch. If you answer with YES, SAVE/RESTORE compares the output with the input at the end of each output volume to ensure that the volumes are equivalent. If answer with NO, SAVE/RESTORE moves on to the next question without performing a verify pass. Type 'Yes' if you want SAV/RES to compare the input and output volumes at the end of the IMAGE copy operation. Type 'No' or press the LINE FEED key if you do not want SAVE/RESTORE to perform the verification.
Proceed (Yes or No)?	Y[ES] or N[O] This question allows you either to double check your dialogue responses and abort the operation if you have made any errors or to proceed with the IMAGE operation. Type 'Yes' to proceed with the operation. Type 'No' to abort and return to the SAV/RES Function: prompt.

If you do not intervene, SAVE/RESTORE uses the input Pack ID as the Pack ID of the output disk. After you answer the FROM RSTS DISK question, the program prints a message in the following general format to tell you the Pack ID it intends to use:

*** Input Pack ID/default Output Pack ID is XXXXXX

If you do not want the output disk to have the same name as the input volume, specify a different Pack ID in the <devspec> response to the TO RSTS DX: DISK question. After you select the Pack ID and answer the question, SAVE/RESTORE prints:

*** Output Pack ID is XXXXXX

The following procedure describes an IMAGE operation that copies an RP04 to an RP04 and verifies the transfer in the same pass. The letters are keyed to the explanation that follows this sample run:

1. Mount the source disk on drive DB0:, write-locked.
2. Mount the destination disk on drive DB1:, write-enabled.
3. Type RUN $SAVRES, and then press the RETURN key.

```
a  $ RUN $SAVRES

b  SAV/RES Function: IMAGE/NOERROR

c  From RSTS disk? DB0:

   *** Input Pack ID/default Output Pack ID is SOURCE

d  To RSTS DB: Disk? DB1:DESTIN

   *** The volume on DB1: is a RSTS disk
   *** Pack ID is OUTPUT

   *** Pack will be reinitialized
   Mount it anyway <NO>? YES

   *** Output Pack ID is DESTIN

e  Verify (Yes or No) <NO>? YES

f  Proceed (Yes or No)? YES

   *** Begin IMAGE copy from DB0: to DB1: at 11:02 AM

   *** Begin VERIFY pass from DB0: to DB1: at 11:04 AM

   *** 0 differences found

      Dismount Device:   DB1:
               Pack ID:  DESTIN
     Pack Clustersize:   8
        Creation date:   Wednesday, 1-Jul-81

          Please label this volume!

   --- IMAGE copy operation completed at 11:07 AM

g                Summary Report

   IMAGE copy of DB0:SOURCE to DB1:DESTIN

         Input Device:   DB0:
              Pack ID:   SOURCE
     Pack Clustersize:   8
        Creation date:   Wednesday, 1-Jul-81

        Output Device:   DB1:
              Pack ID:   DESTIN
     Pack Clustersize:   8
        Creation date:   Wednesday, 1-Jul-81
```

(continued on next page)

```
IMAGE copy started on Wednesday, 1-Jul-81, at 11:02 AM

h              Run Statistics

   Transfer Totals:

        Total of 24400 blocks transferred

   Error Totals:

        Total of      0 bad compares.

        Total of      0 new bad blocks encountered on source.

        Total of      0 new bad blocks encountered on destination.

        0 files structurally altered.

   Timing Totals:

        Total elapsed time:   0 hrs.,   5 mins.,  22 secs.

        Total wait    time:   0 hrs.,   0 mins.,   0 secs.

        Total process time:   0 hrs.,   5 mins.,  22 secs.

i  SAV/RES Function: ^Z

   $
```

a When you are under the control of your keyboard monitor, you type RUN $SAVRES and then press the RETURN key to run the SAVE/RESTORE program.

b SAVE/RESTORE prints the SAV/RES FUNCTION prompt to determine which function you wish to perform. Because you need to create a copy of an RP04, you select the IMAGE function by typing IM/NOERROR and by pressing the RETURN key. You attach the /NOERROR switch because the data on the disk is valuable. The /NOERROR switch causes SAVE/RESTORE to end the operation whenever the program encounters a run-time error. Before the program aborts, it prints an error message indicating the source of the problem.

c In reply to the FROM RSTS DISK question, you specify the disk from which you are to extract the data. You enter the device name and unit number of the input disk and end the response with the RETURN key. After this response, IMAGE reads the Pack ID from the input disk and displays it on your terminal.

d IMAGE requests the device name and unit number of the output disk. You type DB1:DESTIN to indicate that the RP04 is on drive 1 and you want DESTIN to be the new Pack ID. As soon as you press the RETURN key, SAVE/RESTORE tells you that the output medium is a RSTS/E disk, its Pack ID is OUTPUT, and it will be initialized. SAVE/RESTORE initializes the disk when you type Y and press the RETURN key in response to the mount question. A message that SAVE/RESTORE prints informs you that the disk was initialized to the specified Pack ID.

e Because you are copying valuable data, you type YES to have IMAGE determine if the input data equals the data that was transferred to the output disk. If IMAGE finds that the data are not equivalent, it informs you of these differences.

f You are satisfied that you made no errors in the dialogue sequence and feel that the operation should proceed. After you press the RETURN key, IMAGE informs you when it begins to transfer data and when it begins the verify pass. After completing the data comparison, IMAGE prints the number of differences found between the two devices. It then issues the output device dismount message. Immediately after IMAGE prints the dismount request, it prints a message to show you the copy operation successfully ended.

g IMAGE prints the Summary Report unless you suppress it by using the /NOSTATS switch. The report includes the Pack ID and label information of each disk, in addition to the date and time that the IMAGE operation was run.

h The Run Statistics Report provides a total figure for the number of blocks transferred, the errors encountered, and the times required to perform the IMAGE operation.

i SAVE/RESTORE returns you to the SAV/RES FUNCTION prompt when the operation ends. You press CTRL/Z to return to your keyboard monitor (DCL in this example).

9.5.8 IDENTIFY Dialogue

If you want label information from a SAVE Set volume or a RSTS/E file-structured disk, use the IDENTIFY function. SAVE/RESTORE asks only one question before the IDENTIFY operation begins. Table 9–8 describes this dialogue question. An example of an IDENTIFY operation appears after the table.

Table 9–8: IDENTIFY Dialogue Question

Question	Response and Description
From Device?	<devspec> SAVE/RESTORE prints the device label characteristics of the SAVE Set or RSTS/E file-structured disk for the device specified in the dialogue response. If you include a SAVE Set Name or Pack ID, SAVE/RESTORE checks to see if it matches the one actually on the device. If it does not, the program prints a warning message (see Section 9.5.3) and then prints the FROM DEVICE question again. You must mount and write-lock the device (for protection) before responding to the FROM DEVICE question. Type 2 characters to specify the input device, followed by a single digit (0–7) to specify the unit number. Include a SAVE Set Name or Pack ID if you want SAVE/RESTORE to ensure it matches the one on the volume. Mount this drive, write-locked, before answering the question.

The following IDENTIFY procedure lists the label characteristics of both a SAVE Set and a RSTS/E disk. The letters are keyed to the explanation that follows this sample run.

The IDENTIFY procedures are:

1. Mount a SAVE Set volume on MM1:, write-locked.
2. Mount the RSTS/E disk on drive DR1:, write-locked.
3. Type RUN $SAVRES, and then press the RETURN key.

```
a   $ RUN $SAVRES

b   SAV/RES Function: ID MM1:

c              Device:   MM1:
             Set Name:   SAVSET
                Seq #:   1
              Density:   1600
       Creation date:    Wednesday, 1-Jul-81
    Expiration date:     Friday, 1-Jan-82

d   SAV/RES Function: ID DR1:

              Device:   DR1:
             Pack ID:   TSTPAK
   Pack Clustersize:    8
      Creation date:    Friday, 15-Jan-81

e   SAV/RES Function: ^Z

    $
```

a To run SAVE/RESTORE you type RUN $SAVRES, and then press the RETURN key.

b The SAVE/RESTORE program prints the SAV/RES FUNCTION prompt. You respond to it with the first two letters of the IDENTIFY function and with the device name and unit number of the medium you wish to identify. Include a Pack ID if you want SAVE/RESTORE to compare the ID you specify with the one on the input volume.

c IDENTIFY prints label information for the tape SAVE Set.

d SAVE/RESTORE returns to the SAV/RES FUNCTION prompt. In response, you ask SAVE/RESTORE to identify the disk pack mounted on DR1:. SAVE/RESTORE prints the information as soon as you end your response.

e You press CTRL/Z to return to your keyboard monitor (DCL in this example).

9.5.9 Full Function Command Line

Use the full function command line format if you do not want to answer each dialogue question separately. The general format is:

SAV/RES Function:<function><outdevspec> = <indevspec>[<switch(es)>]

Function can be SA[VE], RE[STORE], IM[AGE], or ID[ENTIFY] and the switches can be any of those listed as valid with the associated function as described in Section 9.4. The device specifications are presented in Section 9.5.2.

If you specify a full command in response to the SAV/RES FUNCTION prompt, SAVE/RESTORE prints only the PROCEED question (except during an IDENTIFY question):

```
Proceed (Yes or No)?
```

You must type either YES or NO to this question. Type YES to have SAVE/RESTORE begin the operation or type NO to abort.

SAVE/RESTORE does the same standard volume label checking during the full function command line operation as it does during the longer dialogue procedure. If a discrepancy occurs, SAVE/RESTORE prints the error message and then returns to the SAV/RES FUNCTION prompt. At that point, you must type the full function command line again.

9.6 SAVE/RESTORE and Booting

SAVE/RESTORE always attempts to create bootable media. The output of a RESTORE or IMAGE copy operation is bootable if the original disk contained an INIT.SYS file in [0,1] (even if the original disk was not bootable). The first output volume that the SAVE operation produces is always bootable because SAVE/RESTORE copies the INIT.SYS file from the system disk and hooks it on this volume. Note that only the first volume in a SAVE Set is bootable.

If a SAVE volume is booted, a portion of INIT.SYS is loaded, allowing you to perform the following INIT options:

- SAVE/RESTORE
- DSKINT
- BOOT
- HARDWR option specifying the LIST suboption

The types of SAVE/RESTORE operations you can perform depend on the type of device from which you boot. You can run all four SAVE/RESTORE options if you boot from a RSTS/E disk. However, you can perform only an IDENTIFY, IMAGE, or RESTORE if you boot from a RSTS/E magnetic tape or SAVE volume. Note that if you boot a SAVE volume and subsequently perform a RESTORE operation, SAVE/RESTORE prints a message at the completion of the run:

```
*** Please boot from the system disk

Boot device?
```

You must then specify the device name and unit number of the bootable device. Table 9–9 summarizes the SAVE/RESTORE operations you can perform while booting various devices.

Table 9–9: Booting RSTS/E and SAVE Volumes

Device Booted From	SAVE/RESTORE Operations You Can Perform	SAVE/RESTORE Operations You Can Perform Afterwards
RSTS/E Magnetic Tape	IDENTIFY, IMAGE, RESTORE	IDENTIFY, IMAGE, RESTORE
SAVE Volume (Disk or Tape)	IDENTIFY, IMAGE	IDENTIFY, IMAGE, RESTORE
	RESTORE	None – You receive a message that states you must boot from the system disk.
RSTS/E Disk	SAVE, RESTORE, IMAGE, IDENTIFY	SAVE, RESTORE, IMAGE, IDENTIFY

NOTE

You cannot use the console bootstrap to boot a magnetic tape SAVE Set if (1) the tape has a density of 1600 BPI and (2) the drive the tape is mounted on has a TM02 formatter. (Use the HARDWR LIST suboption of INIT to determine the type of formatters on your drives.) You can, however, boot a 1600 BPI tape on a drive with a TM02 formatter if you use the BOOT OPTION prompt of INIT.SYS. That is, you can boot under these circumstances using software but not hardware boot procedures. As an additional restriction, you can boot a magnetic tape SAVE Set only on TS11, TU45, TU16, TU77, and TE16 tape drives.

9.7 Operator Interface During Processing

SAVE/RESTORE requires operator intervention at certain points during processing. The operator must mount and dismount devices, reset hung devices, and abort operations when necessary. The following sections describe these procedures.

9.7.1 Mounting and Dismounting Volumes

A SAVE Set can contain more than a single volume. Because you may be processing multi-volume SAVE Sets, the SAVE/RESTORE program contains procedures that allow you to mount the additional volumes in a set. When SAVE/RESTORE needs an additional volume to complete a SAVE or RESTORE operation, it prompts you with the message:

```
Mount volume #nn of SAVE Set <setnam>
Device?
        .
        .
        .
Proceed (Yes or No)?
```

The letters nn represent the volume sequence number and <setnam> indicates the SAVE Set Name and unit number of the drive on which the next volume has been readied. If you mount a device of the same type, you can respond to this prompt with only a unit number. After you press the RETURN key to terminate your device specification, SAVE/RESTORE performs volume label checking, which is described in Sections 9.5.3 and 9.5.4. At this point, if the additional volume you mount violates any SAVE/RESTORE rule, the program prints the error message immediately after the DEVICE question.

SAVE/RESTORE prints a dismount message when:

- SAVE/RESTORE exhausts a SAVE Set volume during a SAVE or RESTORE operation
- SAVE/RESTORE completes the data transfer to the output volume in an IMAGE copy operation
- You mount an incorrect volume

The dismount message generated for a SAVE Set (Format 1) differs from a dismount message printed for a RSTS/E disk (Format 2):

Format 1

```
Dismount Device:   <device>
        Set Name:  [setnam]
          Seq #:   nn
        Density:   [ 800]
                   [1600]
   Creation Date:  <(day-of-week), dd-mmm-yy>
Expiration Date:   <(day-of-week), dd-mmm-y>
```

Format 2

```
Dismount Device:   <device>
        Pack ID:   <packid>
Pack Clustersize:  mm
   Creation Date:  <(day of week), dd-mmm-yy>
```

The <device> field represents a two character device mnemonic followed by a single digit unit number. The SAVE Set Name replaces <setnam>, and the disk pack identification replaces <packid> during a valid SAVE/RESTORE operation. Density figures are printed only for magnetic tape SAVE Sets. Finally, if SAVE/RESTORE has just written to the volume, it prints the PLEASE LABEL THIS VOLUME message immediately after the volume's dismount message.

9.7.2 Re-accessing Devices

During an operation, you may find that SAVE/RESTORE does not have access to a previously accessible device. If a device does become hung or is write-protected, the program notifies you in the warning message:

```
%%% Device hung or write-locked
Retry (Yes or No)?
```

Correct the condition causing the problem if possible, and then type YES to attempt to gain access to the device. Type NO to abort the attempt.

9.7.3 Aborting SAVE/RESTORE

To end a SAVE/RESTORE operation at any time, type CTRL/C. This returns control to your keyboard monitor if you running SAVE/RESTORE on line. If you are running under the control of the INIT.SYS program and the operation ends, INIT.SYS returns you to its OPTION prompt. An abort during a SAVE, RESTORE, or IMAGE operation leaves the output volume unusable. You do not, however, jeopardize any IDENTIFY volumes if you choose to abort this operation.

9.8 SAVE/RESTORE Summary Report

SAVE/RESTORE prints a summary report at the end of SAVE, RESTORE, and IMAGE operations. The report identifies the operation that was run, lists characteristics of the input and output devices, and generates a list of data transfer, error, and timing statistics. It also documents the time and date of the operation. You receive the summary report automatically unless you suppress its output (or if you make /NOSTATS the default by applying a patch given in the *RSTS/E Release Notes*). Suppress the output by using the /NOSTATS switch with a function response (see Section 9.4).

9.8.1 Summary Report Format and Example

The format of the Summary Report varies with the operation. At times the Input Device characteristics (see Figure 9–1) identify a SAVE Set, as in the case of a RESTORE operation. In a SAVE operation, the input device characteristics represent those of a RSTS/E disk. The Output Device characteristics conform to the same pattern. Thus, depending on the operation, the SAVE Set format may be represented under either the Input or Output Device category. The RSTS/E disk format may be also printed under the Input or Output Device categories.

The general format of the SAVE/RESTORE Summary Report is illustrated in Figure 9–1.

Figure 9–1: Summary Report Format

```
SAVE
RESTORE      of  { disk<packid>      }  to  { disk<packid>      }
IMAGE            { SAVE Set <setnam> }      { SAVE Set<setname> }

      Input Device:  <device>
           Pack ID:  <packid>
  Pack Clustersize:  nn                                  *RSTS/E disk format
     Creation date:  <day of week, dd-mmm-yy>

                     [magnetic tape]
      Output Device: [disk]
           Set Name: <setnam>                            *SAVE Set format

     4 of volumes:   nn
          Density:   [ 800]
                     [1600]
     Creation date:  <day of week, dd-mmm-yy>
    Expiration date: <day of week, dd-mmm-yy>            (continued on next page)
```

Run Statistics

Transfer Totals:

Total of nnnnnnn blocks transferred

Error Totals:

Total of nnn bad compares.

Total of nnn new bad blocks encountered on source.

Total of nnn new bad blocks on destination disk.

Total of nnn files structurally altered.

Timing Totals:

Total elapsed time:	nn hrs.,	nn mins.,	nn secs.
Total wait time:	nn hrs.,	nn mins.,	nn secs.
Total process time:	nn hrs.,	nn mins.,	nn secs.

The following is an example of a Summary Report produced by a SAVE operation:

```
                Summary Report

SAVE of DM1:INPUT to Save Set MYCOPY

     Input Device:   DM1:
          Pack ID:   INPUT
 Pack Clustersize:   4
    Creation date:   Monday, 22-Jan-81

    Output Device:   Mastape
         Set Name:   MYCOPY
     # of volumes:   2
          Density:   1600
    Creation date:   Monday, 22-Jan-81
  Expiration date:   Tuesday, 22-Jan-82

SAVE started on Monday, 22-Jan-81, at 11:49 AM

            Run Statistics

Transfer Totals:

     Total of 23176 blocks transferred

Error Totals:

       Total of     0 new bad blocks encountered on source.

Timing Totals:

       Total elapsed time:   0 hrs.,   7 mins.,  17 secs.
       Total wait    time:   0 hrs.,   0 mins.,  19 secs.
       Total process time:   0 hrs.,   6 mins.,  58 secs.

SAV/RES Function: ^Z

$
```

9.8.2 Summary Report Run Statistics

The run statistics that appear at the end of SAVE, RESTORE, or IMAGE operations are listed and described in Table 9–10. Because each SAVE/RESTORE operation prints its own set of run statistics, Table 9–10 also identifies the operation(s) in which each run statistic is printed. The operations are identified according to the following conventions:

S = SAVE operation

R = RESTORE operation

I = IMAGE operation

Table 9–10: Summary Report Run Totals

Type of Statistic	Operation and Description
Transfer Statistics	
Total of blocks transferred	[SRI] This number refers to the total number of blocks transferred (directory blocks, file blocks). In a SAVE operation an extra set of directory blocks is stored at the end of a SAVE Set. These extra blocks are reflected in the figure for the total number of blocks transferred.
Error Statistics	
Total of bad compares	[SRI] The total number of errors that occurred during the VERIFY phase.
Total of new bad block errors encountered on source	[SRI] The number of previously unreported bad blocks encountered on the source disk.
Total of bad block errors on destination disk	[SRI] The number of previously unreported bad blocks encountered on the destination disk.
Total of files structurally altered	[RI] The number of files that were structually altered to enable relocation on the destination disk. This means contiguous files were made noncontiguous or placed files were moved.
Time Statistics	
Total elapsed time (TE)	[SRI] The total amount of real clock time that elapsed during the run.
Total wait time (TW)	[SRI] The amount of real clock time that elapsed while waiting for operator responses and magnetic tape rewinds.
Total process time (TP)	[SRI] The amount of real clock time that was used for processing (TE-TW = TP).

9.9 SAVE/RESTORE Error Handling

SAVE/RESTORE issues dialogue, mount, transfer, and verification error messages. When you encounter an error message during a dialogue or a mount, SAVE/RESTORE asks you to correct the condition, where appropriate, and then asks you to reenter the command. Errors you encounter during either of these phases never affect the integrity of the data being transferred. However, when SAVE/RESTORE reports an error during a transfer or verification phase, it has detected a problem that might corrupt the data being transferred. Depending on the nature of the error, a single file or perhaps an entire volume might be affected. Generally, you encounter less critical errors during a dialogue or mount phase than in a transfer or verification operation.

SAVE/RESTORE does not report an error when it finds a bad block on the output device, unless it is unable to reallocate that block in a way that is transparent to you. For instance, the one-block file SHIFT.LES is moved without notice if a free cluster can be found. On the other hand, the program issues an error message if it must move the placed file, MOVEME.NOT.

The /NOERROR switch causes SAVE/RESTORE to abort the run if it encounters any condition that warrants an error message. If you do not specify the /NOERROR switch, the program takes whatever action it deems necessary. SAVE/RESTORE always informs you of any corrective action taken so that you have the option of aborting the operation, if desired.

The most important problem that SAVE/RESTORE encounters during a RESTORE or IMAGE copy operation is finding a bad block on a RSTS/E file-structured disk when it wants to write to that block. The program attempts to correct the problem by performing the following actions in the order listed:

- Marks a contiguous file as noncontiguous and relocates the bad cluster but not the entire file.
- Marks the file as non-placed if the bad cluster occurs at the beginning of a placed file.
- Finds enough contiguous free pack clusters on the output disk to hold the number of clusters that must be moved. In other words, if a file's cluster size is 16, it finds 16 free contiguous blocks for the cluster. It proceeds if the reallocation worked and aborts if it has not worked.

SAVE/RESTORE always tells you about the changes it makes to a file that are caused by this bad block processing. If there is no way to relocate a cluster, the program aborts the operation. Unless you specify the /NOERROR, SAVE/RESTORE never terminates an operation when it encounters a bad block that results in a nonfatal error (see Section 9.4).

9.9.1 General SAVE/RESTORE Error Messages

The errors described in Table 9–11 can occur during SAVE/RESTORE operations:

Table 9–11: General SAVE/RESTORE Error Messages

Message and Meaning
%%% Bad block in first n blocks of output volume The current volume is unusable. Sufficient space must be available at the beginning of a SAVE Set to store certain critical information necessary for a SAVE/RESTORE operation. Mount a different volume and try again.
??? Bad block in MFD SAVE/RESTORE aborts the operation when it finds a bad block in the Master File Directory (MFD).
%%% Bad block in SATT.SYS There is a bad block in the input SATT.SYS file. SAVE/RESTORE does not use an input disk that has a corrupt SATT.SYS file.
??? Bad block on magtape - can't continue SAVE/RESTORE requires a magnetic tape SAVE volume to be long enough to contain a bootstrap, label information, and INIT.SYS. Whenever the program encounters a bad block before it transfers all required information, it aborts the current operation.
%%% Bad data in [MFD] [<[P,PN] FILENAME.TYPE>] [UFD [P,PN]] [GFD,nnn] A cluster contained in the described account or file could not be relocated due to a fatal read error on the output volume.
??? Bad directory for device SAVE/RESTORE has discovered a bad link or bad block in the input or output directory structure.
%%% Bad INIT.SYS on system disk SAVE/RESTORE encountered a bad block in INIT.SYS while transferring it from the system disk to a SAVE volume.
%%% Can't mix device types in SAVE Set A SAVE Set cannnot be composed of both tapes and disks. Only all disk or all tape SAVE Sets are legal.
%%% Can't read bad block file SAVE/RESTORE could not read the bad block file on a RSTS/E disk or a SAVE volume.
??? Can't read SAVE volume SATT A nonrecoverable I/O error occurred while SAVE/RESTORE attempted to read the Storage Allocation table on the SAVE volume.

(continued on next page)

Table 9–11: General SAVE/RESTORE Error Messages (Cont.)

Message and Meaning
%%% Can't write boot The bootstrap could not be written to the first block of the output disk.
%%% Can't write INIT.SYS SAVE/RESTORE encountered a fatal bad block while transferring INIT.SYS to SAVE volume.
%%% Device hung or write-locked Retry (Yes or No)? The output device is off line or is write-locked. Correct the condition and try again. Type Yes to continue or type No to abort.
%%% Device mnemonic must be specified Enter the device name along with the unit number.
%%% Disk must be DSKINTed When you are running SAVE/RESTORE on line, the output disk must be RSTS/E file-structured or a SAVE volume. Use the DSKINT option of INIT.SYS to reinitialize the disk. Refer to the *RSTS/E System Generation Manual* for a description of the DSKINT option.
%%% Duplicate switches You specified the same switch more than once.
??? Error in BADB.SYS allocation SAVE/RESTORE was unable to rebuild the bad block file (BADB.SYS) after completing the current operation.
??? Fatal output bad block A bad block was encountered at a critical position, preventing SAVE/RESTORE from completing the current operation.
%%% Illegal command Your response to a prompt was in an unacceptable format.
%%% Illegal command: No output device specied The full function line command you specified did not include an output device specification.
%%% Illegal date The date you specified was either in an unacceptable format or was included in a function other than SAVE.
%%% Illegal density The density you specified was in an unacceptable format, was included where none was acceptable, or specified an illegal density value.
%%% Illegal option When booting from a SAVE volume, you can use only the RESTORE, IMAGE, and IDENTIFY options.

(continued on next page)

Table 9–11: General SAVE/RESTORE Error Messages (Cont.)

Message and Meaning
%%% Illegal switch You used a switch in an unacceptable manner. %%% Illegal switch combination You specified conflicting switches. %%% Input disk has only nn% free clusters, Mount it anyway <No>? SAVE/RESTORE discovers the input disk has very few clusters that are not allocated and may encounter problems copying or restoring the disk. You may want to delete some unnecessary file to increase the chances that SAVE/RESTORE can eventually complete a successful RESTORE. %%% Input disk should be Rebuilt The disk you mounted is "dirty." The SATT.SYS file may be corrupt. SAVE/RESTORE requires a valid SATT.SYS; therefore, rebuild the disk using the REFRESH option of INIT or the ONLCLN program. See the *RSTS/E System Generation Manual* for a description of REFRESH and Section 7.6 of this manual for information about ONLCLN. %%% Its expiration date has not passed Mount it anyway (Yes or No)? A SAVE Set on an output volume contains an expiration date that has not expired. SAVE/RESTORE prints the SAVE Set Name and expiration date before issuing this message. Confirm whether you should use this volume or mount a different volume if appropriate. %%% Magtape select error Retry (Yes or No)? When access to a magnetic tape drive was attemped, the selected unit was off line. %%% No default-type <cr> for more information You typed a LINE FEED in response to a question that has no default. ??? No INIT.SYS on system disk There is no INIT.SYS in account [0,1] on the system disk. %%% No previous question - type <cr> for more information You attempted to return to the previous question but it does not exist. %%% Non-existent or hung device You specified a disk or magnetic tape that does not exist on your system or is not on line. Check your configuration or the device type again and retry. ??? Non-recoverable I/O error SAVE/RESTORE encountered a fatal error condition while trying to recover from a DEVICE HUNG OR WRITE LOCK error. The current operation is aborted and you are returned to the SAV/RES FUNCTION prompt.

(continued on next page)

Table 9–11: General SAVE/RESTORE Error Messages (Cont.)

Message and Meaning
%%% Not a valid device You specified an unacceptable device type in response to the question. This may occur if you specify. 1. A device other than magnetic tape or disk 2. A tape rather than a disk (or vice versa) 3. A wrong disk type on an IMAGE 4. A device without including a unit number, for example, DK: instead of DK0:
%%% Only n% of the output disk clusters are available for relocation Mount it anyway <No>? SAVE/RESTORE finds there are not many free blocks available for relocation on the output disk. DIGITAL recommends you use another disk for output when you can; the chances that the operation will not succeed are high enough to justify replacing the output disk.
??? Operation aborted at user request The user specified the /NOERROR switch and one of the errors described in Table 9–12 occurred, or the user typed No to a RETRY (Yes or No) prompt.
%% Same device specified You designated the same device for both input and output. Specify the correct output device.
%%% Tape won't respond to density XXXX SAVE/RESTORE attempted to set the density of the specified magnetic tape to XXXX but the hardware rejected the request. Specify a density setting that is legal for that device, or use a different device.
%%% Tape won't respond to any density The input volume cannot be read at any legal density setting.
%%% This is not the correct volume The SAVE Set or Pack ID you specified does not match the one on the mounted device. A DISMOUNT DEVICE message follows the error message and identifies the volume if it is a SAVE Set or a RSTS/E file-structured disk. This message also occurs if SAVE/RESTORE requests the next volume of a Save Set and during a: 1. SAVE you specify a device containing a volume of the current Save Set 2. RESTORE you specify a device containing a volume that is not part of the current Save Set or is not the correct sequential volume of the current Save Set Mount the correct volume.
%%% This volume has no label SAVE/RESTORE did not find a valid SAVE Set label or Pack ID on the specified input volume.
%%% Too many bad blocks on input disk The number of bad clusters on the input disk exceeds the RSTS/E limit.

(continued on next page)

Table 9–11: General SAVE/RESTORE Error Messages (Cont.)

Message and Meaning
%%% Too many bad blocks on output disk The number of allocated clusters on the original source disk exceeds the number of available clusters on the output disk or the number of bad clusters on the output disk exceeds the RSTS/E limit (161 bad clusters). Use a different disk pack.
%%% Unrecognized switch You have specified an invalid switch.
??? Volume is full There was not enough room for the relocation of data clusters due to the blocks encountered on the output volume.
%%% Wrong size disk The output volume of an IMAGE or RESTORE must be the same size as the original source. (See discussion of LIKE disks in Section 9.2)
%%% You will be writing to the booted device If you write to this device, you need to reboot when you are finished with SAVE/RESTORE.

9.9.2 Transfer Errors-Fatal and Nonfatal

Transfer errors you may encounter in a SAVE/RESTORE operation can be either fatal or nonfatal. Fatal transfer errors cause an immediate termination of the current operation while nonfatal errors, if the /NOERROR switch is not in effect, allow the operation to continue despite the problem. On the other hand, if you specify the /NOERROR switch and one of the errors in Table 9–12 does occur, SAVE/RESTORE terminates the operation immediately (which indicates the output is not a valid medium). In other words, the nonfatal errors listed and described in Table 9–12 become fatal if the /NOERROR is attached to the function response.

Whether or not you specify /NOERROR, the error message ???VOLUME IS FULL occurs if SAVE/RESTORE cannot relocate a file due to a bad block on the output device. The relocation does not take place because SAVE/RESTORE cannot find any additional clusters available or does not find as many free clusters as it needed. This results from too many bad blocks on the output volume.

Table 9–12 contains a description of all nonfatal transfer errors (if you do not specify /NOERROR).

Table 9–12: SAVE/RESTORE Nonfatal Transfer Errors

Message and Meaning
`%%% Bad block on input, PCN=nnnnnn` A bad block was encountered on an allocated cluster on the input volume. A copied version of the data in that block might be corrupt.
`%%% Bad block on input, output PCN's affected:` A bad block was encountered on the input SAVE volume, which will affect certain pack clusters on the output disk. A copied version of the data in that block might be corrupt.

`%%% Bad block on input in MFD [PCN=nnnnnn]`
`in UFD [P,PN] [, PCN=nnnnnn]`
`in GFD, nnn [, PCN=nnnnnn]`
`in file [P,PN]FILENAME.TYPE,[PCN=nnnnnn]`
A bad block was encountered on an allocated cluster on the input volume. A copied version of the data in that block might be corrupt.

`%%% Bad Compare, PCN=nnnnnn` The data on the input volume did not compare with the corresponding data on the output volume. (See the explanation following this table.)

`%%%Bad compare on input in MFD[PCN=nnnnnn]`
`in UFD [P,PN] [, PCN=nnnnnn]`
`in GFD, nnn [, PCN=nnnnnn]`
`in file [P,PN]FILENAME.TYPE,[PCN=nnnnnn]`
The data on the input volume did not compare with the corresponding data on the output volume. (See the explanation following this table.)

`%%% File XXXXXX has been changed to non-contiguous` SAVE/RESTORE found a bad block in file XXXXXX. In order to relocate the block, the program made the file noncontiguous. Consider the consequences of this change and proceed accordingly.
`%%% Placed file XXXXXX has been moved from aaaaaa to bbbbbb` A bad block was encountered in file xxxxxx. In order to relocate it, the first block of the file was moved. Consider the consequences of this change and proceed accordingly.

During a verification pass, SAVE/RESTORE may detect differences between the input and output. If there are differences, SAVE/RESTORE informs you with two types of messages:

1. %%%Bad compare, PCN = nnnnn

2. %%%Bad compare in [MFD [, PCN = nnnnn]]
   [UFD [p,pn] [, PCN = nnnnn]]
   [GFD, nnn [, PCN = nnnnnn]]
   [file [p,pn]filenam.typ [, PCN = nnnnnn]]

SAVE/RESTORE first prints a general error message and then, after some additional processing, prints the second error message telling you more specifically the accounts and/or files in which the bad comparisons occurred.

Specifically, if you request transfer verification, SAVE/RESTORE checks whether the data it has written matches the corresponding data it read. (It does this after the intial transfer phase.) If there is a difference, SAVE/RESTORE prints:

%%%Bad compare in [MFD [, PCN = nnnnn]]

The SAVRES program transfers only clusters that are marked as allocated in the Storage Allocation Table. During the actual transfer and verify pass, it can tell only which Pack Cluster Number (PCN) it has completed. If /NOERROR is in effect and you are performing a SAVE or IMAGE copy, SAVE/RESTORE then tries to pinpoint the problem by scanning the input directory structure before aborting. If you are doing a RESTORE, SAVE/RESTORE simply aborts. This difference exists because SAVE/RESTORE must have a complete directory to scan in order to find accounts and/or files that caused the problems. In the case of a RESTORE, the output volume is not yet complete and SAVE/RESTORE cannot scan it.

If /NOERROR is not in effect, SAVE/RESTORE stores each PCN that caused a problem in an internal table as it continues with the verify pass. If SAVE/RESTORE encounters more entries than can be stored in the table, it prints the message:

```
"Affected file will not be reported"
```

This means there is no more room to store the information. At the end of the verification pass, SAVE/RESTORE prints the total number of variations found in the form:

```
%%% nn differences found
```

After completing the transfer, SAVE/RESTORE scans the input (SAVE or IMAGE) or (RESTORE) volume directories to find the accounts and/or files in which the bad comparisons occurred. One or more of the following messages is displayed:

%%%Bad compare in [MFD [, PCN = nnnnn]]
[UFD [p,pn] [, PCN = nnnnn]]
[GFD, nnn [, PCN = nnnnnn]]
[file [p,pn]filenam.typ [, PCN = nnnnnn]]

Chapter 10
Updating RSTS/E Software

Software maintenance is an ongoing process. DIGITAL is continually enhancing its software and fixing known software problems. When these changes to the software take place between releases, the changed code is made available by *updates*. The concepts presented here apply to updating RSTS/E and all its optional layered products.

This chapter focuses on the automated process of updating software. *Automated updates* are changes to code that you apply from update files, rather than having to type them in. By contrast, *manual updating* is the process of typing in changes to code. Few users should have to update software manually. The guidelines for choosing the manual update process are explained in this chapter; the actual details of manual updates are contained in Appendix C.

10.1 Contents of This Chapter

This chapter has two parts:

Part I Background Information

This part gives you an overview of updating RSTS/E software. It explains what update kits are and what is involved in the update process. Part I also describes reference documents that contain additional information on updating software.

Part II Programs for Updating RSTS/E Software

This part lists the programs available for updating software and explains when to use each one. Individual sections then describe each program in more detail and show examples of their use.

If you are inexperienced in updating RSTS/E software, you will probably find both parts helpful. System managers who are familiar with the update process may want to skim Part I or turn directly to Part II.

10.2 Kinds of Update Files

Updates can apply to either RSTS/E itself or to its layered products. There are two kinds of update files, or "patches":

- Mandatory patches, which you must install to keep your system supported. Mandatory patches fix known errors.
- Feature patches, which allow you to tailor your system to your liking. All feature patches are optional.

10.2.1 Mandatory Patches

A mandatory patch corrects software that might otherwise malfunction. For example, DIGITAL could issue an update to the monitor if one of the SYS calls returned an incorrect value. If you did not install the patch, your system users could run into problems when trying to use the SYS call.

Some mandatory patches do not apply to all systems. For example, if your system monitor is not configured to include FMS–11, it would not make sense for you to install mandatory FMS–11 patches to that monitor. Only sites with FMS–11 support would be required to install them.

Mandatory patches for certain portions of RSTS/E, such as the monitor, can be installed automatically during system generation. The system generation dialogue includes questions that allow you to update code as you install it. DIGITAL advises you to apply all the available updates at this time, to be sure your software is as current as possible. (Refer to the *RSTS/E System Generation Manual* for directions on updating software during installation.) Other mandatory patches are applied after you generate your monitor, such as when you install utilities.

10.2.2 Feature Patches

Feature patches let you take advantage of any optional features or tailor the system for your installation. For example, you might want to install a feature patch to LOGIN that lets users at your site enter certain commands without being logged in. Unlike mandatory patches, feature patches are always optional: no problems occur if you do not install the update.

Some guidelines for choosing optional feature patches are:

- Be cautious about any feature patches that might decrease system security. (This is especially true for sites that handle sensitive data, such as payroll files.)
- Choose feature patches that are the most useful for your system. (Certain feature patches apply more to some systems than to others.) Consult the available reference documents (described in Section 10.7) to decide which patches are best for your system.

10.3 What Are Update Kits?

Update kits, sometimes known as patch kits, are issued periodically to supported RSTS/E customers. These kits contain all the updates that have been published from the release of the current version up to the present.

The update kit provides everything you need for updating RSTS/E software, including a text file with specific update instructions. Update kits are available on the same media as distribution kits.

The two main components in an update kit are:

- The update files themselves, which contain corrections to programs. An update file may be either a replacement file for a module that was distributed with the system or a patch that will actually modify an existing module.
- Command files, which read update files and install them in the software. Command files often apply groups of related update files from the kit.

10.4 When are Update Kits Released?

DIGITAL supplies a kit called "Update Kit A" with each new release of RSTS/E. This kit contains updates to optional layered products, as well as feature patches. In addition, Update Kit A may contain mandatory patches to RSTS/E to correct problems discovered after the new software was completed.

Subsequent update kits are labeled in alphabetic sequence. For example, the first kit distributed after a release is called Update Kit B.

The contents of each update kit are cumulative. This means that each new update kit includes updates from all previous kits, as well as any new ones. Updates from previous kits that have already been installed may generate an error during their installation. These errors are to be expected – updates already installed are not reinstalled – and no harm is done to the software.

Three documents related to updating RSTS/E software are published with each new release: the *RSTS/E Release Notes*, the *RSTS/E Maintenance Notebook*, and the *RSTS/E Software Dispatch Review*. These documents are described in more detail in Section 10.7.

DIGITAL publishes articles in the monthly *RSTS/E Software Dispatch* (see Section 10.7) to describe known software problems. The articles identify the reason for a problem and, in some cases, tell how to install a correction manually. A manual correction is published only for critical problems when no alternative is available. All other problems are corrected only by the update kits, and the article would present any known "work-arounds."

You would install updates manually from an article only if you needed to update your software immediately. If it is known that an update will be distributed, then the article includes the name of the kit that will contain the

update file. (Be aware, however, that DIGITAL may not issue updates to correct a problem when it is only feasible to include the software correction in the next release.) Articles also document any guidelines, notes, or restrictions that may apply to the software.

10.5 Automated Patching

Automated patching allows you to install updates automatically from a kit, rather than typing in patches by hand.

Automated patching has three major advantages over manual patching: it saves you time, it is easier to use, and it is less error prone. You simply specify a command file and let a patching program enter the patch automatically. Using the command file helps you avoid making typographical errors. (Manual patching primarily involves typing numbers and symbols, which can be hard to check for mistakes.)

10.5.1 Prebuilt Tasks and Replacement Modules

A *prebuilt task* is a DIGITAL–supplied executable program that runs under the RSX run-time system. A *replacement module* is a replacement for a whole program as the program was originally released, rather than updates to the code. Where possible, DIGITAL ships both .BAS (source module) and .TSK (prebuilt task) files to make the correction.

For example, the BACKUP package (which consists of several programs) might have some errors corrected by replacement modules. DIGITAL would ship new versions of the programs to replace any defective programs in the BACKUP package.

The programs BUILD and PBUILD (described in Part II) apply replacement modules the same way they apply update files that replace code. The advantage to you is that replacement modules are faster to install than patches to code because replacement modules are already built and are distributed as a single unit.

Both .BAS and .TSK files are provided as replacement modules on an update kit. Patching programs install the appropriate file (.BAS or .TSK) based on what you specify as the run-time system for the module.

For example, when an update is distributed for a BASIC–PLUS program, such as SYSTAT, one replacement module is named SYSTAT.TSK and the other is named SYSTAT.BAS. The two SYSTAT files on the update kit represent two ways to update the original SYSTAT program.

The .TSK files are for users who specify RSX as their primary run-time system. By contrast, the .BAS files are for users who:

- Are applying feature patches
- Want to apply their own modifications to the software
- Compile the .BAS file under BASIC–PLUS or BASIC–PLUS–2

Updating software with .TSK files is much faster than with .BAS files. The RSX file (SYSTAT.TSK) is copied over directly from the patch account, whereas the BASIC–PLUS version (SYSTAT.BAS) must be compiled (under BASIC–PLUS or BASIC–PLUS–2) in order to replace the program.

10.5.2 Editing an Update File

Some feature patches require you to supply system-specific or optional information before you can install them. (This is true whether you install the update from a kit or enter it manually from an article.)

Fortunately, editing an update file is as easy as editing any text file. All you need to do is use a text editor (such as EDT) to modify the contents of the update.

NOTE

Be sure to copy the file from the update kit before editing the file. You should never edit the original files supplied on distribution or update kits.

For example, the following update file PA0301.003 allows you to specify a minimum cache residency time in seconds. (Remember, this is only an example. Do not apply this update to your system.) An exclamation mark (!) precedes the line with the variable "n." and indicates that the line will be interpreted as a comment until it is edited. The update will not work until you supply a value for "n." and delete the exclamation mark.

The instructions in the update file preceding the code explain that the file requires editing, although this example shows only the section of code that contains the variable to be changed. The instructions also provide any specific directions, such as the range of possible values for "n.".

```
Module name? RSTS
Base address? ..CAGE
Offset address? 0
 Base    Offset  Old        New?
!??????  000000  000074  ? n.
??????  000002  ??????  ? ^Z
Offset address? ^Z
Base address? $$0301
Offset address? 0
 Base    Offset  Old        New?
??????  000000  ??????  ? Q!4
??????  000002  ??????  ? ^C
```

Use the editor of your choice to edit the update file.

The following example shows how to edit the file, using line editing with EDT from the DCL keyboard monitor. The example changes the value of

"n." to "30." and deletes the exclamation mark. (In the real example, the file would be larger than 12 lines because it would contain comments in the beginning.)

```
$ EDIT PA0301.003 (RET)
    1            Module name? RSTS
*"n." (RET)
    5            !??????  000000  000074  ? n.
*s/n./30./ (RET)
1 substitution
    5            !??????  000000  000074  ? 30.
*s/!// (RET)
    5            ??????  000000  000074  ? 30.
1 substitution
*EXIT (RET)
PA0301.003      12 lines

$
```

Now you can install the edited update file.

10.6 A Word About Manual Patching

You have to apply patches manually if you install:

- "Paper patches" from articles. Section 10.7 describes reference documents that include updates you can type in manually.
- Patches to non–DIGITAL software.

Manual patching can be an advantage if you want to install a feature patch, or if a mandatory patch is necessary for your installation. (Manual patches are provided only for critical problems.)

However, there are several disadvantages when you type in a patch instead of using a command file:

- The chances for typographical errors increase. You could enter incorrect values, causing new problems to occur, which you may not recognize as being yours.
- Typing in a patch takes longer than copying it from a tape.
- You may not be able to "undo" an incorrect patch because you may not know the value that was there before you replaced it with a new value. (Because of the possibility of errors, you should keep a hard-copy log of updates installed manually.)

  For example, if you entered an instruction incorrectly, the software could fail, and you may not be able to find the instruction you mistyped. Sometimes the only way to "undo" a bad patch is to reinstall the distributed version of the software. In the case of a monitor patch, you would have to do another system generation.

RSTS/E patching programs warn you of most errors before they take effect, so you can correct them or start over. For example, the CPATCH program uses "checksums" to validate your input against the existing code. Checksums note any discrepancies between your input and what CPATCH is expecting, and notifies you right away in case of errors (although only after you exit). In addition, ONLPAT gives you an error message if you enter an incorrect module name or base address. However, when installing patches manually, you have to verify addresses and previous contents yourself.

When you use manual patching to install a "paper patch," type exactly what is printed. This way you do not need to fully understand the commands used for manual patching. If you catch a mistake before pressing the RETURN key on each line, you can use the standard keyboard editing characters, such as CTRL/R, CTRL/U, and the DELETE key. Furthermore, you can:

- Press CTRL/C and start the patch over. Note that CTRL/C is different from uparrow C (^C) which can be used as a response in the ONLPAT dialogue.
- Reenter input if you cause an error message and are reprompted.

10.7 Reference Documents

The following sections describe the documents that DIGITAL provides to inform you of the latest RSTS/E patches. The first three documents are distributed with each release, the fourth is distributed monthly, and the fifth is distributed on each update kit.

10.7.1 The RSTS/E Release Notes

The *RSTS/E Release Notes* describe new features in the latest release of RSTS/E and explain differences between the latest version and previous version. *RSTS/E Release Notes* contain information directed primarily to a system manager, as well as special information required for generating RSTS/E.

The *RSTS/E Release Notes* are included with every version of RSTS/E. System managers are strongly encouraged to read them before installing the system.

10.7.2 The RSTS/E Maintenance Notebook

The *RSTS/E Maintenance Notebook* is published with each release. At release time, the *RSTS/E Maintenance Notebook* contains notes, feature patches, and mandatory patches to the software supplied on a RSTS/E distribution kit (RSTS/E and its bundled products). It establishes a notebook so you can add documentation corrections, software problems and solutions, and programming notes. You should review it to see which update articles and notes apply to your installation.

10.7.3 The RSTS/E Software Dispatch Review

The *RSTS/E Software Dispatch Review* is distributed with every new release, to bring layered product patches and articles up to date. It contains all information published in previous monthly issues of the *RSTS/E Software Dispatch* that is still valid.

Whereas the *RSTS/E Software Dispatch Review* describes only optional layered software, the *RSTS/E Maintenance Notebook* contains RSTS/E notes and feature patches. DIGITAL recommends that system managers combine the *RSTS/E Maintenance Notebook* first with the *RSTS/E Software Dispatch Review* and then with monthly issues of the *RSTS/E Software Dispatch*.

10.7.4 The RSTS/E Software Dispatch

The *RSTS/E Software Dispatch* is a monthly publication that keeps you aware of the latest changes to software and documentation between software releases. It contains mandatory and feature patches, notes on replaced modules, documentation corrections, and programming notes. In addition, it includes announcement articles, software restrictions, and SPDs (Software Product Descriptions).

To keep your system documentation up to date, you should pull the *RSTS/E Software Dispatch* apart when you receive it and file the articles in their proper sequence in your *RSTS/E Maintenance Notebook*.

10.7.5 The PATCHn.DOC File

The first file on an update kit is named PATCHn.DOC, where "n" is the letter of the kit. (For example, PATCHB.DOC is the first file on Update Kit B.) PATCHn.DOC is an ASCII text file that describes the contents of the kit and provides any special instructions for applying the update files.

PATCHn.DOC provides the most recent and specific information available about the current updates for RSTS/E. DIGITAL recommends that you print a copy for directions on updating software from that kit.

Part II Programs for Updating RSTS/E Software

Table 10–1 lists the programs available for updating RSTS/E software. The sections following Table 10–1 provide more detail and show examples of each program.

Table 10–1: Programs for Updating RSTS/E Software

Program	Function
PATCPY	Copies "packages" of files from an update tape to disk. (Updates can be applied only from disk – not from tape – except during system generation.)
ONLPAT	Updates binary files, such as monitors (.SIL), run-time systems (.RTS), or tasks (.TSK).
BUILD	When used with the /PATCH option, builds new software and inserts any patches in the process, or copies .TSK files if appropriate.
PBUILD	Updates any RSTS/E system software that is already installed.
CPATCH	Applies feature patches to BASIC–PLUS source programs.

10.8 Using the PATCPY Program

PATCPY lets you copy files from the update tape to disk. You need to use PATCPY only when your update kit is supplied on tape, or if you do not have enough disk drives to hold the distribution disk and the patch disk. RSTS/E can apply patch files only from disk media; if you receive your update kits on disk, you do not need to use PATCPY.

With PATCPY, you can copy files in groups: you cannot do this as easily with the COPY or PIP commands. You can specify any number of "packages" and individual update files to be copied from the kit. Each update kit includes the latest PATCPY program, to include any packages that might have been redefined for the current kit. When you select a package, PATCPY copies all the files associated with the package over to disk.

Two versions of the PATCPY program are supplied with each update kit:

- PATCPY.BAS – BASIC–PLUS source program
- PATCPY.TSK – an executable file

If your primary run-time system is BASIC–PLUS, you can run PATCPY directly from the tape:

```
Ready

RUN MM0:$PATCPY (RET)
```

If instead you want to run PATCPY.TSK, you need to copy the program to disk:

```
$ COPY MM0:$PATCPY.TSK [200,200]*.* (RET)
[File MM0:[1,2]PATCPY.TSK copied to [200,200]PATCPY.TSK]

$ RUN [200,200]PATCPY (RET)
```

The program works the same way in both cases.

As previously mentioned, you can copy "packages" with PATCPY. A package is a collection of mandatory updates and command files that apply to a RSTS/E program. To illustrate the concept of packages, consider the first few lines from a directory of an update kit, in this case mounted on MM0:.

```
$ DIR MM0:[*,*] (RET)

 Name .Typ     Size     Prot      Name .Typ     Size     Prot     MM0:[1,2]
PATCHA.DOC       17    <155>    PATCPY.BAS       57    <155>
PATCPY.TSK       50    <155>    MONITR.CMD       14    <155>
BASIC .CMD        1    <155>    DECNTC.CMD       14    <155>
RJ2780.CMD       21    <155>    RSXRTS.CMD        1    <155>
INIT  .CMD        1    <155>    PA0101.001        4    <155>
PA0101.002        2    <155>    PA0101.003        2    <155>
PA0101.004        2    <155>    SYSGEN.CMD        1    <155>
MONITR.CMD       14    <155>    PA0301.001        2    <155>
PA0301.002        2    <155>    PA0301.003        2    <155>
                                      .
                                      .
                                      .
```

The first file on the update kit is "PATCHA.DOC," which contains the instructions for applying Update Kit A. The next two files are the source and executable versions of PATCPY. The rest of the files on the kit contain updates to software. (The actual directory listing is much longer than the excerpt shown here.)

The file names MONITR, BASIC, and so forth correspond to package names. These packages contain only mandatory patches. Any feature patches associated with each package are listed individually after the package name, in the format PAnnnn.nnn. For example, the feature patches for INIT are PA0101.001 through PA0101.004. When you specify a package name with PATCPY, any feature patches for that package are also copied over. (You can also copy a group of feature patches without copying the package itself.)

Note that only the mandatory patches are installed automatically with an update program. You must explicitly name any feature patch files in order for them to be installed.

The following is an example of using PATCPY:

```
$ RUN [200,200]PATCPY (RET)
PATCPY  V8.0    RSTS/E  V8.0  Mugwump

Enter distribution device/PPN<SY:[1,2]>: (RET)
Enter output device/PPN <SY:[200,200]>: (RET)
Packages to patch? ? (RET)
```

(continued on next page)

```
Valid Package names are:

ALL             INIT            SYSGEN          MONITR          BASIC
RJ2780          DECNET          TECO            EXEC            BUILD
BIGPRG          BACKUP          SPLER           SPL             DEVTST
DCL             RSX             RT11            RMS             SORT
*STANDARD*      BP2             COBOLV4.4       DMS500V2.1      DECAL
DECMAIL         DIBOL           DTRV2.4         FORTRAN         F77V4.1
INDENT          3271V2.1

Packages to patch? SYSGEN (RET)
Other wild card strings? PA01??.*,PA1012.007 (RET)
Copying MM0:[1,2]PA0101.001 to SY:[200,200]PA0101.001
Copying MM0:[1,2]PA0101.002 to SY:[200,200]PA0101.002
Copying MM0:[1,2]PA0101.003 to SY:[200,200]PA0101.003
Copying MM0:[1,2]PA0101.004 to SY:[200,200]PA0101.004
Copying MM0:[1,2]SYSGEN.CMD to SY:[200,200]SYSGEN.CMD
Copying MM0:[1,2]PA1012.007 to SY:[200,200]PA1012.007
 6 files copied
Copy operation complete

$
```

In this example:

- PATCPY is run from [200,200], where it was copied from the update kit on MM0:.
- The distribution device is account [1,2] on the system disk.
- Files are copied to SY:[200,200], which is the default storage location for update files.
- The question mark (?) asks for help by requesting a list of valid responses. PATCPY then displays the available package names. Note that package names may change with each update kit, depending on the status of a layered product (new release, support ceasing for previous version, or other reasons).

  Be aware that not all RSTS/E programs have mandatory patches. (This means you could specify a valid package name but find that the package is not copied, because it is not included on the kit.) Refer to PATCHn.DOC for the list of available packages.
- The only package you specify is SYSGEN, which has no feature patches associated with it.

  If you specify ALL instead of individual package names, then the entire contents of the update kit are moved to SY:[200,200]. The kit contains all update files for products distributed on RSTS/E, including optional layered software. You might specify ALL if you have plenty of disk space available, and if most of the packages apply to software on your system. By contrast, you might want to select individual packages to save time and space. This is especially true if your system has only a few layered products.
- The question "Other wild card strings?" lets you copy over individual update files. You might want to copy individual files for feature

patches. In this example, you specify all the feature patches for INIT and one other feature patch (PA1012.007, which is included further down in the list of patch files).

- PATCPY then copies the files you selected from the update kit to the specified patching location (SY:[200,200] in the example). Notice that PATCPY searches the directory from start to finish and copies files in the order it finds them. A message appears as each file is copied.

  When it finishes, PATCPY prints the number of files copied, followed by "Copy operation complete," and returns you to the keyboard monitor prompt. At this point, you can use the other programs for updating RSTS/E software.

10.9 Using the ONLPAT Program

ONLPAT updates binary files, such as monitors (.SIL), run-time systems (.RTS), and tasks (.TSK) on a RSTS/E system that is already installed. (For a new system, or for online system generation, refer to the *RSTS/E System Generation Manual* for instructions on updating binary files.) File PATCHn.DOC on the update kit lists the programs you can update with ONLPAT.

This section explains how to use ONLPAT when installing updates from a kit. If you use ONLPAT to manually install updates from a printed article, you simply type the commands exactly as shown. However, if you are updating non–DIGITAL software and need to know the details of using ONLPAT, refer to the complete description in Appendix C.

You can keep a log of the update process by using the following format when you enter a command file name:

logfile = cmdfile

For example:

```
Command file name? ONLPAT.LOG=[200,200]MONITR.CMD(RET)
```

The log file can be useful if you later want to analyze the process, although log files are not necessary. (You might not need a log file if, for example, you are installing updates at a hard-copy terminal.) If you do not want a log file, enter only the name of the command file.

Consider the following example, which automatically applies mandatory patches to a monitor named V80.SIL. (Articles with the sequence numbers 3.x from the *RSTS/E Maintenance Notebook* describe updates similar to the one in this example.)

```
$ RUN $ONLPAT (RET)
Command file name? ONLPAT.LOG=[200,200]MONITR.CMD (RET)
!
!          MONITR.CMD
!          RSTS/E V8.0 Patch Kit 'A'
!
```

(continued on next page)

```
!         Seq 3.x - Executive
!         Includes all mandatory RSTS/E V8.0 Monitor, Terminal Service,
!         File Processor, and Device Driver patches published in the
!         RSTS/E Software Dispatch.
!         Patches included in this file are listed below.
!
!
!         Seq 3.9.1 M, June 1982
!         Executive
!         FMS Monitor Patches
!
!
! SYSTEM MAY CRASH WITH MIXED PICTURE FIELD INPUT
! MANDATORY FMS MONITOR PATCH
! COPYRIGHT (C) 1982 BY DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
!
!
!         This patch will only apply if you generated FMS-11 support in
!         your monitor.  If the FMS-11 code was not included, the patch
!         will fail with a "Module not found in SIL" error after you
!         specify FMS as the Module name.
!
File to patch? [0,1]V80.SIL (RET)
Module name?  FMS
Base address?  $FMSSE
Offset address?  456
 Base    Offset  Old      New?
??????  000456  112402  ? 000137
??????  000460  004767  ? FMSPAT+30
??????  000462  006156  ? 000240
??????  000464  103003  ? 000240
??????  000466  005265  ? 000240
??????  000470  000036  ? 000240
??????  000472  000771  ? 000240
??????  000474  077310  ? 000240
??????  000476  016502  ? _^Z     .     (CTRL/Z for new offset)
Offset address?  6000
 Base    Offset  Old      New?
??????  006000  112002  ? 000137
??????  006002  004767  ? FMSPAT+0
??????  006004  000634  ? 000240
                                .
                                .
                                .
```

In this example:

- You run ONLPAT and specify MONITR.CMD as the command file. ONLPAT looks for MONITR.CMD in the default storage location for update files, which is SY:[200,200]. The file ONLPAT.LOG records the update process as it happens.
- Comment lines about the update file are then displayed. The comments identify the reasons for the update and provide any notes that may be of interest. In the example, the comments state that the update will be installed only if your system software includes FMS–11.
- ONLPAT installs the mandatory patches in MONITR.CMD to the monitor named V80.SIL, which was specified after the "File to patch?" prompt. If you press the LINE FEED key instead of specifying the

name of the file to patch, ONLPAT applies the update file to the monitor currently installed on your system.

- ONLPAT then displays the patch installation process at your terminal. (The process is stored concurrently in the log file, ONLPAT.LOG in this example, if you specified one.) When the process is complete, ONLPAT shows you the number of updates installed or not installed:

```
2 patches installed
4 patches skipped
```

At this point, all the mandatory patches to the monitor V80.SIL are installed.

- ONLPAT then prompts you for another command file. This allows you to continue updating binary software on your system. A CTRL/Z response ends the program and returns you to the keyboard monitor prompt:

```
Command file name? ^Z

$
```

(Be sure to use CTRL/Z instead of CTRL/C to end the program.)

10.10 Using BUILD/PATCH

The /PATCH qualifier of the BUILD program automatically installs updates as you build new software. (If there are no update files for the software you build, then BUILD does not attempt the update procedure.)

The example in this section updates BIGPRG programs as they are built. Note that prebuilt tasks are installed as part of the build procedure. This is done when you specify RSX as the run-time system and you request that CSPCOM be used. The example also shows a module replacement being copied from the patching account SY:[200,200] for the PMDUMP program.

You can create a log file of the build procedure and also specify that the build is to run detached, in the format shown after the "Source Input Device" prompt in the BUILD dialogue:

logfile = dev:/det

For example:

```
Source Input Device <SY:>: ? BIGPRG,LOG=MM0:/DET RET
```

In the following example, only the device (MM0:) is specified. This means there is no log file and the job is not run detached.

```
$ RUN $BUILD RET
BUILD  V8.0 RSTS V8.0     Mugwump
System Build <No> ? RET
Source Input Device <SY:> ? MM0: RET
```

(continued on next page)

```
Library Output Device <SY:> ? RET
Target System Device <SY0:> ? RET
Library Account <[1,2]> ? RET
Control File is ? BIGPRG RET

*** Copying file MM0:[1,2]BIGPRG.CTL to BLD25.TMP ***

Function (Build/Patch, Patch, Build) <Build/Patch> ? RET
Patch file input location <SY:[200,200]> ? RET
Save patched sources <No> ? RET
Run-Time System <RSX> ? RET
Use CSPCOM <YES> ? RET
Additional Control File is <None> ? RET

!*** Processing started on 22-Feb-83 at 11:01 AM ***

>ASSIGN MM0:/MO:1
>ASSIGN SY0:SYSDSK
>ASSIGN SY:SYSTEM
>ASSIGN [1,2]
>ASSIGN MM0:INPUT
>!********** BIGPRG.CTL - LARGE PROGRAM BUILD
! Copyright (C) 1979, 1981, 1982, 1983
! by Digital Equipment Corporation, Maynard, Mass.

>
>
>RUN SY:[1,2]PIP.SAV
*SY:[1,2]FIT.TSK<232>/RTS:RSX=MM0:[1,2]FIT.TSK
*SY:[1,2]FIT.TSK/MO:64=SY:[1,2]FIT.TSK
*^Z
                                        .
                                        .
                                        .

>
>
>!
!         PA1112.CMD
!
!         Seq 11.12
!         PMDUMP
! COPYRIGHT (C) 1983 BY DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
!
! Replace PMDUMP (Seq 11.12)
!
RUN SY:[1,2]PIP.SAV
*SY:[1,2]PMDUMP.TSK<232>/RTS:RSX/MO:64=SY:[200,200]PA1112.TSK
*^Z
>
>
>
>
>!*** BUILD Complete ***

!*** Processing ended on 22-Feb-83 at 11:13 AM ***
```

In this example:

- You run BUILD for individual programs, rather than doing a system build. The input device is the update kit on MM0:. The system disk is the library output device, and SY0: is the target system device. The library account is [1,2].

- The control file specified is BIGPRG. If you do not specify a file type when you supply the control file name, BUILD looks for the .CTL version. (Note the message stating that BUILD copied BIGPRG.CTL from the update kit to the temporary file BLD25.TMP. The 25 in BLD25.TMP is the number of the job running BUILD.)
- You specify BUILD/PATCH and the default location for update files (SY:[200,200]). It is usually not necessary to save the patched sources. When you specify RSX and CSPCOM, with no additional control files, the building and patching process begins.
- The file PA1112.TSK, which is a replacement module from the update kit, is included in the BUILD/PATCH operation.
- A message is displayed when the process is complete.

10.11 Using the PBUILD Program

You use the PBUILD program to update existing software. Refer to PATCHn.DOC to determine which software is updated with PBUILD.

The following two examples update existing BIGPRG programs. The first one specifies RSX as the primary run-time system and uses BASIC–PLUS–2 to compile the updated software. The second shows BASIC–PLUS as the primary run-time system. If you do not specify a file type when prompted for a command file, PBUILD looks the for .CMD version.

You can create a log file of the PBUILD procedure and also specify that the build is to run detached, in the format shown after the command file prompt (#) in the dialogue:

logfile = cmdfile/det

For instance:

```
#BIGPRG.LOG=[200,200]BIGPRG/DET (RET)
```

In the following example, however, only the input command file (SY:[200,200]BIGPRG) is specified. This means there is no log file and the job is not run detached.

1. Example using the RSX run-time system and the BASIC–PLUS–2 compiler:

```
$ RUN $PBUILD (RET)
PBUILD     V8.0 RSTS V8.0     Mugwump
Read files to patch from <SY:[1,2]>: MM0: (RET)
Compile patched programs <YES>: (RET)
Library device <SY:[1,2]>: (RET)
System device <SY0:[1,2]>: (RET)
Save patched sources <NO>: (RET)
#[200,200]BIGPRG (RET)

*** Copying file SY:[200,200]BIGPRG.CMD to PBLD27.TMP ***
```

(continued on next page)

```
Run-Time System <RSX> ? RET
Use CSPCOM <YES> ? NO RET
Name of BASIC-PLUS-2 compiler <$BP2IC2> ? RET
Additional Control File is <None> ? RET

!*** Processing started on 23-Feb-83 at 11:22 AM ***

>ASSIGN MM0:/M0:1
>!
!          BIGPRG.CMD
!
!          Seq 11.0
!          Big Programs Patch Command File
! COPYRIGHT (C) 1983 BY DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
!

>
>!         PA1112.CMD
!
!          Seq 11.12
!          PMDUMP
! COPYRIGHT (C) 1983 BY DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
!

>RUN [1,2]BP2IC2.TSK

PDP-11 BASIC-PLUS-2 V2.0-00

BASIC2

SCALE 0
 Scale factor has been set to  0

BASIC2

OLD SY:[200,200]PA1112.BAS

BASIC2

COMPILE SY:[1,2]PMDUMP.OBJ/OBJ/CHA/LIN/NODEB/WOR/NOCRO/NOLIS/FLAG:NODEC
PA1112     11:23 AM 23-Feb-83

BASIC2

SCRATCH

BASIC2

EXIT
>

>RUN SY:[1,2]TKB.TSK

TKB>SY:[1,2]PMDUMP,TSK/FP=SY:[1,2]PMDUMP,OBJ,SY:[1,11]BP2OTS,OLB/LB

TKB>/

Enter Options:
TKB>UNITS=12

TKB>ASG=SY:5:6:7:8:9:10:11:12

TKB>//
>RUN SY:[1,2]PIP.SAV
*SY:[1,2]PMDUMP.OBJ/DE:NO
*^Z
```

(continued on next page)

```
>RUN SY:[1,2]PIP.SAV
*SY:[1,2]PMDUMP.TSK<232>/RE
*^Z
>
>
>
>!*** PBUILD Complete ***

!*** Processing ended on 23-Feb-83 at 11:25 AM ***

$
```

In this example:

- When you run PBUILD, you specify that the update files are stored on MM0: and that the programs are to be compiled. The library device is [1,2] on the system disk, and the system device is on SY0:. There is usually no need to save the patched sources.
- You specify BIGPRG as the command file, which is in the default storage location for update files (SY:[200,200]). (The update files were already copied from MM0: to SY:[200,200] with PATCPY.) If you do not specify a file type when you supply the command file name, PBUILD looks for the .CMD version. (Note the message stating that PBUILD copied BIGPRG.CMD from the update kit to the temporary file PBLD27.TMP. The 27 in PBLD27.TMP is the number of the job running PBUILD.)
- You specify RSX as the run-time system and BASIC–PLUS–2 as the compiler. When you respond that there are no additional control files, the updating process begins. PBUILD displays a message when the process is complete and returns you to the keyboard monitor prompt.

2. Example using the BASIC–PLUS run-time system:

```
$ RUN $PBUILD (RET)
PBUILD    V8.0 RSTS V8.0    Mugwump
Read files to patch from <SY:[1,2]>: MM0: (RET)
Compile patched programs <YES>: (RET)
Library device <SY:[1,2]>: (RET)
System device <SY0:[1,2]>: (RET)
Save patched sources <NO>: (RET)
#[200,200]BIGPRG (RET)

*** Copying file SY:[200,200]BIGPRG.CMD to PBLD25.TMP ***

Run-Time System <RSX> ? BASIC (RET)
Additional Control File is <None> ? (RET)

!*** Processing started on 22-Feb-83 at 11:13 AM ***

Ready

ASSIGN MM0:/M0:1

Ready

!
!          BIGPRG.CMD
!
```

(continued on next page)

```
!          Seq 11.0
!          Big Programs Patch Command File
! COPYRIGHT (C) 1983 BY DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
!
!
!        PA1112.CMD
!
!        Seq 11.12
!        PMDUMP
! COPYRIGHT (C) 1983 BY DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
!
! Replace PMDUMP (Seq 11.12)
!

SCALE 0

Ready

OLD SY:[200,200]PA1112.BAS

Ready

COMPILE SY:[1,2]PMDUMP<232>

Ready

!*** PBUILD Complete ***

!*** Processing ended on 22-Feb-83 at 11:20 AM ***

Ready
```

The preceding example works the same way as the first PBUILD example, except that you specified "BASIC" as the run-time system. Note that the file PA1112.BAS, which is a replacement module from the update kit, is included in the PBUILD operation.

10.12 Using the CPATCH Program

CPATCH is a specialized text editor that lets you install feature patches to BASIC–PLUS source programs.

This section explains how to use CPATCH when installing updates from a kit. If you use CPATCH to manually install updates from a printed article, you simply type the commands exactly as shown. However, if you are updating non–DIGITAL software and need to know the details of the process, refer to the complete description of CPATCH in Appendix C.

The following example shows the use of CPATCH to install an update to LOGIN. Articles with the sequence number 10.12 from the *RSTS/E Maintenance Notebook* contain similar updates. The example assumes that your primary run-time system is BASIC-PLUS, although the procedure is similar if you have another primary run-time system.

The example shows that you can keep a log of the update process by entering a command file name in the following format:

logfile = cmdfile

For example:

```
File to patch - PA1012.LOG=[200,200]LOGIN.BAS (RET)
```

The log file can be useful if you later want to analyze the update process. If you do not want a log file, type only the name of the command file. (You might not need a log file if, for example, you are installing updates at a hard-copy terminal.)

CPATCH installs the update from the kit as follows:

```
$ RUN $CPATCH (RET)
CPATCH    V8.0 RSTS V8.0    Mugwump
File to patch - PA1012.LOG=[200,200]LOGIN.BAS (RET)
#[200,200]PA1012.007
*H/2!/V
2!          PROGRAM  : LOGIN.BAS
*H/32240  /DV
32240     DATA HELP, $HELP, 4, 3
*H/32250  /DV
32250     DATA SET, $TTYSET,3, 4
*H/32260  /DV
32260     DATA SYSTAT, $SYSTAT,2, 4
*H/32270  /DV
32270     DATA QUEUE, $QUE, 2, 4
*EX
Patch from _SY:[200,200]PA1012.007 complete.
#^Z
File to patch - ^Z

$
```

In this example:

- You run CPATCH and specify a log file (PA1012.LOG) and the name of the program to update (LOGIN.BAS).
- You then specify the update file (PA1012.007) to be applied to LOGIN.BAS.
- The CPATCH program installs the update and prints a message when the process is complete.

Then you recompile the updated LOGIN program and enter it into the system library:

```
OLD LOGIN (RET)

Ready

COMPILE SY0:$LOGIN<232> (RET)

Ready
```

After you complete these steps, you can save the new version of the program and delete the source (.BAS) version from the public structure. (If you do not save the new version, you can reapply the feature patch.)

Chapter 11
DCL Commands for Disk and Tape Handling

This chapter provides background information about disk and tape handling on RSTS/E. Then it describes the INITIALIZE, MOUNT, and DISMOUNT commands, which are summarized in Table 11–1.

Some uses of INITIALIZE, MOUNT, and DISMOUNT require privilege. (For example, INITIALIZE is privileged when used for disks but is nonprivileged for tapes.) The privileged uses of these commands are described only in this manual, whereas the nonprivileged uses are described both here and in the *RSTS/E DCL User's Guide.*

Table 11–1: DCL Commands for Disk and Tape Handling

Command	Description
INITIALIZE	Initializes a disk or magnetic tape. Use this command to overwrite all existing files with zeros and prepare the medium for writing new files. (See pages 11-3 and 11-9.)
MOUNT	Logically mounts a disk or magnetic tape. Use this command after you physically mount the disk or tape on the device. (See pages 11-11 and 11-18.)
DISMOUNT	Logically dismounts a disk or magnetic tape. Use this command before you physically dismount the disk or tape from the device. (See pages 11-20 and 11-22.)

11.1 Working with Disks and Tapes

The system manager or operator is often the person responsible for loading and unloading disk packs and magnetic tapes. Indeed, *only* a privileged user can load disks on the public structure. Thus, you need to be familiar with commands that prepare tapes and disks for use: initialization, mounting, and dismounting.

11.1.1 Initializing Disks and Tapes

Initializing a disk or tape makes it "like new."

- If you have an old disk or tape to recycle, you initialize it to get rid of all the existing files that were written to it and start afresh.

- If you have a new disk or tape, you initialize it to perform all the necessary actions that make it usable on your system. (The INITIALIZE command descriptions explain these necessary actions.)

Certain new disks require *formatting* before they are initialized. Formatting writes timing and sense marks onto the disk and destroys any information that the disk contains. Disk formatting can be done only with the DSKINT option of INIT.SYS, as described in the *RSTS/E System Generation Manual.*

You must shut the system down to format the following disks before using them on a RSTS/E system:

RK05, RK05F, RP02, RP03, RP04, RP05, and RP06

These disks need to be formatted only once. All other disks have been formatted at the factory and need not be formatted again.

The DSKINT option also creates a RSTS/E file structure on the disk. You can later use the DCL INITIALIZE command to clear the disk of files and recreate the RSTS/E file structure.

The INITIALIZE command:

- Lets you specify the file structure of disks and tapes.
- Has different qualifiers for disk initialization than tape initialization.
- Requires privilege when used for disks.

INITIALIZE for disks is a privileged operation; tape initialization is available to privileged and nonprivileged users. Section 11.2 describes disk initialization. Section 11.3 describes tape initialization; this command is also described in the *RSTS/E DCL User's Guide.*

11.1.2 Mounting and Dismounting Tapes and Disks

Each time you want to use a different tape or disk than is currently on the device, you must logically dismount the current disk or tape (using the DISMOUNT command), and physically remove the disk or tape. Then you physically mount the new disk or tape and turn the drive online.

After physically mounting and activating the medium, you must logically mount the new disk or tape (using the MOUNT command). This allocates the drive for whatever use is appropriate. For example, a tape drive is allocated for use only through your account. A disk drive with a pack initialized and mounted as public becomes part of the public structure. It can be used by anybody on the system.

The MOUNT command prepares an initialized disk or tape for processing by system commands or user programs. The DISMOUNT command releases a disk or tape that was previously accessed with a MOUNT command. Both MOUNT and DISMOUNT have different qualifiers when used with disks or tapes. MOUNT and DISMOUNT are described in Sections 11.4 through 11.7.

11.2 INITIALIZE for Disks

The INITIALIZE command creates a RSTS/E file structure on a disk. That is, it prepares a disk so that files can be written to it and read from it. (Be aware that INITIALIZE overwrites with zeros any data previously written to the disk.)

In creating a RSTS/E file structure for the disk, INITIALIZE writes several structures to the disk. For example, INITIALIZE creates a User File Directory (UFD) for account [0,1] that contains entries for the files SATT.SYS (the storage allocation table) and BADB.SYS (the bad block file).

Format

INITIALIZE device-name[:] pack-id

Command Qualifiers	Defaults
/[NO]EXERCISE = n	/EXERCISE = FULL
/[NO]RETAIN	/RETAIN
/CLUSTER_SIZE = n	n = device cluster size
/MFD_CLUSTER_SIZE = n	n = 16
/DATE = ACCESSED	/DATE = MODIFIED
/DATE = MODIFIED	/DATE = MODIFIED
/INDEX = position	/INDEX = MIDDLE
/PRIVATE	/PRIVATE
/PUBLIC	/PRIVATE
/[NO]QUERY	/QUERY
/[NO]WRITE	/WRITE

Command Parameters

device-name[:]

Specifies the name of the drive on which the disk is physically mounted.

pack-id

One to six alphanumeric characters to be used when logically mounting the disk. RSTS/E uses this pack-id as a system-wide logical name for the disk.

Command Qualifiers

/EXERCISE = n
/NOEXERCISE

Controls whether to check for bad blocks by "exercising" the disk. This means that the number of patterns you select (n) are written to the disk, and then each block is read to ensure that the patterns were written correctly. If the patterns read do not match the patterns written, the block

is assumed to be bad. Any bad blocks found are added to the "bad block" file (BADB.SYS in account [0,1] on the disk). Blocks listed in BADB.SYS are not used for storing data.

You can have the INITIALIZE command run from zero to three patterns by specifying 0, 1, 2, or 3 for n. Each pattern consists of three octal words; the pattern used for each of these numbers is:

Pattern	Three Octal Words
1	155555 133333 066666
2	133333 066666 155555
3	066666 155555 133333

For example, /EXERCISE = 2 causes pattern 1 to be written to and read from the disk, followed by pattern 2 being written to and read from the disk.

You can also select /EXERCISE = FULL, and all three patterns will be used, one at a time. That is, each block is written to and read from the disk three times, each time with a different pattern.

/EXERCISE = FULL is the default. Using FULL increases the probability that all bad blocks are found, and decreases the probability that you will lose information later by writing into a bad block. (/EXERCISE = FULL is equivalent to /EXERCISE = 3.)

/NOEXERCISE indicates that you do not want to check for bad blocks. For example, if you previously initialized the disk, and feel that the current bad block file is accurate, you can save time by specifying the /NOEXERCISE qualifier. This still writes one pattern all over the disk, but it does not read it back in. (/NOEXERCISE is equivalent to /EXERCISE = 0.)

/RETAIN
/NORETAIN

Controls whether to use a previously initialized disk's bad block file. A disk that has not been previously initialized has no bad block file. In this case, /RETAIN and /NORETAIN are meaningless, and are ignored if specified. /RETAIN is the default on a previously initialized disk.

If you use both the /RETAIN and /EXERCISE qualifiers, the INITIALIZE command keeps the old bad block file and adds any additional bad blocks it finds while it is checking the disk for bad blocks.

If you specify /NOEXERCISE, then you should use /RETAIN. Because /NOEXERCISE does not exercise the disk for bad blocks, you should use /RETAIN to keep the existing bad block information.

The /NORETAIN qualifier is useful if problems in a disk drive have been causing apparent bad blocks on a disk. After you learn that the problem

is with the drive, you can recover space on the disk by using /NORETAIN.

/CLUSTER_SIZE = n

Declares the pack cluster size, which is the minimum allocation unit, in 512-byte blocks, for the disk. A file written to this disk takes a multiple of n blocks.

In general, the cluster size specified (n) must be a power of 2 (1, 2, 4, 8, 16), equal to or greater than the device cluster size, but not greater than 16. If you do not specify a cluster size, the device cluster size is assumed. See Table 11–2 for the list of device cluster sizes.

The /CLUSTER_SIZE = n option affects performance. In general, a large pack cluster size speeds disk I/O operations, but wastes disk space. For example, if you declare /CLUSTER_SIZE = 16 for a disk, a file consisting of 8193 bytes (16 512-byte blocks plus one byte) requires 2 pack clusters, or 32 blocks. A /CLUSTER_SIZE = 8 for the same file would require three clusters, or 24 blocks, of disk storage.

Table 11–2: Disk Size and Cluster Size

Disk Type	Device Cluster Size	Acceptable Pack Cluster Size (/CLUSTER_SIZE = n)	Total Device Size (blocks)
RX50	1	1, 2, 4, 8, 16	800
RF11	1	1, 2, 4, 8, 16	1024 times number of platters
RS03	1	1, 2, 4, 8, 16	1024
RS04	1	1, 2, 4, 8, 16	2048
RK05	1	1, 2, 4, 8, 16	4800
RK05F	1	1, 2, 4, 8, 16	4800 for each unit; 2 units per drive
RL01	1	1, 2, 4, 8, 16	10220
RL02	1	1, 2, 4, 8, 16	20460
RD51	1	1, 2, 4, 8, 16	21600
RC25	1	1, 2, 4, 8, 16	50902 for each unit; 2 units per drive
RK06	1	1, 2, 4, 8, 16	27104
RK07	1	1, 2, 4, 8, 16	53768
RP02	2	2, 4, 8, 16	40000
RP03	2	2, 4, 8, 16	80000
RM02	4	4, 8, 16	131648
RM03	4	4, 8, 16	131648
RP04	4	4, 8, 16	171796
RP05	4	4, 8, 16	171796
RA80	4	4, 8, 16	237208
RM80	4	4, 8, 16	242575
RP06	8	8, 16	340664
RA60	8	8, 16	400175
RM05	8	8, 16	500352
RA81	16	16	888012

/MFD_CLUSTER_SIZE = n

Declares the minimum allocation unit, in 512-byte blocks, for the Master File Directory, or MFD. The MFD is a special "catalogue" structure created during initialization. Along with other key structures, it is updated each time accounts and files are added to or deleted from the disk. Because the MFD is accessed often, you increase performance when the MFD cluster size is large. A small MFD cluster size, on the other hand, can save some disk space.

The MFD cluster size (n) can be 4, 8, or 16; it must be greater than or equal to the value entered for the CLUSTER_SIZE qualifier. The default MFD cluster size is 16.

/DATE = ACCESSED
/DATE = MODIFIED

The /DATE qualifier lets you specify whether to maintain files by their date of last access (= ACCESSED) or of last modification (= MODIFIED). The default is /DATE = MODIFIED.

/INDEX = position

Positions the SATT.SYS file on the disk. SATT.SYS is the storage allocation table created in account [0,1] during initialization.

The position argument can be:

BEGINNING	Puts SATT.SYS at the beginning of the disk, which avoids "fragmenting" the disk. That is, more contiguous space is available to store other files on the disk.
MIDDLE	Puts SATT.SYS in the middle of the disk. For moving-head disks, it helps to locate SATT.SYS near the middle of the disk, to reduce average seek times for the disk heads. MIDDLE is the default if you do not specify the /INDEX option.
n	Puts SATT.SYS at device cluster number n, where n can range from 1 to the total device size divided by the device cluster size. (See Table 11–2 for these values.)

/PRIVATE

Allows only users who have accounts on a disk to access it. If you do not specify /PUBLIC or /PRIVATE, then /PRIVATE is assumed.

You can create a system disk by initializing the disk as /PRIVATE and then using the REACT program to create and position accounts [1,1] and [1,2]. Note that in this case you must transfer files to the disk and "hook" the disk in order to make it a RSTS/E system disk.

/PUBLIC

Allows anyone with an account on the system to access the disk.

You can use /PUBLIC to identify any disk on the system as public, except the system disk. When the disk is mounted, it is considered part of the public structure. Mounting a public disk requires privilege. After it is mounted, the disk is a logical extension of the system disk, and any user can create files on it.

In general, DIGITAL recommends against using /PUBLIC, because it can degrade performance over time. The following paragraphs describe some considerations if you decide to select /PUBLIC for disks on your system.

If your system disk is nearly full, and you want to dedicate one or more additional disks to general file storage, then you might add public disks to your system. Users generally are not aware of whether their files are on the system disk or a public disk.

Because a public disk is an extension of the system disk, it should always remain mounted during timesharing. In addition, you must remount it every time you restart the system. If you dismount the disk during timesharing, some users' files will mysteriously (to them) disappear. Dismounting a public disk will cause disruption unless you transfer all of the files on the disk to the system disk or to some other public disk.

Before adding a public disk to your system, consider the alternative of adding a private disk for use by specific users. DIGITAL recommends this alternative because it yields better system performance. However, it means that you must explicitly create accounts on the private disk (using REACT). In addition, users who have accounts on the disk must explicitly refer to that disk in file specifications.

Because of these considerations, the system provides several safety checks on initializing and mounting a public disk. First, you must be privileged to do so. Second, you must specify /PUBLIC when you initialize the disk *and* when you mount the disk.

/QUERY
/NOQUERY

When you use the /QUERY qualifier, INITIALIZE displays the characteristics of a disk you have selected that already has a RSTS/E file structure, and asks if you still want to initialize the disk.

Thus, /QUERY lets you verify that you are initializing the disk you intended, thereby keeping you from accidentally wiping out the wrong disk. If you specify /NOQUERY, INITIALIZE simply displays the characteristics of the disk before overwriting its contents. /QUERY is the default.

The display of disk characteristics is similar to:

```
This disk pack appears to be a RSTS/E formatted
disk with the following characteristics:

Pack ID :                THELMA
Pack Cluster Size :      4
Pack is currently :      Private,
                         Update access date on writes,

Proceed (Y or N)?
```

You must respond to the prompt with YES or NO, or some abbreviation of these. NO cancels the initialization; YES requests that it proceed. There is no default response to the prompt.

/WRITE
/NOWRITE

Determines whether the disk should default to read-only (/NOWRITE) or read/write (/WRITE) access when it is mounted. The default is /WRITE.

When mounting a disk, users can override the read-only access by specifying the /WRITE qualifier in the MOUNT command.

11.3 INITIALIZE for Tapes

Use INITIALIZE to prepare a new tape or to recycle a tape that contains no useful files. The INITIALIZE command for tapes overwrites any data on the tape and, for ANSI tapes, writes a label that is used to identify the tape in a MOUNT command. (ANSI tapes require a label.) INITIALIZE allocates the tape drive if it is not already allocated.

INITIALIZE for tape does not require privilege; it is described here for your convenience.

Format

INITIALIZE device-name[:] [label]

Qualifiers	Defaults
/DENSITY = nnn	See discussion.
/FORMAT = ANSI	See discussion.
/FORMAT = DOS	See discussion.

Prompts

Device: magtape[:]

Label: label

Proceed (Y or N)?

Command Parameters

device-name[:]

Specifies the name of the drive on which the tape is physically mounted.

[label]

Specifies the identification label to be encoded on the tape. The label can consist of a maximum of six alphanumeric characters.

An ANSI-format tape requires a label; you are prompted for a label if you do not specify one. This label is then checked against the label you specify when you later use the MOUNT command (see Section 11.5).

DOS tapes do not allow labels; any label you specify for them is ignored.

Command Qualifiers

/DENSITY = nnn

Specifies the density in bits per inch (bpi) at which the tape is to be written. You can specify a density of either 800 or 1600 bpi if the tape drive supports it.

The default density is the lowest bpi at which the controller can operate, unless you applied a patch to make the default 1600 bpi. (See Sequence Numbers 1.1 and 22.19 of the *RSTS/E Maintenance Notebook* for information about these patches.)

/FORMAT = ANSI
/FORMAT = DOS

Specifies the tape format. If you do not specify a format, the system uses the default specified at system generation. As mentioned earlier, ANSI format requires a label; DOS format ignores any label specified.

11.4 MOUNT for Disks

The MOUNT command prepares an initialized disk for processing by system commands or user programs. You must first physically mount the disk on a device drive, put the drive online, and then use the MOUNT command to logically mount the disk.

Only privileged users can mount a disk that has been initialized as public.

For privileged users, MOUNT rebuilds a so-called "dirty" disk, which is a disk that has been physically dismounted without being logically dismounted with the DISMOUNT command. This is the only time that the "dirty" bit is set, although you may choose to rebuild the disk at other times. Refer to the description of the ONLCLN program in Chapter 7 for more information about rebuilding disks.

If MOUNT discovers the disk is dirty, it displays the message:

```
Disk is being rebuilt - wait...
```

The rebuild operation proceeds and the disk is logically mounted when the dollar prompt ($) for the next DCL command appears. However, if the rebuild operation discovers blocks that have been allocated to more than one file (doubly allocated blocks), MOUNT displays a message asking which file to allocate each doubly allocated block to. You can choose a file at that point or stop and mount the disk without rebuilding, to examine the situation further. (See the discussion of /[NO]REBUILD in the following description of MOUNT command qualifiers.)

Format

MOUNT device-name[:] pack-id [logical-name[:]]

Command Qualifiers	Defaults
/PRIVATE	/PRIVATE
/PUBLIC	/PRIVATE
/[NO]SHARE	/SHARE
/[NO]WRITE	See discussion.
/[NO]REBUILD	See discussion.

Prompts

Device: device-name[:]

Pack-id: pack-id

Command Parameters

device-name[:]

Specifies the physical or logical name of the drive containing the disk you want to logically mount.

pack-id

Specifies the one to six character alphanumeric pack identification label written to the disk during initialization. (See the INITIALIZE command description in Section 11.2.) MOUNT then verifies that the pack-id written on the disk is the same as the label you specified in the MOUNT. This process helps make certain that you physically mounted the right disk.

You must include a pack-id to mount a disk. If you omit the pack-id, you will be prompted for it.

[logical-name[:]]

Lets a privileged user assign a system-wide logical name to the disk drive specified by a device name. If you do not specify a logical name, the system uses the pack-id label as a system-wide logical name for the drive. You may want to use this logical name to keep the pack-id secret from nonprivileged users. Nonprivileged users can see logical names assigned to devices in SYSTAT. They can also dismount a private disk as long as they know the pack-id. The specification of a logical name by a nonprivileged user is ignored, but the mount succeeds.

Command Qualifiers

/PRIVATE
/PUBLIC

Declares that the disk you are mounting is accessible to all users on the system (a public disk), or only accessible to users who have accounts on the disk (a private disk). The default is /PRIVATE.

The action taken for /PRIVATE and /PUBLIC depends on two things:

1. Whether the user doing the MOUNT is privileged or nonprivileged
2. Whether the disk was initialized as public or private

In general, the disk is safeguarded, even against a privileged user, from being inadvertently made available for public use.

For Privileged Users:

Privileged users can mount a disk as either public or private. In fact, only privileged users can mount a disk as public, and if that is your

intent, you must specify /PUBLIC. However, even a privileged user cannot override a disk initialized as private by trying to mount the disk using /PUBLIC. If you try, you get the error message:

```
?Can't mount a private disk as public
```

If a privileged user mounts a disk with either the /PRIVATE or /SHARE qualifier (/SHARE is described later in this section), the disk is mounted private regardless of whether it was initialized as public or private.

If a privileged user does not specify /PRIVATE, /PUBLIC, or /[NO]SHARE, then even a disk that was initialized as public is mounted as private. That is, only users who have accounts on the disk can use the disk. A warning message is displayed if the disk was initialized as public:

```
%Public disk mounted as private
```

For Nonprivileged Users

A nonprivileged user can MOUNT a disk that was initialized as private. The user can explicitly declare the usage with the /PRIVATE or /SHARE qualifier; however, this is not required. The user can also mount the disk with the /NOSHARE qualifier, to make the disk accessible only to the job that mounts it (as described later in this section).

A nonprivileged user can mount only private disks. The MOUNT command checks to see how the disk was initialized. If the user tries to mount a disk that was initialized as public, that user gets the error message:

```
?You must be privileged to mount a public disk
```

Furthermore, if a nonprivileged user tries a MOUNT with the /PUBLIC qualifier for a disk initialized as private, the mount fails and the following error message is displayed:

```
?Can't mount a private disk as public
```

Table 11–3 summarizes the possibilities of the /PUBLIC, /PRIVATE, and /[NO]SHARE qualifiers. (If none of these qualifiers is specified, /PRIVATE is assumed.)

/SHARE
/NOSHARE

Controls whether to limit disk access only to the job that mounted the disk (/NOSHARE) or to mount it as shared (/SHARE). A disk mounted as shared is accessible to only those users with an account on the disk.

User is:	INITIALIZE for disk was: Public	Private
Nonprivileged	Any MOUNT returns an error.	MOUNT with /PUBLIC returns an error. MOUNT with no qualifier or with /PRIVATE, /SHARE, or /NOSHARE succeeds.
Privileged	MOUNT with /PUBLIC succeeds: the disk is accessible to all users. MOUNT with /PRIVATE or /SHARE succeeds: the disk is accessible to only those users with an account on the disk. MOUNT with /NOSHARE succeeds; the disk is accessible only to the job that mounts it. MOUNT with no qualifier succeeds as private disk: the disk is accessible to only those users with an account on the disk, and a warning message is displayed.	Same as above.

A disk mounted as shared is the same as a disk mounted as private. A nonprivileged user can mount only a private disk as shared or nonshared. A privileged user can mount a private or public disk as shared or nonshared.

The /SHARE qualifier is equivalent to /PRIVATE. If you do not specify /PRIVATE, /PUBLIC, or /[NO]SHARE, then the default action is to mount the disk as private (shared).

The /NOSHARE qualifier conflicts with either /PUBLIC or /PRIVATE. /SHARE can be used with either /PUBLIC or /PRIVATE; its presence in the same command line as the other two qualifiers has no effect.

WRITE
/NOWRITE

Controls whether data can be written to the mounted disk.

If you specify /NOWRITE, the disk is write-protected, which means that no users can write to the disk. /NOWRITE protects files on the disk; they can only be read from, not written to.

If you specify /WRITE, users who access the disk can write to it. However, if you are a privileged user and the disk is dirty, but you do not request a rebuild (either by default or with the /REBUILD qualifier), the disk is mounted read-only regardless of the /WRITE qualifier. If the disk is dirty and you do request a rebuild, but the drive is write-protected, then the mount fails because the rebuild cannot be performed. A non-privileged user cannot mount a dirty disk; in this case, the system manager should mount it.

When you do not specify /WRITE or /NOWRITE, the default depends on whether the disk was initialized as read/write or read-only, and whether the disk is clean or dirty. For disks initialized as read-only, the default is to always give read-only access (/NOWRITE) to the disk. For disks initialized as read/write, there are a few more conditions to consider. If the disk is clean, and the drive is not write-locked, /WRITE is the default. If the disk is clean and the drive is write-locked, /NOWRITE is the default, and the following warning message is displayed:

```
%Device write protected
```

If the disk is dirty, the default for /[NO]WRITE depends on whether or not you specified the /NOREBUILD option. In general, a dirty disk is mounted read-only (/NOWRITE is the default). Specifically, if you MOUNT a disk with /NOREBUILD and do not specify either /WRITE or /NOWRITE (that is, the disk is dirty and you do not want it rebuilt), the disk is mounted read-only (/NOWRITE is the default). If you do not specify /NOREBUILD, an attempt is made to rebuild the disk. If this operation succeeds, then write access is granted (/WRITE is the default).

/REBUILD
/NOREBUILD

Controls the rebuilding of a disk, regardless of whether the disk was initialized as read/write or read-only. A dirty disk is one that has been physically dismounted (removed from the drive) without being logically dismounted first by a DISMOUNT command.

A DISMOUNT checks to see that no files are currently open on a disk. If files are open, the DISMOUNT does not succeed. If someone makes a mistake and physically removes a disk pack from a disk drive while programs are still using files on the disk, the usual "clean-up" operations that the RSTS/E monitor performs may not have been done. For example, the monitor keeps the file SATT.SYS (the storage allocation table for the disk) in memory and updates it when files are added to or deleted from the disk. If someone physically dismounts a disk before the monitor has written SATT.SYS back to disk, the data in the old SATT.SYS will probably be incorrect.

Rebuilding a disk is a privileged operation which performs "clean-up" operations that are necessary before such a disk can be used again. Specifically, a rebuild operation:

1. Deletes files marked for deletion.
2. Locates blocks that have been allocated to more than one file. The system displays a message listing the files to which such blocks have been allocated, and asks you to delete all but one of the files.
3. Deletes invalid directories.
4. Deletes all files that have the type .TMP, are marked for deletion, or have no accounting entry.
5. Builds a new storage allocation table (SATT.SYS), to show current file allocations (after deletions accomplished by rebuilding).
6. Zeroes all blocks that were in the old storage allocation table (SATT.SYS), but not in the new one. This is done as a safety precaution, in case these blocks belonged to files with a protection code of <128> (which are always zeroed when they are deleted).

A MOUNT operation for disk automatically checks to see if the disk is dirty. If the disk is dirty, and you are privileged, a rebuild occurs automatically, regardless of whether you specify /REBUILD. The following message is displayed to indicate that the disk is being rebuilt:

```
DISK is being rebuilt - wait ...
```

However, a nonprivileged user who tries to mount a dirty disk gets the error message:

```
?DISK needs rebuilding but you are not privileged
```

The /REBUILD qualifier forces a rebuild on a disk. It allows the privileged user to rebuild a disk that the system has not identified as "dirty." For example, some programs leave temporary files in the user's account, rather than deleting them before the program exits. Such a disk is not flagged as "dirty." If you want to get rid of the temporary files on a disk, you can mount the disk using the /REBUILD qualifier.

If you specify /REBUILD and the drive is write-protected, you get the error:

```
?Can't rebuild disk because device is write protected
```

A nonprivileged user who specifies /REBUILD gets the error:

```
?You must be privileged to rebuild the disk
```

The /NOREBUILD qualifier overrides an automatic rebuild of a dirty disk. You may not want to take the time to rebuild the dirty disk. Or, the disk may have doubly allocated blocks, and you do not want to risk deleting them. In this case, you can mount the disk with the /NOREBUILD qualifier (thus granting only read-only access) and try to fix the situation. For example, you could use the COPY command to copy the files to new locations (where they do not contain doubly-allocated blocks) and try to sort out the situation from there.

When a privileged user specifies /NOREBUILD to mount a dirty disk, the disk is mounted locked and read-only, and two warning messages are displayed:

```
%DISK needs rebuilding
%DISK is locked and mounted private, read-only
```

The disk is thus accessible only to privileged users.

The specification of the /NOREBUILD qualifier by a nonprivileged user to mount a "clean" (rebuilt) disk has no effect, and the mount succeeds.

11.5 MOUNT for Tapes

You issue a MOUNT command after you have physically mounted a tape on a tape unit and turned it online. MOUNT verifies that the tape is properly loaded onto the unit and checks an ANSI tape for the label specified in the MOUNT command. (ANSI tapes must have labels.)

Format

MOUNT device-name[:] [label]

Command Qualifiers	Defaults
/DENSITY = nnn	See description.
/FORMAT = argument	See description.
/[NO]WRITE	/WRITE

Prompts

Device: device-name[:]

Label: label

Command Parameters

device-name[:]

Specifies the physical or logical name of the drive on which the tape is physically mounted.

[label]

This parameter is necessary only for ANSI tapes. If you try to mount an ANSI tape without specifying a label, you are prompted for a label.

A label is not required (and is ignored) when you mount a tape in DOS or foreign format.

The label specifies the one to six character alphanumeric label written to the tape when it was initialized (see the INITIALIZE command description in Section 11.3). MOUNT then verifies that the label on the tape is the same as the one you specified in the MOUNT command. This process helps make certain that you have mounted the right tape.

Command Qualifiers

/DENSITY = nnn

Specifies the density in bits per inch (bpi) at which the tape will be read or written. You can specify 800 or 1600 if the tape drive supports it. Note that some kinds of drives support only one type of density. For example, the TS11 supports only 1600 bpi.

The density defaults to that which is currently on the tape. If this density is not supported by the tape drive, you get the error message:

```
?Data error or incorrect density
```

If you specify a density other than the one with which the tape was initialized, you get the error:

```
?Incorrect density or uninitialized tape
```

/FORMAT=ANSI
/FORMAT=DOS
/FORMAT=FOREIGN

Indicates whether the tape is in a standard format used by the RSTS/E operating system. The default format is determined at system generation.

The *RSTS/E DCL User's Guide* describes ANSI and DOS formats. A tape is FOREIGN if it is not in ANSI or DOS format.

NOTE

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.

/FORMAT=FOREIGN suppresses the message "?Bad directory for device" if you mount the tape at the wrong density.

/WRITE
/NOWRITE

Checks to see whether the tape is read/write (the write ring is in the tape) or read-only (the write ring is not in the tape).

The /WRITE qualifier checks to see if the tape can be written to as well as read from, if the write ring is in the tape. The /NOWRITE qualifier checks to see if the tape can only be read from, if the write ring is out of the tape.

If the write ring is not present and you specify /WRITE, you get the error message:

```
?Device is write protected
```

11.6 DISMOUNT for Disks

Releases a disk previously accessed with a MOUNT command. You issue this command before you take the drive off line, or before you physically dismount the disk.

Only a privileged user can logically dismount a public disk. Nonprivileged users can dismount private disks if they specify the pack-id label.

The DISMOUNT command deallocates the disk if it was allocated to you. You cannot DISMOUNT a device if there are open files on it. If you try, the system displays the error message:

```
?Account or device in use
```

Format

DISMOUNT device-name[:] [pack-id]

Command Qualifier	Default
/PUBLIC	None.

Prompts

Device: device-name[:]

Pack-id: pack-id

Command Parameters

device-name[:]

Specifies the disk unit containing the pack to be dismounted. If you do not specify a unit number, the following error message is displayed:

```
?Unit number needed
```

pack-id

The pack-id specified when the disk was initialized and mounted. The pack-id is deassigned as a system-wide logical. Nonprivileged users *must* specify the pack-id when dismounting a private disk.

The pack-id is optional for privileged users.

Command Qualifiers

/PUBLIC

Only a privileged user can dismount a public disk. (The disk must have been mounted and initialized as public.)

If you do not specify /PUBLIC, the system assumes that the disk was mounted as private. If you try to dismount a public disk without the /PUBLIC qualifier, you get the error message:

```
?DisK is mounted Public
```

If you try to dismount a disk that was mounted as private by using the /PUBLIC qualifier, you get the error message:

```
?DisK is mounted Private
```

Nonprivileged users can dismount private disks, but only if they specify the pack-id. A nonprivileged user who tries to dismount a public disk gets the error message:

```
?You must be Privileged to dismount a Public disK
```

If the disk was mounted with the /NOSHARE qualifier, and any job except the one that mounted the disk attempts to dismount it, the following error message is displayed:

```
?Account or device in use
```

11.7 DISMOUNT for Tapes

Releases a 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. The DISMOUNT command deallocates the tape drive if it was allocated to you. You cannot DISMOUNT a tape if there are open files on it. If you try, the system displays the message:

```
?Account or device in use
```

Format

DISMOUNT device-name[:] [label]

Command Qualifiers	**Defaults**
/[NO]UNLOAD	/UNLOAD

Prompts

Device: device-name[:]

Command Parameters

device-name[:]

Specifies the unit containing the tape to be dismounted. If you do not specify a unit number, the default is unit 0.

label

Can be used to specify the label on an ANSI tape. It is not necessary to specify a label with the DISMOUNT command. However, if you are using a hardcopy terminal, you can specify a label to keep a record of the tape being dismounted. The label you specify is ignored; it is not checked against the label on the tape.

Command Qualifiers

/UNLOAD
/NOUNLOAD

Specifies whether to unload the tape from the drive. The default is /UNLOAD, which unloads the tape.

Chapter 12
DCL Commands for Using the Micro-RSTS Spooler

This chapter provides background information about the micro-RSTS ("small") spooler. The chapter then describes the DCL (DIGITAL Command Language) commands for using the small spooler. Only a privileged user can use these commands, each of which is summarized in Table 12–1.

Table 12–1: Commands for Using the Small Spooler

Command	Description
START/QUEUE/MANAGER	Starts up the small spooling package. (See page 12-7.)
STOP/QUEUE/MANAGER	Shuts down the small spooling package, either immediately or on completion of any currently active jobs, depending on the qualifiers you select. (See page 12-9.)
INITIALIZE/PRINTER	Identifies a line printer or terminal to be used for printing. (See page 12-10.)
DELETE/PRINTER	Removes a line printer or terminal from the list of defined print devices. (See page 12-12.)
STOP/PRINTER	Stops printing on a device. Depending on the qualifier you select, printing stops immediately, at the end of the current file listing, or on completion of the current job. (See page 12-13.)
START/PRINTER	Restarts printing on a device, after a STOP/PRINTER command. Depending on the qualifier you select, printing resumes at the point where it was stopped, the start of the current file, the start of the current job, or a specific page in the current file listing. (See page 12-14.)
Although the /PRINTER qualifier in the above commands is the preferred syntax, you can instead use the /DEVICE qualifier to mean the same thing. The /DEVICE qualifier is permitted for compatibility with RSX syntax.	

12.1 Managing the Small Spooler — SPL

The small spooling package, also called SPL, provides an alternative to the standard ("large") RSTS/E spooler, which is described in Chapter 5. SPL is the only spooler available on micro RSTS.

SPL is a system utility designed to print files queued for output on a specified line printer or hardcopy terminal. Because print devices can be accessed by only one job at a time, print spooling reduces the "bottleneck" of many users trying to gain access to a print device at the same time. With SPL, users can issue print requests that are queued (maintained in a "waiting list"), and processed when the request reaches the top of the queue. Print requests are generally processed on a first-come, first-serve basis. However, options are available to let you move jobs up in the queue, or to wait for a specified date and time to elapse before the job is printed.

When you start up the spooling package, you initialize the devices — line printers or terminals — to be used for printing. (You can define a maximum of four devices for print spooling.)

Additionally, you assign a *form name* to each print device, describing the attributes of the paper currently installed on that printer. For example, the print device can be loaded with paper for blank checks, and print jobs can supply data to be printed on the check forms.

Print requests can be processed only on devices whose assigned form name matches the job's specified form. If the job's form name does not match that of the device, the job remains in the queue and is not printed until you assign that form name to a device.

After printing starts, the system manager can issue commands to:

- Stop printing on a device (to correct a paper jam, replenish paper, or change forms)
- Continue printing on the same or another device
- Shut down the spooling package
- Change the form name for a device

This chapter describes all the commands used to control operation of the small spooling package. Note that only privileged jobs (those running in a [1,*] account) can issue these commands.

You can install both spooling packages and operate them concurrently on the same RSTS/E system. See Section 12.2 for details on how to install and use both packages on the same system.

SPL offers several advantages over the large spooling package:

1. Reduces system load. SPL's QUEUE program requires only one job on the system, versus a minimum of three for comparable services from the large spooler's OPSER/SPOOL program.
2. Executes commands faster. The package is optimized for better performance.
3. Lets you print files with any RMS file structure.

Note that SPL also has some limitations. While the package supports concurrent printing on up to four print devices, the performance when driving four high-speed print devices from within SPL's single job is not as fast as from the large spooler. In addition, SPL does not include batch processing services, only print spooling.

For information about general user commands for the spooler, such as PRINT, SHOW QUEUE, and DELETE/JOB, refer to the *RSTS/E DCL User's Guide.*

12.2 Providing Two Spoolers on One RSTS/E System

A RSTS/E system can support both the large and small spooling packages concurrently. Both spoolers are necessary for sites that want to use the small spooler for printing, but must continue to use the large spooler for batch processing.

Please note the following when installing both packages:

- Two versions of the DCL keyboard monitor are provided. One version recognizes commands for the small spooler; the other recognizes commands for the large spooler. For example, the DCL that supports the small spooler does not recognize the SUBMIT command, while the DCL that supports the large spooler does not recognize the START/PRINTER command. Choose the version of DCL that offers support for the package used more frequently.

- You can access the small spooling package by running its QUEUE.TSK interface program. When you type RUN $QUEUE, QUEUE announces itself and prompts for a command. You can enter both user and operator commands this way.

- The small spooling package can be accessed by a standard CCL. For example, you can define the CCL "SPL" as follows:

```
$ UT CCL SPL-=$QUEUE.TSK;PRIV 30000 (RET)
```

  After the CCL is defined, users or operators can issue standard commands by prefixing the command with the "SPL" CCL, as in:

```
$ SPL PRINT $NOTICE.TXT/COPIES=2 WILSON (RET)
Job #10 entered on queue PRINT
```

- The UU.SPL monitor directive will route print requests to the small spooling package if its user receiver ID (QMAN) is defined in the system receiver table. This means that any print requests issued will automatically be routed to the small spooler for those sites that have the package started. No code changes are required for applications that use the UU.SPL directive. See the *RSTS/E System Directives Manual* or the *RSTS/E Programming Manual* for further details on the UU.SPL directive.

- Note that all queues associated with each package are independent of one another. Jobs placed in one queue do not affect jobs in the other queue.
- If you choose to use both packages for printing on the same device, the small spooler can "monopolize" the device, due to the improved performance of the small spooler over the large package.

12.3 Managing Forms for the Small Spooler — The Forms Definition File

The small spooling package provides a Forms Definition File (FDF) to define forms used with the package. The FDF is most useful for sites that use multiple forms as part of their daily printing requirements. The FDF provides a central location for maintaining all form definitions, and eliminates the need for a manager to respecify a form's attributes whenever a new form is installed. "Attributes" in this case means the characteristics described later in this section, such as the length and width of the form.

The name of the Forms Definition File supplied on the RSTS/E distribution kit is [1,2]FORMS.SYS. As distributed, this file contains the attributes for the default form NORMAL. Any form name specified in the commands INITIALIZE/PRINTER, START/PRINTER, and PRINT (including the default form NORMAL) must be defined in the FDF.

Because this file is an ASCII stream file, you can edit it using any standard text editor (such as EDT) to add new form definitions, delete existing form definitions, or change the attributes of any currently defined forms.

This section describes the contents of the Forms Definition File and how you change the file for forms at your site. The copy of [1,2]FORMS.SYS provided on the distribution kit is:

```
!         FORMS.SYS - Micro RSTS V8.0 Forms Definition File
!
NORMAL    LENGTH=66    WIDTH=132    JOB_PAGES=2    FLAG_PAGES=2    SIMULATE
```

As shown in the first two lines of the file, you can include comment lines by using an exclamation mark (!) as the first character of the line. Comment lines can make the file easier to read.

The general format of a form definition within the FDF is a single line of text, followed by any standard line delimiter:

form-name [attribute–1 attribute–2 ..... attribute-n]

Attributes are separated from other parts of the definition by one or more spaces or tabs. (Note that they do not use slashes; you cannot access these attributes from DCL. Attributes are intended for the FDF only.)

The first field in the definition must be the form name; form attributes can follow in any order. The fields that make up a form definition are:

form-name
: The form-name can consist of 1–6 alphanumeric characters. You can specify form names using either uppercase or lowercase characters. However, the name is always forced to uppercase for display or matching purposes.

attribute-n
: You can specify attributes in any order and can abbreviate them to their first two characters. If you do not specify an attribute in the FDF, its default value is assumed.

The following are the attributes available for defining forms.

LENGTH = n
: Indicates the number of lines per printed page for the specified form. The value n can range from 1 to 255. The default length is 66; this is standard spacing for an 8–1/2 by 11 inch form with vertical spacing of 6 lines per inch.

WIDTH = n
: Indicates the maximum number of characters that can appear in a printed line for the specified form. The value n can range from 1 to 255. Typical form widths are 132 for standard printer listings and 80 for terminal listings. The WIDTH value determines where a printed line will be truncated (if the file is being printed with the /TRUNCATE qualifier of the DCL PRINT command in effect) or wrapped around to a new line (if the file is being printed with the /NOTRUNCATE qualifier in effect).

The default width is 132.

[NO]SIMULATE
: Determines whether the spooling package should, on encountering a form-feed character, send the character directly to the device (NOSIMULATE), or simulate the form-feed by transmitting the proper number of line-feed characters (SIMULATE). Specifically:

  - Most high-speed printers can interpret a form-feed character (ASCII 10) as an instruction to move the paper to the start of the next page, or to the next "top-of-form." In such cases, you should specify /NOSIMULATE because the device can handle form-feeds.
  - If the device cannot interpret form-feeds as top-of-form characters, the spooler can send an equivalent number of line-feed characters to simulate a form-feed. For these devices, specify /SIMULATE.

Use SIMULATE for devices that either do not recognize form-feed characters or cannot use them to properly position paper at the top of page, based on the form's defined length. The SIMULATE attribute causes SPL to translate any form-feed characters into the correct number of line-feed characters to position the paper at the next top of page.

Use NOSIMULATE for devices that can process form-feed characters directly to position the paper at the next top of page, based on the form's defined length. Note that some devices can process form-feed characters for "standard" length forms (that is, standard for that device), but cannot do so for non-standard form lengths.

The default is SIMULATE.

The next two qualifiers both work the same way. The only difference between them is that /JOB_PAGES=n separates print jobs, while /FLAG_PAGES=n separates files within a print job.

In both cases, you might want to specify 0 (zero) for n if you have a slow print device. This lets you save time by not printing extra pages.

JOB_PAGES=n

Defines the number of job header pages to be printed at the beginning of each job listing. The value n can range from 0 to 127. Job header pages display information about the job (job name, project-programmer number, and so forth), printed in large block letters. Job header pages make it easy to separate job listings.

For special jobs like checks, where you do *not* want job header pages, specify JOB_PAGES=0.

The default is JOB_PAGES=1.

FLAG_PAGES=n

Defines the number of file header pages ("flag" pages) to be printed at the beginning of each file listing. The value of n can range from 0 to 127. File header pages display information about the file (file name, project-programmer number, and so forth), printed in large block letters. File header pages make it easy to separate printouts of files within a job. Note that a user can suppress the printing of file header pages by including the /NOFLAG_PAGES qualifier with the file to be printed.

For special jobs like checks, where you do *not* want file header pages, specify FLAG_PAGES=0.

The default is FLAG_PAGES=1.

12.4 START/QUEUE/MANAGER

This command starts the small spooling package by creating the spooler job and then initializing its print queue.

Format

START/QUEUE/MANAGER

This command has no qualifiers.

The START/QUEUE/MANAGER command creates a single detached job that contains the spooling package. Only the SHOW QUEUE command can be used when the package is not running. You get an error message if you issue the START/QUEUE/MANAGER command and the package is already started.

When the package is first started, the queue manager module "compresses" the queue file. This means that it removes any logically deleted records — a process that can take from a few seconds to several minutes, depending on the size of the queue. Any jobs still waiting for processing in the queue retain their status from the last spooling session. Next, the queue manager module waits for confirmation from each print spooler module, indicating that it is online. After all spooler modules are online, the queue manager returns confirmation that the spooling package is now fully operational.

Note that printing does not begin until you issue commands to initialize any devices to be used for printing and identify the forms installed on those devices. Devices initialized for printing are not retained across spooling sessions; you must reinitialize print devices and reassign forms each time you restart the package.

Because the spooling package contains several modules within a single job, the name associated with this job (which appears as part of a SYSTAT display) changes dynamically according to the module currently running. The job names that can appear in a SYSTAT display, and their associated status are:

Job name	Status
SPL.QM	Queue manager module running
SPL.S0	Spooler 0 module running
SPL.S1	Spooler 1 module running
SPL.S2	Spooler 2 module running
SPL.S3	Spooler 3 module running
SPL.XM	Send/receive module running
SPL...	Spooling package currently idle

In addition, several receiver names are used in the package. The spooling package uses the RSTS/E "multiple RIB" (Receiver ID Block) feature to permit several modules to communicate with one another by local send/receive within the same job. (Local send/receive is described in the *RSTS/E Programming Manual.*) The following receiver names are reserved by the spooling package, and appear in the "Message Receivers" table whenever the package is running:

Receiver	Usage
QMAN	User/operator commands to queue manager
QSPL	Spooler messages to queue manager
SPL0	Queue manager messages to spooler 0
SPL1	Queue manager messages to spooler 1
SPL2	Queue manager messages to spooler 2
SPL3	Queue manager messages to spooler 3

12.5 STOP/QUEUE/MANAGER

This command shuts down the small spooling package, either immediately (causing all currently active jobs to be aborted) or after all currently active jobs are completed. (In neither case, however, does it delete print requests of jobs that are waiting in the queue.)

Format	
STOP/QUEUE/MANAGER	
Command Qualifier	**Default**
/[NO]ABORT	/NOABORT

The STOP/QUEUE/MANAGER command causes the queue manager module to shut down all other modules in the package. After all other modules are confirmed as having gone offline, the queue manager itself terminates, causing the single spooling package job itself to be killed.

Qualifiers

/[NO]ABORT

Indicates whether any jobs currently running should be aborted before shutting down the spooling package. Use /ABORT to shut down the package immediately; use /NOABORT to allow any currently running jobs to complete before proceeding with the package shutdown. (If you later issue the START/QUEUE/MANAGER command, printing resumes with the next job in the queue.)

The default is /NOABORT.

12.6 INITIALIZE/PRINTER

This command defines a line printer or terminal to be used for printing.

Format

INITIALIZE/PRINTER device[:]

Command Qualifiers	Defaults
/FORMS[= form-name]	See discussion.
/[NO]SHAREABLE	/SHAREABLE

Prompts

Device: device-name[:]

Command Parameters

device[:]

Identifies a line printer (LPn:) or terminal (KBn: or TTn:) to be used for printing. Logical device names are permitted, but they must refer to a physical line printer or terminal. You get an error message if you specify a device other than a line printer or terminal. If you do not provide a unit number:

- Zero (0) is assumed for a line printer
- Your current keyboard number is assumed for a terminal

You can initialize up to four print devices for use by the spooling package. You can change devices by issuing a DELETE/PRINTER command (described in the next section), followed by another INITIALIZE/PRINTER command that specifies the new device you want to use for printing.

Command Qualifiers

/FORMS[= form-name]

Identifies the form currently installed on this device. Only jobs submitted with this form-name can be printed on this device.

Any form-name you specify (as well as the default form NORMAL) is checked against forms defined in the Forms Definition File (see Section 12.4). The system displays an error message if the form-name does not exist or if it contains an invalid attribute.

If you do not specify the /FORMS qualifier, then the device is initialized without a form, preventing any jobs from being started on that device. You can later issue a START/PRINTER command to assign a form to the device. If you specify /FORMS without an argument, then the default form-name NORMAL is assigned.

/[NO]SHAREABLE

Specifies whether the initialized device can be shared with other jobs on the system. The default is /SHAREABLE.

If you specify /NOSHAREABLE, the device is allocated to the spooler's job, thereby preventing other jobs on the system from gaining access to the device. This state is useful, for example, when the device specified is a terminal and you want to prevent anyone from logging in on the device while it is under the control of the spooling package. The device remains allocated to the spooler's job until it is deleted from the spooler's device tables (with the DELETE/PRINTER command), or the spooling package is shut down (with the STOP/QUEUE/MANAGER command). You get an error if you specify /NOSHAREABLE for a device currently allocated to another job on the system.

If you specify /SHAREABLE, the device is allocated to the spooling job only while it is needed for printing. The device is available to other jobs when the spooler is not using it. If the spooler finds a device unavailable when starting a job, it simply waits until the device becomes available and then proceeds.

12.7 DELETE/PRINTER

This command removes a printer from the list of print devices defined for the spooling package.

Format

DELETE/PRINTER device[:]

Prompts

Device: device-name[:]

This command has no qualifiers.

Command Parameters

device[:]

Identifies the line printer (LPn:) or terminal (KBn: or TTn:) to be deleted from the list of defined printers. The device specified must have been defined by an INITIALIZE/PRINTER command. You can delete only idle print devices. An error is displayed if the device to be deleted is currently processing a job.

If the device was initialized with the /NOSHAREABLE qualifier, it is deallocated from the spooler's job, making it available to other jobs on the system.

12.8 STOP/PRINTER

This command stops printing on a specified device.

Format

STOP/PRINTER device[:]

Command Qualifiers	Defaults
/JOB_END	See discussion.
/FILE_END	See discussion.

Prompts

Device: device-name[:]

Command Parameters

device[:]
Identifies the device at which to stop printing. The device you specify must have been defined by an INITIALIZE/PRINTER command.

If you do not specify a qualifier, then printing on the device stops immediately (as soon as the device's print buffers are empty). You can use qualifiers to stop printing at the end of the current file copy, or on completion of the job. After printing is stopped, the device will not resume printing until you issue a START/PRINTER command.

The STOP/PRINTER command is useful for changing forms between jobs, replenishing the paper supply, or correcting a paper jam.

Command Qualifiers

You can use the following qualifiers to stop printing at a defined point in the current job. These qualifiers are ignored for an idle printer. However, the device is still marked as stopped, and no additional jobs will be started on the device until you issue a START/PRINTER command.

You can specify only one of these qualifiers in a STOP/PRINTER command; you get an error message if you use both of them.

/JOB_END
Indicates that printing should stop after the device has finished printing the current job. Use this qualifier to change forms between jobs. The START/PRINTER command lets you assign a new form to the device.

/FILE_END
Indicates that printing should stop after the current file copy is finished printing. This qualifier is useful for replenishing the paper supply while printing a job that consists of several files.

12.9 START/PRINTER

This command restarts printing on a stopped device.

Format

START/PRINTER device[:]

Command Qualifiers	Defaults
/FORMS[= form-name]	See discussion.
/NEXT_JOB	See discussion.
/RESTART	See discussion.
/TOP_OF_FILE	See discussion.
/BACKSPACE[= n]	See discussion.
/FORWARDSPACE[= n]	See discussion.
/PAGE = n	See discussion.

Prompts

Device: device-name[:]

Command Parameters

device[:]
Identifies the device where printing is to resume. This device must have been defined by an INITIALIZE/PRINTER command.

Forms Qualifier

/FORMS[= form-name]
Specifies the form now installed on this device. Only jobs submitted with the same form name can be printed on this device.

Any form-name you specify (as well as the default form NORMAL) is checked against forms defined in the Forms Definition File (see Section 12.4). An error is displayed if the form name specified does not exist or contains an invalid attribute.

Any form attributes previously assigned to the device are replaced by the new form's attributes. Note, however, that form changes do not take effect until the start of a new job; if you issue a START/PRINTER command and include a new form while a job is in progress, the job is completed using the previous form attributes. Use the /JOB_END qualifier with the STOP/PRINTER command to allow any current job to complete before changing forms.

If you do not specify the /FORMS qualifier, then printing resumes with the currently defined form. If you specify /FORMS without an argument, then the default form NORMAL is assigned.

Job Restart Qualifiers

The following qualifiers indicate where processing should resume for the currently active job. SPL ignores a job restart qualifier if the device specified is currently idle. Note that you can specify only one of these qualifiers in a START/PRINTER command. An error message is displayed if you include more than one restart qualifier in the same command.

/NEXT_JOB
Indicates that the current job should be aborted and printing should resume at the next job.

/RESTART
Indicates that printing should resume at the beginning of the current job copy. If the job was submitted with a /JOB_COUNT qualifier argument larger than one, then only the current job copy is restarted.

/TOP_OF_FILE
Indicates that printing should resume at the beginning of the current file copy. If the file being printed was submitted with a /COPIES qualifier argument larger than one, then only the current file copy is restarted.

/BACKSPACE[=n]
Indicates that printing should resume n pages back in the current file listing. If you do not include an argument, then printing resumes one page before the current page. If the argument specified extends beyond the beginning of the file, then printing resumes at the beginning of the current file copy.

/FORWARDSPACE[=n]
Indicates that printing should resume n pages forward in the current file listing. If you do not include an argument, then printing resumes one page after the current page. If the argument specified extends beyond the end of the current file, then printing resumes at the next file copy.

/PAGE=n
Indicates that printing should resume at page n in the current file listing. If the argument specified extends beyond the end of the file, then printing resumes at the next file copy.

Appendix A
Auxiliary System Program Files

Certain auxiliary files are built during the system library build procedures. The program that builds each file is stored in the library along with the file. If, for any reason, the file is damaged or destroyed, the file can be created by running the related program, as described in this appendix.

A.1 Character Generation File — CHARS.QUE

The line printer spooling program SPOOL requires the character generation file CHARS.QUE. The file is a virtual core array and is stored on the system disk during system generation by commands in the SPLER.CTL file. To create the CHARS.QUE file, first make sure that the old copy is deleted from the system library. The CHARS program terminates with an error if a file named CHARS.QUE exists in the system library directory. Next, run the CHARS program by typing the following command:

```
$ RUN $CHARS
CHARS  V8  RSTS  V8  TIMESHARING
```

After terminating, CHARS returns control to your keyboard monitor.

A.2 Batch Command Decoding File — BATCH.DCD

The BATCH system program requires the command decoding file BATCH.DCD. The file is a virtual array and is created during system build by commands in the SPLER.CTL file. To create the BATCH.DCD file, first make sure that the old copy is deleted from the system library. Run the BATDCD program while logged into the system under a privileged account. The program terminates with an error if a file named BATCH.DCD exists in the system library directory. The following sample dialogue shows the proper procedure:

```
$ RUN $BATDCD
BATDCD  V8  RSTS  V8  TIMESHARING
```

After terminating, BATDCD returns control to your keyboard monitor.

A.3 Backup Prompt File — BACKUP.PRM

The BACKUP system package requires the file BACKUP.PRM to store prompting text for commands and special parameters for creating the work file. The file is created in the system library during system generation by commands in the BACKUP.CTL file. To create the BACKUP.PRM file, run the BACPRM.BAC program while logged in to the system under a privileged account. The following sample dialogue shows the procedure:

```
$  RUN $BACPRM
BACPRM V8 RSTS V8 TIMESHARING

$
```

After terminating, BACPRM returns control to your keyboard monitor.

A.4 Error Package Data File — ERRDAT.FIL

The system error package requires the ERRDAT.FIL file to store the data used in the error package for processing errors. The file is created in the Error Package Library during system build by commands in the BUILD.CTL file. To create the ERRDAT.FIL file, run ERRBLD.BAC program while logged in to the system under a privileged account. The following sample dialogue shows the procedure:

```
$  RUN $ERRBLD
ERRBLD V8 RSTS V8 TIMESHARING

$
```

After terminating, ERRBLD returns control to your keyboard monitor.

Appendix B
Number Conversion

Many applications require a number based on bit values in a PDP–11 word. The following list shows the octal and decimal values for each bit in the PDP–11 word.

Bit Octal Number	Octal Value	Decimal Value
0	1	1
1	2	2
2	4	4
3	10	8
4	20	16
5	40	32
6	100	64
7	200	128
8	400	256
9	1000	512
10	2000	1024
11	4000	2048
12	10000	4096
13	20000	8192
14	40000	16384
15	100000	32768 (32767 + 1)

Appendix C
Manual Patching with ONLPAT and CPATCH

With the advent of prebuilt tasks and other replacement modules, the need for manual patching should steadily decrease. The trend will be to supply RSTS/E users with whole modules of replacement code, rather than requiring you to manually patch in changes.

However, some users may need to use ONLPAT or CPATCH for the following reasons:

1. Patches to an already built monitor still require ONLPAT.
2. Feature patches to BASIC–PLUS software still require CPATCH.
3. Certain layered products are released at different times from RSTS/E, so patches to them may not be included on regular RSTS/E update kits. If so, you may have to type in the patches or take them from other distribution media.
4. If you are building your own update kits, DIGITAL provides you the tools to create and install the kits manually.

The ONLPAT and CPATCH programs are referenced in Chapter 10, but the full details are provided here if you need to patch binary or source code manually.

C.1 Patching RSTS/E Binary Code — ONLPAT

The ONLPAT.SAV program allows you to patch binary code. It can run in keyboard mode to install individual patches from a terminal or can run from a command file.

C.1.1 Using ONLPAT in Keyboard and Command File Mode

ONLPAT gives you a choice. You can make patches to the software by entering the corrections manually at your keyboard or you can use a previously-created command file to apply the patches for you. You make the choice at the beginning of the ONLPAT dialogue when the program asks for the name of the command file you want to use (if one exists). Your response determines how ONLPAT gets the patch information. If you enter a command file name, ONLPAT uses it to make the corrections. When you press the RETURN key only, ONLPAT assumes the patch information will come from the keyboard. It expects keyboard input as well when you enter a file name followed by an equal sign (=). This indicates you want ONLPAT to create a log file of the patch. The remainder of this section describes how to patch software using either of these methods.

When entering patches at the keyboard, you follow procedures similar to those you use for the PATCH option of INIT.SYS. Unlike the PATCH option, ONLPAT does not install a patch until you type an up-arrow/C (^C) at the end of the last line in the patch. (Do not type CTRL/C unless you want to abort the patching operation.) Having the program wait until you have completely entered the patch information allows you to review the entire patch and make changes before the patch is installed.

The ONLPAT program can apply patches to the monitor or to programs written in MACRO, such as PIP.SAV. To install a published patch, you first look in the *RSTS/E Maintenance Notebook* or the *RSTS/E Software Dispatch* for the article containing the patch to install. Once you have read the article thoroughly, you can begin to create a command file or manually install the patch.

An ONLPAT example follows:

```
 RUN $ONLPAT
Command file name? RET
File to patch?  RSTS.SIL RET
Module name?  RSTS RET
Base address?  ..CAGE RET
Offset address?  0 RET
 Base    Offset  Old      New?
??????  000000  000074  ?  3. RET
??????  000002  ??????  ?  ^Z
Offset address?  ^Z
Base address?  $$0301 RET
Offset address?  0 RET
 Base    Offset  Old      New?
??????  000000  000000  ?  Q!4 RET
??????  000002  000000  ?  ^C
```

Table C–1 lists and describes the types of responses ONLPAT accepts in response to the program dialogue. The questions asked by ONLPAT are described in Table C–2.

Table C–1: Responses to ONLPAT Questions

Response	Meaning
Number	Enter an octal number from 0 and 177777; leading zeros are optional. Or, enter a decimal number in the range 0. to 65535., making sure to include a trailing decimal point to distinguish it from an octal number. The octal or decimal number you enter becomes the new contents of the current location.
Symbol	Substitute a global symbol name for an octal number as the new contents of the current location. The symbol table for the module being patched is part of the SIL file. It contains that module's global symbol names and their values. A global symbol name must be one to six alphanumeric characters and must be defined in the symbol table for the current module.
Expression	Use an expression as a substitute for an octal number. An expression consists of one or more numbers or global symbols, separated by arithmetic operators (+,-, and *). You can use parentheses to group portions of an expression.
LINE FEED key	Advance to the next location without altering the contents of the current location.
circumflex	Return to the previous location without altering the contents of the current location.
CTRL/Z	Type CTRL/Z to return to the previous question.
CTRL/C	Type CTRL/C to abort all patching and return to the program prompt.

Use the information in Table C–2 and the following procedures to apply patches manually:

1. Gather the articles that contain the patches you need. They are found either in the *RSTS/E Maintenance Notebook* or the *RSTS/E Software Dispatch*.

2. Run ONLPAT by typing RUN $ONLPAT. The program begins by printing the first of a set of dialogue questions:

   ```
   Command file name?
   ```

   At this point you press the RETURN key to enter the patch manually at your terminal, or you can type the name of a file, followed by an equal sign, to have ONLPAT create a log file of the patch to be created (as well as go to the terminal). You then answer the remaining ONLPAT dialogue questions following instructions published in the published article. Remember that ONLPAT does not install the patch until you type the two characters up-arrow and C (not CTRL/C) to end the program. Typing up-arrow/C is entirely different than typing CTRL/C. One ends the patch successfully (up-arrow/C) and the other (CTRL/C) stops execution of ONLPAT without making any patches.

Table C–2: ONLPAT Dialogue Questions

Questions and Responses
COMMAND FILE NAME? 1. Enter a command file name to have ONLPAT install patches using commands in the command file. You can have ONLPAT make a record of the patches as well as install them if you include a log file name with the command file name in the command line: Command file name? log file = command file name If you do not include a file type for the command file, ONLPAT selects .CMD. The default file type for the log file is .LOG. 2. Press the RETURN key if you do not have a command file to apply the patches. ONLPAT assumes you want to enter the patches manually from the keyboard. If you want a record of the patches that you apply manually from the keyboard, include only the log file name in the command line: Command file name? log file = You must place an equal sign after the log file name to make sure ONLPAT does not interpret the file you specify as being a command file. Again, ONLPAT uses .LOG as the file type if you do not specify one with the file name.
FILE TO PATCH? Press the LINE FEED key if you are applying a patch to the installed monitor SIL. Another response may be necessary however. If so, enter the response suggested either by the *RSTS/E Maintenance Notebook* or the *RSTS/E Software Dispatch* article.
MODULE NAME? Enter the module name included with the patch described either in the *RSTS/E Maintenance Notebook* or the *RSTS/E Software Dispatch* article.
BASE ADDRESS? Type the base address included with the patch described either in the *RSTS/E Maintenance Notebook* or in the *RSTS/E Software Dispatch* article.
OFFSET ADDRESS? Type the offset address included with the patch described either in the *RSTS/E Release Notes* article or the *RSTS/E Software Dispatch* article. The offset address is the first location to be patched relative to the specified base. The response suggested in the patch article will be a number, symbol, or expression as described in Table C–1.

ONLPAT prints a message telling you when it has installed the patch:

```
PATCH COMPLETE
1 PATCH INSTALLED
```

ONLPAT then returns to the command file question, allowing you to exit the program or enter another patch.

To install a patch with a command file:

1. Create a command file as described in Section C.2.
2. Run the ONLPAT program by typing RUN $ONLPAT, and press the RETURN key. The program prints a question asking for the name of the command file that contains the patches. Enter a command file in the following format:

```
logfile=command file
```

   On the input side of the command line, type the file name of the file that contains the commands ONLPAT will use to install the patches. If you want a record of the patch(es) being applied, type the name of a log file. The default file type for log file is .LOG and is .CMD for the command file. If you do not include a specification for the log file, ONLPAT does not produce a log file. ONLPAT executes the command file and if you did not include the name of the file to be patched in the command file, ONLPAT pauses, allowing you to enter the file name. When ONLPAT finishes processing, it prints a summary of the patches installed (and skipped, if any), and then returns to your keyboard monitor.

C.1.2 Patching a Running Monitor with ONLPAT

Normally you can patch the installed monitor Save Image Library (SIL). The monitor is modified on disk, and only after you shut down and then restart the system do the patches become effective. However, patches to monitor overlay code (OVR) are a special case. Because monitor overlays are read in from the monitor SIL (from disk) when they are needed and changes to the monitor do not take place until the system is brought down and then restarted, your system might crash or not work properly if you use patched code from OVR along with an unpatched monitor in memory.

ONLPAT can tell when the file being patched is the installed monitor SIL and in that case imposes certain restrictions:

1. Installs the patch if a patch does not involve OVR. That is, ONLPAT applies the patch to the image of the monitor on disk. (The newly patched monitor is brought in from disk and installed after the system has been shut down and then restarted under timesharing.)
2. Installs the patch if a patch involves only OVR and only one block of OVR is patched. Because the patch affects only one block, ONLPAT can make all monitor modifications with a single disk write.
3. Prints the following message and does not modify the monitor if the patch uses more than one overlay block:

```
PATCH TO MODULE OVR TOO LONG FOR INSTALLED SIL
```

4. Prints the following message and does not modify the monitor if the patch involves both OVR and some other module of the monitor:

```
PATCH TO INSTALLED SIL SPANS OVR AND OTHER MODULE(S)
```

   The only way to install a patch of this type is to shut down the system and either use the PATCH option of INIT, or install another monitor SIL (such as the SYSGEN.SIL) and patch the target SIL while it is not the running monitor.

C.2 Building ONLPAT Command Files

Create ONLPAT command files as follows:

1. Run ONLPAT from any logged in terminal by typing RUN $ONLPAT. The program then prints the COMMAND FILE NAME question.
2. Respond to the command file question in the following format:

```
Patchfile = KB:
```

   You enter on the output side of the command string, the patch file name you are creating and KB: on the input side to indicate you are creating the patch file from the keyboard.

3. Press the RETURN key after you finish entering the command string. ONLPAT then asks:

```
FILE TO PATCH?
```

   Type the name of the file you want to patch, press the RETURN key, and enter the text of the patch described in the *RSTS/E Maintenance Notebook* or the *RSTS/E Software Dispatch.*

After you create the command file, you may need to modify it. Apply the following rules if modifications are necessary:

- Individual command lines in a patch always end with a carriage return.
- When it executes the command file, ONLPAT forces all lowercase characters to uppercase and reduces all tabs and spaces to a single space.
- Blanks are significant. Use them exactly as shown for individual command lines.
- The first lines of an individual patch can be comment lines. A comment line begins with an exclamation mark (!) and can contain any other ASCII data.

- The next line in a patch is optional, but if present, you must enter it as follows:

```
OPTION: PATCH
```

- The next line specifies the file you want to patch and is required:

```
FILE TO PATCH? filename.type
```

  Enter the name of the file that ONLPAT is to patch. If you do not include a file name, ONLPAT prompts you during program execution for the name of the file. ONLPAT uses the file name to specify any subsequent patches in the command file that do not specify the file name.

- Subsequent command lines in a patch are identical to the formats used in the *RSTS/E Software Dispatch*. The following command line is optional, but when you use it must appear with spacing exactly as shown in the *RSTS/E Software Dispatch* articles:

```
BASE     OFFSET  OLD        NEW?
```

  Also, do not use any parenthetical comments.

- Notation must be exact for these constructs:

  ^Z is up-arrow, then Z (not CTRL/Z)

  ^C is up-arrow, then C (not CTRL/C)

  <lf> is <, then l, then f, then > (not LINE FEED key)

C.3 Patching ASCII Source Code — CPATCH

Use the CPATCH program to install patches to the source code of the BASIC–PLUS library programs. CPATCH can patch individual programs, as described in this section or build patching command files, as described in Section C.5.

The general procedures to patch a BASIC-PLUS source file are:

1. Run the program from a privileged account by typing RUN $CPATCH. After printing a header line, CPATCH prints:

```
FILE TO PATCH-
```

2. Respond in the format:

```
out-filespec=in-filespec
```

Enter the file name for the unpatched source file on the input side of the command line and any file specification you want on the output side. The default file type for both files is .BAK. If you do not include an output file, CPATCH assumes the file has the same name as the input file and gives it a .BAK file type.

3. Press the RETURN key in response to the pound sign prompt (#). The prompt printed by CPATCH indicates that it is ready to patch the source program.
4. Edit the source program, using the instructions in the *RSTS/E Maintenance Notebook* or the *RSTS/E Software Dispatch*. CPATCH prints an asterisk prompt (*) to show when to begin editing the source.
5. Type EXIT in response to the asterisk prompt when you finish editing. CPATCH prints a message indicating the patch is installed:

```
PATCH FROM file specification COMPLETE.
```

   CPATCH returns to the pound sign prompt. If there are other patches to the current file, do them now, or type CTRL/Z. If you type CTRL/Z, CPATCH returns to the prompt:

```
FILE TO PATCH-
```

6. Return to the second step in this procedure if there are other files to patch.
7. Type CTRL/Z to exit the program if there are no more files to patch.
8. Follow instructions in either the *RSTS/E Maintenance Notebook* or the *RSTS/E Software Dispatch* for including the patched source program in the system library.

C.4 Running the PBUILD Program

After you build the patching and PBUILD command files as described in Section C.5, run the PBUILD program to install the patches.

C.4.1 PBUILD Dialogue

Type RUN $PBUILD, and press the RETURN key to begin the PBUILD dialogue:

```
READ FILES TO PATCH FROM <SY:[1,2]>:
```

Enter the device and account that contains the unpatched versions of the text files. The device and project-programmer number (PPN) in angle brackets is the default value. If either the device or the PPN is incorrect, enter the correct values in the following format:

```
logfile=commandfile[-]
```

Logfile is the name of the file in which the entire patching procedure is recorded. Device represents the input device name and unit number and is followed by the project-programmer number (account number). The DETACH switch causes PBUILD to detach from the keyboard while performing the patching operation. If you specify both a log file and the DETACH switch, some patching operations are faster because they are not limited by the speed of the terminal.

For example, if the files are on the public structure under account [120,5], the correct response is [120,5]. If the files are on magnetic tape drive MM0: on account [1,2], the correct response is MM0:. If both the account and the device are wrong, you must enter both, such as MM0:[120,5]. If both are correct, then just enter a carriage return.

After you specify the input device, PBUILD prints:

```
COMPILE PATCHED PROGRAMS<YES>:
```

If you press the RETURN key, PBUILD compiles the resulting patched file and prints additional patch questions. Type NO and PBUILD does not ask the next three patch questions.

If you want PBUILD to compile the files, it asks where to store the compiled programs:

```
LIBRARY DEVICE <SY:[1,2]>:
SYSTEM DEVICE <SY0:[1,2]>:
```

The library device is the device and account that PBUILD uses for the programs that normally go to the library account. The system device refers to the disk that contains the RSTS/E system. The compiled programs normally reside in the library account.

If you want to have PBUILD save the patched sources, type YES in response to the following question:

```
SAVE PATCHED SOURCES<NO>:
```

PBUILD skips the next question and deletes the patched sources after compiling if you type NO or press the RETURN key.

If you choose to retain the patched sources, the program prints:

```
WRITE PATCHED SOURCES TO <SY:[200,200]>:
```

Your response indicates on which device and in which account you want the patched sources to reside. DIGITAL recommends that you not place the patched sources in account [1,2]. If you have no particular device or account for the sources, you can accept the default. This allows PBUILD to place the sources in account [200,200] on the public disk structure.

When PBUILD prints the pound sign prompt (#), type a command line in the format:

logfile = commandfile[-]

CPATCH uses the log file to write the log of the actual text file edits as they take place. If you do not specify a log file, CPATCH uses the file name KB:PBUILD.CMD. If you specify a file, the default file type is .LOG. The command file field is the name of the file containing the commands to PBUILD. The default file type is .CMD. In both the log and command files, the default device is SY: and the default account number is the current user account. As an option, you can place a hyphen (-) after the command file specification to cause PBUILD to prompt you for additional command files. Because the prompting of the additional control file question is also dependent on other factors, PBUILD may ask for additional control files if you do not attach a hyphen to the command file specification. After you press LINE FEED, PBUILD chains to the BUILD program, which controls the remainder of the dialogue.

Once you enter a correct command file name, BUILD prints a message to indicate it is copying the specified command file to the public disk structure. The variable [nnn,nnn] represents the command file account number, and the nn in the PBLDnn.TMP specification is the job number. The format of the message is:

*** COPYING FILE dev:[nnn,nnn]command file TO SY:PBLDnn.TMP ***

BUILD then preprocesses the command file. Normally at this point, during the preprocessing of the first command file, BUILD prints a run-time system question that asks you for the name of the run-time system under which you want the executable programs to run. The format of the question is:

Run-Time System<xxxxxx>?

BUILD prints the name of the system primary run-time system in angle brackets. Press the LINE FEED key to accept the default, or enter the name of another run-time system. If you decide not to accept the default, note that the run-time system you specify must include a keyboard monitor and have a default executable file type of .BAC or .TSK. If BUILD does not accept your response, it prints an error message:

1. XXXXXX IS NOT A KEYBOARD MONITOR
2. RUN-TIME SYSTEM MUST HAVE A DEFAULT EXTENSION OF .BAC OR .TSK
3. XXXXXX IS NOT INSTALLED
   ATTEMPT TO INSTALL XXXXXX<Yes or No>?

After BUILD prints either of the first two messages, it prints the RUN-TIME SYSTEM question again. Enter a new run-time system name that conforms to the run-time system restrictions indicated in the error message.

If BUILD prints the XXXXXX IS NOT INSTALLED message, it has found the specified run-time system in account [0,1] but has discovered the run-time system is not installed. After printing the message, BUILD asks if you want to install the run-time system:

ATTEMPT TO INSTALL XXXXXX<No>?

BUILD inserts the run-time system name in the XXXXXX field, which represents the run-time system that BUILD will attempt to install. Type NO if you want BUILD to return to the RUN-TIME SYSTEM prompt. One of the following cases occurs if you type YES:

1. BUILD installs the run-time system successfully and prints the message, XXXXXX INSTALLED. BUILD asks the CUSP COMPILER question, described below, if you selected RSX or BP2COM as your run-time system.

2. BUILD installs the run-time system, determines whether it includes a keyboard monitor and has the correct file type, and then prints one of the following two error messages:

   a. XXXXXX IS NOT A KEYBOARD MONITOR
   b. RUN-TIME SYSTEM MUST HAVE A DEFAULT EXTENSION OF .BAC OR .TSK.

   After printing either of these messages, BUILD tells you it is removing the previously installed run-time system by printing the XXXXXX WILL BE REMOVED message. The program returns to the RUN-TIME SYSTEM question.

3. BUILD may detect errors other than those already described. When it does, it prints:

   RUN-TIME SYSTEM OPERATION FAILED<text>

   The text can be any valid RSTS/E error message. After printing the error, BUILD returns to the RUN-TIME SYSTEM prompt.

If you selected either the BP2COM or RSX run-time system, BUILD asks the following CUSP COMPILER question:

USE THE CUSP COMPILER 'CSPCOM'<YES>?

Type YES if your run-time system is RSX; type NO if it is BP2COM.

After the preprocessing of the first command file, BUILD may prompt you for the name of another command file by printing:

ADDITIONAL CONTROL FILE IS <process file>?

Type another command file specification and terminate the response by pressing the LINE FEED key. Like the previous command file procedure, BUILD prints the COPYING FILE message and then preprocesses the file. BUILD continues to print the ADDITIONAL CONTROL FILE IS question until the default is selected. Press the LINE FEED key to accept the default.

BUILD prints the current date and time and begins executing the command files. If you did not attach a hyphen to the original command file specification, PBUILD may not ask for additional control files. Therefore, if you need to process more than one command file, attach a hyphen to the logfile = commandfile[-] specification to ensure BUILD asks the ADDITIONAL CONTROL FILE question. Note that PBUILD prints the COPYING FILE message, preprocesses the command files, but does not execute any command file until you specify all files.

During the preprocessing phase, BUILD may ask additional dialogue questions. These questions cannot be listed here; they are initiated by the particular command file that BUILD is processing. There are many control files that might be processed, each with its own unique set of questions. You must therefore refer to documentation regarding the particular command file to answer these questions.

C.4.2 PBUILD/BUILD Terminal Output

PBUILD/BUILD performs all patch operations by printing the required commands on a pseudo keyboard and echoes them to the user's terminal. Every operation causes an output to the terminal to provide an accurate log of all operations. The steps that occur in a patch operation are summarized below. Note that the commands described are those which are printed on the pseudo keyboard and thus require no user input:

1. RUN $CPATCH

   This runs the CPATCH program.

2. outfile = infile

   This string is typed to the prompt FILE TO PATCH issued by CPATCH. Infile is the program to be patched, with the input device and account number being that which is entered at the beginning of the dialogue. The output file can be one of two possible values. If the patched sources are to be saved, and the device is a file-structured disk, the output file will be the same as input file but with the device and account number selected in answer to the WRITE PATCHED SOURCES TO question. If the sources are not to be saved, or if they will be saved on a device other than disk, the device and account number will be the library account.

3. logfile = commandfile

   A command line in this format is typed in response to the CPATCH prompt. CPATCH uses the command file to perform the patch to the text file. The file name is obtained from the command file currently being executed by PBUILD. If the name as specified within the command file is missing either a device or an account number specification, the device and/or account number that contained the command file executing is used. The log file in this command line represents the log file you enter in response to the PBUILD pound sign prompt (#). This step is repeated for each patch to the particular source program.

4. ^Z (CTRL/Z)

   This character combination is forced to the keyboard after all patches have been applied. It causes CPATCH to print the FILE TO PATCH prompt.

5. ^Z

   A second CTRL/Z causes the program to return to the RSTS/E keyboard monitor.

6. OLD outfile

   This is typed only if the compile option was selected in the dialogue. It causes the source file to be compiled.

7. COMPILE comfile

   This is typed if the compile option was selected. It causes the compiled version of the program to be saved as the file named comfile. The file to compile normally has the same file name as the output file but has the same device and account number as the library device entered in the dialogue. If a new name was provided in the PBUILD command file, that name appears as the file to compile.

8. RUN [1,2]PIP.SAV
   *<permfile> = <outfile>
   ^Z

   If the patched sources are to be saved and the save device is not a disk, then these commands are printed on the pseudo keyboard. The permanent file has the same file name as output file but has the same device and PPN as the response to the WRITE PATCHED SOURCES TO question.

9. RUN [1,2]PIP.SAV
   *<outfile>/DE:NOWARN
   ^Z

   This command is used to delete the output file that was created in the library account. It only appears if the previous copy operation was performed or if the current patched sources were not to be saved.

10. RUN [1,2]PIP.SAV
    *<backfile>/DE:NOWARN
    ^Z

    If the output file and the input file are the same file name, CPATCH creates a backup file (backfile). If that occurs, this command deletes it.

In a normal patch operation only some of the CPATCH commands actually appear. However, CPATCH does perform some combination of the above commands for each patch in the PBUILD command file.

C.5 Building CPATCH Command Files

This section describes how to create new files used by CPATCH to perform the actual edit of your source programs.

C.5.1 Building the Patching Command File

To make a new patch, you must create a CPATCH command file by running the CPATCH program. Enter the commands at your terminal and use a file as the log. The output log becomes the new CPATCH command file.

C.5.1.1 File Naming Convention – Patch files are much easier to manage if you adopt a consistent file naming convention. The convention used for all patches to the Commonly Used System Program (CUSP) Library is presented here as an example.

CPATCH command files correspond to articles in the *RSTS/E Maintenance Notebook*. The patch file names are based directly on the article sequence number and have the form: PAnnmm.kkk. The letters nn and mm identify the component and subcomponent numbers, respectively, and kkk is the number of the article corresponding to a particular patch. There is always an article corresponding to all command files. However, there are articles that have no corresponding patches (notes, restrictions, and so forth).

A typical example might be the program QUMRUN. Assume the first article in the *RSTS/E Software Dispatch* concerning QUMRUN describes a patch. The article's sequence number is 14.6.1, where the number 14 is the component number (Spooler and Operator Services Package), 6 is the subcomponent number, and 1 indicates it is the first article published about QUMRUN. The corresponding patch file name would be PA1406.001.

C.5.1.2 Creating the CPATCH Command File — Run the CPATCH program to create the CPATCH command file. The file for which the patch file is being created is specified at the FILE TO PATCH prompt, and the keyboard is specified as the command file with the CPATCH command file being created as the log file. The patch is then manually applied. Note, however, that CPATCH is a line editor which means its text buffer contains only a single line of text at any one time. When the patch is completed, the log file contains a CPATCH command file that performs the same patch when applied using CPATCH. The commands to create the CPATCH command file to perform the first patch to QUMRUN is an example.

You first start the program by typing:

```
RUN $CPATCH
```

After printing a header line, CPATCH prints:

```
FILE TO PATCH-
```

The response to this prompt would be:

out:QUMRUN.BAS = in:QUMRUN.BAS

You place on the input side of the command line the device name and PPN that contains the unpatched source file QUMRUN.BAS. On the output side, specify any device and PPN. Use a different device name and PPN than the one used for the input specification. You then can test the patched program before replacing the original source.

CPATCH prints a pound sign prompt (#) to which the correct response for the example patch would be:

```
PA1406.001=
```

With PA1406.001 on the output side of the command line, CPATCH uses it as the log file. There is no specification on the input side, and thus CPATCH uses the terminal (KB:) as the default command file. Note, however, that you may want to check or determine the patch checksum by including the /CS switch, as described in Section C.5.3.

CPATCH prints the asterisk prompt (*) again. At this point, use CPATCH editing commands to perform the patch on the file. The following sections introduce you to the editing capabilities of the CPATCH program.

C.5.2 Editing with CPATCH

After CPATCH prints the asterisk prompt, you are ready to use the CPATCH line editor to perform the patch. When the patch is complete, use the EX command to return to the pound sign prompt (#). A CTRL/Z returns the FILE TO PATCH prompt, and another CTRL/Z exits the program. The file PA1406.001 (from the example in the previous section) then becomes the CPATCH command file required to perform the patch.

The CPATCH editor is a character-oriented text editing program written in BASIC–PLUS for use on the RSTS/E operating system. It reads ASCII files from any input device, makes specified changes, and writes on any output device. You operate the editor through the use of commands typed at your terminal. The basic editing process can be divided into three parts:

1. Reading the input text into an internal buffer
2. Changing the text stored in the buffer
3. Transferring the revised text to a new file

The following sections list terms and definitions and describe the commands used with the CPATCH editor.

C.5.2.1 CPATCH Editor Terms and Definitions – The CPATCH editor enables you to perform editing operations on ASCII text. The program stores the text to be edited in an intermediate area of memory called a buffer. It does not alter text in the input files.

The editor refers to text in the buffer by using a character location pointer called a cursor. The cursor is considered to reside between any two characters. At the start of editing operations, the cursor precedes the first character in the buffer. You move the cursor during editing operations according to the type of editing being performed. You can refer to text in the buffer as so many characters or lines preceding or following the cursor.

To edit text, you must specify a command or series of commands in response to the asterisk prompt printed by the CPATCH program. The commands are classed according to the type of operation they perform:

Cursor Manipulation	Moves the cursor without altering text within the buffer.
Character Search	Finds a specified occurrence of text within the buffer in order to facilitate editing.
Character Manipulation	Adds to and removes characters or a line of text from the buffer.

Many editor commands are character-oriented. That is, they affect a specified number of characters preceding or following the cursor. The argument of these commands specifies the number of characters in the buffer on which to operate. The number of characters you specify with the argument n is the same forward (n) as backward (–n). The LF, CR, and NUL characters, although not printed, are embedded in text lines, counted as characters in character-oriented commands, and treated as any other text characters.

Some commands are line-oriented. The argument of these commands specifies the number of lines on which to operate. Because the editor counts the line-terminating characters to determine the number of lines on which to operate, an argument n does not affect the same number of lines forward (positive number n) as it affects backward (negative number –n). For example, the argument –1 applies to the line beginning with the first character following the second previous end-of-line and ending with the character preceding the cursor. The argument 1 in a line-oriented command, however, applies to the text beginning with the first character following the cursor and ending at the first end-of-line. Thus, if the cursor is at the center of the line, the argument –1 affects one and one-half lines backwards from the cursor and the argument 1 affects one-half line beyond the cursor.

Character search and manipulation commands operate in command mode. The text used to search or being manipulated can be a single character or a group of characters (called a string). If the text does not contain a CR or LF character and is small enough to fit on a single typed line, command mode can be used to specify the text. In command mode, the editor expects the first character following a search or manipulation command to be a delimiting character for the desired text. The editor uses as the text the characters between delimiters. The delimiting character, therefore, cannot appear in the text, nor can the characters CR and LF.

To specify the word INPUT as the text in command mode, you can use the following string with the search or manipulation commands:

```
/INPUT/
```

The editor uses the characters between the slash delimiters (/) as the text. In command mode, you can use any printable character that does not appear in the text as a delimiter. If the delimiting character appears in the text, the editor attempts to interpret the remaining characters as commands.

To prevent ambiguities when a file is used for command input, CPATCH requires all "invisible" characters, (ESCAPE, LINE FEED, and so forth) to be translated to a visible form. This translation is always done on log file output to permit log files to be used for command input with no modification. Because the output is used for patch verification, it must be typed exactly. An error causes a rejection of the patch.

The translation is:

Command File Format	Control Characters
<TAB>	ASCII 9 (horizontal tab)
<FF>	ASCII 12 (form feed)
<ESC>	ASCII 27 (escape)
<LF>	ASCII 10, 13, 0 (line feed sequence)
<CR>	ASCII 13,10 (carriage return sequence)
<null>	ASCII 0 (isolated null character)
<13>	ASCII 13 (isolated carriage return character)
<10>	ASCII 10 (isolated line feed character)

C.5.2.2 CPATCH Editor Commands – The commands you can use with the CPATCH editor are listed below:

Advance Command (nA)

The Advance command is a line-oriented command that moves the cursor to a point preceding the first character of a line, depending on the size of the argument used. If you do not include an argument, the editor selects one line forward as the default. That is, the A command moves the cursor to the beginning of the next line. Arguments and their effect on the Advance command are listed as follows:

nA Advances the cursor n lines and positions the cursor at the beginning of the line.

0A Moves the cursor to the beginning of the current line.

Change Command (nC/xxxx/)

The Change command changes a specified number of characters following the location of the cursor. The text </xxxx/> must be set off by delimiters.

The C command is equivalent to an Insert command followed by a Delete command. Arguments and their effect on the C command are described as follows:

nC Replaces n characters following the cursor with the specified text. The cursor is placed after the inserted text. The C command does not affect text beyond the current line.

–nC Replaces n characters preceding the cursor with the specified text. The cursor is placed after the inserted text. The C command allows you to make changes to the current line only.

0C Replaces the current line up to the cursor with the specified text.

Delete Command (nD)

The Delete command is a character-oriented command that deletes n characters in the page buffer beginning at the cursor. If you do not include a value for n, the editor deletes the character immediately following the cursor. After executing the command, the editor places the cursor at the first character following the deleted text. The following list describes each argument and its effect on the Delete command:

nD Deletes n character following the cursor. The D command does not allow you to delete text beyond the current line.

–nD Deletes n characters preceding the cursor. You cannot use the D command to delete text other than on the current line.

0D Deletes the current line up to the cursor. The editor positions the cursor at the first character following the deleted text.

Get Command (nG/xxxx/)

The Get command is a search command that allows you to search for the nth occurrence of the specified text (/xxxx/) starting at the current cursor location. If you do not include a value for n, the editor searches for the first occurrence of the text. The search ends when the editor either finds the nth occurrence or encounters the end of the buffer. If the search is successful, the editor places the cursor at the end of the searched text. The editor prints the G command followed by a question mark (?) to indicate an unsuccessful search. In that case, the cursor follows the last character in the buffer.

Whole Command (nH)

The Whole command reads each page of the primary input file into the buffer until the nth occurrence of the specified text object is found. The editor begins at the cursor and searches the current buffer until the nth occurrence of the text is found or the end of the buffer is reached. If the search is successful, the editor positions the cursor immediately following it. If the editor does not find the nth occurrence in the current buffer, it writes the buffer to the primary output file, clears the buffer, reads the next page of the primary input file into the buffer, and continues the search. The search is unsuccessful when the nth occurrence is not found and the end of the primary input

file is reached. The H command followed by a question mark indicates an unsuccessful search. When this occurs, the editor copies the entire contents of the primary input file to the primary output file and places the cursor at the beginning of an empty buffer. The H command operates in a forward direction only; thus, negative arguments are not allowed.

Insert Command (I/xxxx/)

The Insert comand allows you to insert text (/xxxx/). The editor inserts the text and places the cursor after the last character of the inserted text. You cannot use an argument with the Insert command. Up to 80 characters per line can be specified by typing the the letter I on one line followed by the RETURN key and the text to be inserted on the following line(s). Execution of the command occurs when you press the LINE FEED key. If the text does not contain a carriage return or a line feed character, it can be typed on the same line as the I command but must be set off by delimiters.

Jump Command (nJ)

The Jump command moves the cursor over a specified number of character locations. If you do not include an argument, the J command moves the cursor one character position forward. Arguments and their effects upon the Jump command are described as follows:

- nJ Moves the cursor forward one character. The J command does not affect any text beyond the current line; that is, a command such as –400J only moves the cursor to the beginning of the current line.
- –nJ Moves the cursor backward n characters. The J command does not affect any text beyond the current line; that is, you cannot move the cursor further than the end of the current line.
- 0J Moves the cursor to the beginning of the current line.

Kill Command (nK)

The Kill command removes n lines of text (including the carriage return and line feed characters) from the page buffer beginning at the cursor and ending with the nth end-of-line. The editor places the cursor at the beginning of the line following the deleted text. The following list describes each argument and its effect on the Kill command:

- nK Removes the character string (including the CR and LF sequence) beginning at the cursor and ending at the nth end-of-line.
- 0K Removes the current line up to the cursor location.

List Command (nL)

The List command prints at your terminal lines of text as they appear in the buffer. An argument preceding the L command indicates the portion of the text to print. The cursor is unaffected by the L command. Arguments and their effect on the List command are described as follows:

- nL Prints one line of text at the terminal. The L command accepts only zero and one as arguments.
- 0L Prints the current line up to the cursor.

Verify Command (V)

The Verify command (V) prints at your terminal the entire line in which the cursor is located. It provides an easy means of determining the location of the cursor after a search is completed and before any editing commands are given. You can also use the V command after you have typed an editing command to allow you to check your input. The V command does not take arguments.

EX Command (EX)

To end an editing session, you use the EX command. It writes the buffer to the primary output file, transfers the remainder of the primary input file to the output file, closes all open files, and renames the temporary file as the edited, primary output file. The following dialogue shows the procedure:

```
*EX

#^Z
```

The editor prints the pound sign prompt (in response to which you can enter more input and output specifications). Typing CTRL/Z ends the editing session and control returns to your keyboard monitor.

You should periodically use the EX command to end the editing session. The primary input file remains intact while you are editing. The temporary file retains the revised text as edits are made. If the system crashes during the editing session, the primary input file is not disturbed, but the edits you are making are lost. Therefore, it is recommended that you frequently exit to update the primary output file. In the event of a system crash, the amount of editing lost is limited to those edits made since the beginning of the latest session.

C.5.3 Verifying the Patch

The CPATCH program provides two switches to verify the accuracy of the patching operation. To make sure CPATCH applies a patch correctly, you must specify the /CS switch while creating a patch file and the /CS:n switch when you are applying the patch.

To create a patch file, run CPATCH and answer the FILE TO PATCH prompt. After you enter the name of the file to patch and press the RETURN key, CPATCH prints a pound sign prompt (#). Because you are creating a file to store the patches, enter its file specification followed by an equal sign, attach the /CS switch, and press the RETURN key. CPATCH prints an asterisk prompt (*) to indicate its readiness to accept the patch.

After you enter the patch and terminate your input by typing EX, CPATCH calculates a number, called a checksum, and prints it on your terminal. CPATCH uses this number while applying the patch to verify the patch was correctly incorporated. CPATCH prints the PATCH COMPLETE message to mark the end of the operation and returns to the pound sign prompt (#).

You can then enter the name of a new patch file or type CTRL/Z to return to the FILE TO PATCH prompt. If you also type CTRL/Z in response to the FILE TO PATCH prompt, CPATCH terminates and returns control to your keyboard monitor. The following terminal output illustrates the use of the /CS switch and the creation of a checksum number:

```
RUN $CPATCH
<CPATCH's header line>
File to patch - PATCHA.BAS=PATCHA.BAS
#PA2325.001=/CS
*G/-10/-1D
*EX
Checksum = 53437
Patch from _KB:[1,247]CPATCH.CMD complete.
#^Z
File to patch - ^Z
```

After creating the patch file, you can use CPATCH again to apply the patch. As before, run CPATCH, specify the file to patch, and press the RETURN key to receive the pound sign prompt (#). In response, attach the /CS:n switch, replacing n with the checksum number that CPATCH calculated previously (53437). Press the RETURN key, enter the patch, and exit by typing EX. If you enter the patch correctly, CPATCH calculates the same checksum number, applies the patch, and prints the PATCH COMPLETE message. Otherwise, it prints an error message. For example:

```
?Actual checksum of 53437 does not match 53537
%Patch from _KB:[1,247]CPATCH.CMD skipped.
```

You must retrace your steps and try again. An example of a successful attempt to apply a patch follows:

```
RUN $CPATCH
<CPATCH's header line>
File to patch - PATCHA.BAS=PATCHA.BAS
#PA2325.001/CS:53437
*G/-10/-1D
*EX
Patch from _SY:[1,247]PA2325.001 complete.
#^Z
File to patch - ^Z
```

The /CS:n switch, n representing a number from 0 to 65535, causes CPATCH to recalculate the checksum as a result of your new input and to compare it to the original calculation. In this case, you entered the patch accurately; CPATCH was able to calculate the same checksum and then apply the patch.

C.5.4 Building the PBUILD Command File

After building a command file of patches, the next step is to build another command file that contains the instructions to PBUILD for installing the patches.

C.5.4.1 Using Comments – Comments may appear anywhere in a PBUILD command file. PBUILD treats any line of text with an exclamation mark (!) as its first character as a comment. If you use the exclamation mark as an account specifier, you must place it in a column other than the first. One or more blanks before the exclamation mark is sufficient.

C.5.4.2 Using Indirect Command Files – Any command line beginning with an ampersand character (@) is used as a file that contains the next commands. The position in the current command file is saved, and all subsequent input comes from the file specified until an end-of-file is reached. Command input then resumes from the saved position in the command file. You can nest indirect command files up to 15 levels deep.

You must observe the following rules when using indirect files.

- The default file type for indirect command files is .CMD. Therefore, you must explicitly specify any other file type.
- If either the device or the PPN is absent from an indirect file name, the device and/or PPN is obtained from the command file name at the previous level of indirect. To avoid ambiguity in device specification, place all command files in the same account and make sure that they do not include a device and PPN. Instead, the device/PPN for all indirect command files should be specified at PBUILD's pound sign prompt, setting the default value for all subsequent indirect command files.

C.5.4.3 Using Underscore (_) as a Quote Character – You can quote any character or command with the underscore character (_). For example, if it is necessary to have the text "$FORCE" not interpreted as a $FORCE command, prefix the command with the underscore character by typing: _$FORCE. In doing so, you alter the way in which the system responds to data following the underscore.

C.5.4.4 Command File Statements – There are four types of PBUILD command file statements:

$FORCE	Forces command lines to the keyboard until the next $PATCH or $END statement.
$PATCH	Identifies the file to be patched.
patchfile	Contains the specification for a patching file.
$END	Indicates the end of patching for one file.

Comment lines can appear anywhere in a PBUILD command file but usually appear at the beginning of the file to identify its content. PBUILD treats any line of text beginning with an exclamation mark (!) as a comment.

To patch a file, the PBUILD command lines must be in the following order:

```
$PATCH statement
Patchfile statement
        .
        .
        .
Patchfile statement
$END statement

$FORCE statement
(commands to be forced)
$END statement
```

$FORCE Statement

The $FORCE statement is not required for patching BASIC–PLUS programs. It allows you to batch commands into the file (to run programs, delete files, and so forth). For example, ONLPAT is run in this manner to patch binary files. If $FORCE is used, it and its related command lines must be placed outside the bounds of a $PATCH/$END set, either before a $PATCH statement or after an $END statement.

The command lines following a $FORCE statement use logical names to allow device and PPN assignments (those made during the PBUILD dialogue). The substitutions are:

^SYSTEM:	Is the value for the device and PPN specified as the library device.
^SYSDSK:	Is the value for the device and PPN specified as the library or the system device.
^INPUT:	Is the value for the device PPN specified as the unpatched source file.
^SAVDEV:	Is the value for the device and PPN specified as the patched source file. If sources are not to be saved, SAVDEV defaults to the system value.
^Z	CHR$(26%)(CTRL/Z)
^C	CHR$(3%)(CTRL/C)

The $FORCE statement is in effect until a $PATCH statement or end-of-file is encountered. Any command line beginning with an at sign character (@) is treated as an indirect command file.

$PATCH Statement

The $PATCH statement is the first statement in a sequence of statements to perform a patch. The format of the statement is: $PATCH filename. You must specify the file name. The default value for the file type is .BAS and the default device and PPN is the one specified in the opening question of the dialogue. If a file type or a device and PPN is specified with file name, it overrides the default. For example, suppose the input device entered at the

beginning of the dialogue was MM:[120,51]. The following list shows various values for file name in the $PATCH statement and the resulting file name after filling in the defaults:

$PATCH filename	Name actually Used
TEST	MM:[120,51]TEST.BAS
TEST.	MM:[120,51]TEST.
SY:TEST	SY:[120,51]TEST.BAS
SY:[1,2]TEST.FIL	SY:[1,2]TEST.FIL

The statement consists of a single file name, which is the name of a command file to be used by CPATCH to perform the source text editing. Any number of $PATCH file statements are permitted. There is one patch file statement for each patch to the program.

The default file type for the patch files is .CMD. The default device and PPN is that of the current PBUILD command file which is open. Again, to avoid confusion, make sure all command files include indirect files and patch files, are in the same account, and have no device and PPN specified.

$END Statement

The $END statement indicates the end of a patch. It also selects a number of options for the operations that override those entered in the dialogue. The format of the end statement is:

$END [/NC] [newname]

When PBUILD encounters the $END statement, it normally performs the OLD and COMPILE operations. However, you can change the operation by providing either (but not both) of two optional arguments. The first is the /NC switch. When you include this switch, no compile operation is performed. The patched source is saved on the account specified as the library in the opening dialogue.

The newname option allows the name of the file to be changed after it is compiled. The default name for the compiled program is ˜SYSTEM:file.BAC; ˜SYSTEM: is replaced by the library device and PPN specified in the opening dialogue, and file is the file name specified in the $PATCH statement. The protection code is always <124>. If any of these values are to be changed, their new value is included as the newname argument. Only those values that are different from the default must be included. For example, if the file's name is to be ˜SYSTEM:file.BAC<232>, only the protection code is different from the default, therefore the argument would be <232>, for example, "$END <232>".

If you want to have a file compiled to the system device instead of the library device, use the newname option. PBUILD substitutes the system device and PPN for the string ˜SYSDSK:. For example, if the system device specified in the opening dialogue is SY0:[1,2], the new name string ˜SYSDSK:<232> results in the file name SY0:[1,2]file.BAC<232>.

C.5.4.5 Sample PBUILD Command File – The PBUILD command file must reference the CPATCH file created in Section C.5.1.2. You can create a new PBUILD file with the required PBUILD commands, or you can insert the commands in an existing file. In either case, use an editor to edit the file.

In the example, the CPATCH file for the first patch to QUMRUN was created. Now the commands to reference this file are inserted in the PBUILD file (either new or existing). The commands required are:

```
$PATCH QUMRUN
PA1406.001
$END
```

Because this is the first patch to QUMRUN, all three of the commands must be added. If there were already existing patches to QUMRUN, you must add the reference to the new patch before the $END statement of any previous patches to QUMRUN. For example, if the second article about QUMRUN is also a patch article, the CPATCH command file for that patch would be PA1406.002. A patch already exists for this program (PA1406.001), therefore insert PA1406.002 immediately before the $END statement in the sequence of commands that apply the previous patch. The resulting command sequence is:

```
$PATCH QUMRUN
PA1406.001
PA1406.002
$END
```

These commands cause PA1406.001 to be applied to QUMRUN. The file PA1406.002 would be applied to the patched file resulting from first applying PA1406.001.

If you want to include checksums, add the /CS:n switch after each patch file reference. For example:

```
$PATCH QUMRUN
PA1406.001/CS:57315
PA1406.002/CS:12345
$END
```

C.6 Error Messages

This section describes the errors that can occur when running the patching programs. Error messages can originate either from the PBUILD or the CPATCH program. Errors from CPATCH occur either when PBUILD is running CPATCH or when CPATCH is running standalone.

Errors can occur in PBUILD when:

1. You enter information required to run the program but the information is incorrect. In this case, PBUILD displays the incorrect input with a text error message and prints the prompt requesting the input again.
2. You enter an incorrect command in a command file. Errors occurring in PBUILD while it is running from the command file are always fatal. PBUILD prints the error message and returns control to your keyboard monitor.

3. CPATCH, which is run by PBUILD, attempts to apply an invalid CPATCH command file as a patch to a program. The program will continue to run.

CPATCH error messages appear at the user's terminal. If they occur while running under PBUILD, PBUILD continues to run. If they occur running CPATCH standalone, they may cause CPATCH to terminate execution. When running CPATCH under PBUILD using patch files supplied by DIGITAL, the first error message in Table C–3 is the only one which should ever occur.

Table C–3: PBUILD Error Messages

Message and Meaning
MUST BE PRIVILEGED TO RUN PBUILD An attempt was made to run PBUILD from a non-privileged account.
?FILE NAME ERROR This error indicates that a bad file name was found in the command file currently executing.
?BAD FILE NAME: <filename> Filename does not follow proper syntax rules. This error can occur during the dialogue.
?<file string> NOT PERMITTED File string is valid but contains a feature that is not permitted for the operation, such as wild card characters or a file name when only a device and PPN was expected.
?FILE NOT FOUND: <filename> The syntax of the file name is correct but refers to a nonexistent file.
?<file string> ERR–<error number> The file string is invalid for a reason other than the previous three error messages. The number returned by the message represents the error number produced by the BASIC–PLUS run-time system.
?COMMAND FILE MUST BE ON DISK A command file name was entered with a device specifier other than a file-structured disk. Command files must be copied from the device to disk before running PBUILD.
?CAN'T COPY <file1> TO <file2> <BASIC–PLUS error message> Occurs after the message !COPYING TEMP FILE <file1> TO <file2>. PBUILD was attempting to copy the source file file1 from the current account to the permanent account. The copy failed for the reason given in the BASIC–PLUS error message. The BASIC-PLUS errors that are most likely are ?NO ROOM FOR USER ON DEVICE and ?DEVICE NOT AVAILABLE. Others are various magnetic tape errors or hardware I/O errors.
FATAL ERROR NO. <error number> ON LINE <error line> LEAVING PROGRAM The program was terminated by an unexpected error, which may indicate a system error or a hardware failure. The error returns the BASIC–PLUS error number and line number.

Table C–4: CPATCH Error Messages

Message and Meaning

?MISMATCH
 	
Occurs when editing a version of the file that was not correct, for example, attempting to patch a file that was already patched. When this happens, the patch is ignored, and the file is restored to the state before attempting to apply the patch. The log file (or KB: if that is the log) shows the cause of the mismatch by printing:
 	
`= cmd=>  string`
`= log=>  string!`
`?mismatch?`
 	
String is the contents of the command file that is the correct program response to the previous command in the command file; string! is the actual program response produced. It can be a verify on a line that has been modified or a CPATCH error message, such as ?SRCH FAIL when a string that was expected has been deleted.
 	
This is a non fatal error when running under PBUILD. PBUILD continues to run normally but the source file used for any commands to follow will be the original, unmodified source.

?SRCH FAIL The string searched for with the G command was not found.
?EOF If the error occurs in response to the H command, it indicates that the string was not found, and the source input file has been read to the end of file. If it occurs while running from a command file, it indicates that an end of file occurred in the command file before an EX command was encountered.
?BAD ARG FOR <command> The argument provided with command is invalid for that command. For example, the L command only permits 0 and 1 as valid arguments.
?BAD COM: <command> Command is not a valid command string.

?CAN'T BE THE SAME <log> = <command>
 	
Unless the log file and the command file are both the keyboard, they must be different files. If they are not, this message is printed:
 	
`FATAL ERROR NO. <err> ON`
`LINE <error line>`
`LEAVING PROGRAM`
 	
The same as in PBUILD. However, err = 10 is possible in normal operation. That error number indicates a protection violation, which can occur because CPATCH does not have to be run in a privileged account.

?FATAL ERROR NO.<error number>ON LINE <error line> IN AUTOED The same as above, however, the error occurred in the program AUTOED which is chained to by CPATCH. This error does not terminate CPATCH, however. It has the same effect as a ?MISMATCH error, but it may still indicate the need for to send DIGITAL an SPR.

(continued on next page)

Table 10–4: CPATCH Error Messages (Cont.)

Message and Meaning
PLEASE RUN CPATCH An attempt was made to run AUTOED as a standalone program. <string> NOT PERMITTED ?BAD FILE NAME: <string> ?,string> ERR = <num> FILE NOT FOUND ?CAN'T COPY <file1> TO <file2> These messages have the same meaning as they do for PBUILD: refer to Table C–3.

Appendix D
DCL Error Messages

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 specific to the LINK command, and error messages for batch processing. Included are the messages you can get when using privileged DCL commands described in this manual, in addition to those for general DCL use.

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. Those messages are listed in the *RSTS/E System User's Guide.*

D.1 Special Characters Used in Error Messages

The question mark (?) and percent sign (%) characters preceding a message indicate its severity. The question mark precedes fatal error messages, which means that the command failed to do what you asked. For example:

```
?Too many parameters
```

In this case, the command failed because you included too many parameters. By contrast, the percent sign identifies a warning message, which means that the command may not have worked as you intended. For example:

```
%Logical name has not been assigned
```

In this case, the MOUNT command worked but no logical name was assigned to the device.

Informational messages do not have a preceding character; they provide extra details about the effects of a command. For example:

```
Queue file being reorganized - please wait...
```

This message informs you that, as a result of your command, the file is being reorganized.

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

D.2 General Error Messages

The following table lists error messages that you can get when using DCL commands.

Table D–1: General Error Messages

Message and Meaning
?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 a user attempts 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.
?Can't mount a private disk as public You tried to mount as public (using the MOUNT command's /PUBLIC qualifier) a disk that was initialized as private.
?Can't rebuild disk because device is write protected A privileged user attempted to rebuild a disk (using the MOUNT command's /REBUILD qualifier), and the drive is write protected. Therefore, the mount and rebuild operations fail.
?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

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning

error occurred when DCL tried to run the program. See the system manager.
 	
`?<command name> command failed (error #<RMS error number>)`
The queue manager failed to complete the command you specified. The system manager should submit an SPR.
 	
`?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 privilege, 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 conflict", which covers a greater number of conditions.)
 	
`?/COPIES argument too large`
The argument you specified for the /COPIES qualifier is greater than the maximum value of 255.
 	
`?/COPIES argument too small`
The argument you specified for the /COPIES qualifier is less than the minimum value of one.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
`?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.
`?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 off line.
`?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 the system manager. 4. The device is a keyboard line for pseudo keyboard use only.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
`?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.
`?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 the system manager.
`Disk is being rebuilt - wait...` This informational message is displayed when a privileged user attempts to rebuild a disk. The disk is logically mounted when the rebuilding operation is complete.
`%Disk is locked and mounted non-shared, read-only` A privileged user specified /NOREBUILD (to suppress rebuilding) and /NOSHARE in the MOUNT command and the disk was dirty. Therefore, the disk is mounted non-shared and locked, and read-only access is granted. ("Locked" means that access is allowed to privileged users only.)

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
%DISK is locked and mounted private, read-only A privileged user specified /NOREBUILD (to suppress rebuilding) and one of the following in the MOUNT command: 1. Either /SHARE or /PRIVATE 2. Neither /SHARE, /NOSHARE, /PRIVATE, nor /PUBLIC The message results because the disk was dirty. Therefore, the disk is mounted private and locked, and read-only access is granted. ("Locked" means that access is allowed to privileged users only.)
?DISK is mounted non-shared This message can be displayed for one of two reasons: 1. A privileged user attempted to dismount a private or public disk that was mounted as non-shared by specifying /PUBLIC in the DISMOUNT command. 2. A nonprivileged user attempted to dismount a private disk that was mounted as non-shared by specifying /PUBLIC in the DISMOUNT command. Therefore, the dismount fails. In both cases, the requests to mount and dismount are from different jobs.
?DISK is mounted private or non-shared This message can be displayed for one of two reasons: 1. A privileged user attempted to dismount a private or public disk that was mounted as non-shared (by the same job) or private by specifying /PUBLIC in the DISMOUNT command. 2. A nonprivileged user attempted to dismount a private disk that was mounted as non-shared (by the same job) or private by specifying /PUBLIC in the DISMOUNT command. Therefore, in both cases the dismount fails.
?DISK is mounted public A privileged user attempted to dismount a public disk that was mounted as public, without specifying /PUBLIC in the DISMOUNT command, so the dismount fails.
%DISK is mounted read-only This warning occurs if you do not specify either read or write access (/NOWRITE or /WRITE) when mounting a disk initialized as read-only. Therefore, you are granted read access only.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
?Disk needs rebuilding but device is write protected A privileged user attempted to mount a dirty disk (and did not specify /REBUILD or /NOREBUILD), but the device is write protected and the automatic rebuilding cannot be performed. Therefore, the mount fails.
?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. The 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 online) 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 the 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.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
2. A wildcard file specification does not match any files. 3. A file is not accessible because of its assigned protection code.
`?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 <form-name> does not exist` The form name you specified was not defined by the 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 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.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
`?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. Therefore, you do not need to specify the tape's density with the MOUNT command.
`?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 error can occur when you use a qualifier in the form /qualifier=argument. The qualifier you used 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.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
?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.
?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:.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
?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, etc), but you also gave a qualifier that can only be used with local operations.
?I/O to detached keyboard This message can result from one of 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_COUNT argument too large The argument you specified for the /JOB_COUNT qualifier is greater than the maximum value of 255.
?/JOB_COUNT argument too small The argument you specified for the /JOB_COUNT qualifier is less than the minimum value of one.
?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.
?Job name needed The command you typed requires a job-name.
?Job number too small The job-number you specified is less than the minimum value of one.
?Job specification needed The command you typed requires a job specification.
?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".)

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
%Logical name has not been assigned This warning can be displayed for one of two reasons: 1. A privileged user specifies a logical name that was not assigned as a system logical. (This means that neither the alternate logical name nor the pack-id label were assigned.) 2. A nonprivileged user specifies an alternate logical name when attempting to mount a disk. The mount succeeds in both cases, but the logical name is not 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 error 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 maximum memory size was reached. 2. While running a program, the system required more memory for string or I/O buffer space, and the job's private maximum memory size 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 bracket ([) or left angle bracket (<), but no right bracket (]) or right angle bracket (>).
?Missing closing quote A quotation mark (") is not matched with another quotation mark. (This error 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.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
?Name or account now exists You attempted to either COPY to or RENAME an existing file. This error 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. The system manager has disabled logins. (Possibly, logins are disabled because the system will be shut down shortly.)
?Non-executable file This error 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 the 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.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
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 non-disk 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 commands COPY and CREATE 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 the 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 files and larger files.
`?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.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
?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 maximum memory size. Either the program must be made privileged to allow it to expand above a private maximum memory size, or the 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.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
`?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 busy` The command you typed could not be completed because the spooling device you specified is currently processing a job.
`?Printer already initialized` The device you specified has been initialized already as a spooling device.
`?Printer not initialized` The device you specified has not been initialized as a spooling device.
`?/PRIORITY argument too large` The argument for the /PRIORITY qualifier is greater than the maximum value. For privileged users, the maximum value is 255; for non-privileged users, 128.
`?/PRIORITY argument too small` The argument for the /PRIORITY qualifier is less than the minimum value of one.
`?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 preceded 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.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning

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.)
 	
`%Public disk mounted as private`
 	
A privileged user mounted a disk initialized as public without specifying /PRIVATE, /PUBLIC, /SHARE, or /NOSHARE. The system mounts the disk as private.
 	
`?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. The system manager should 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 either.
 	
`?Spooling package already started`
 	
The spooling package has been already started.
 	
`?Spooling package not running`
 	
The command you typed cannot be processed until the spooling package is started.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
?Stack overflow This message indicates a system problem. The system manager should 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 on COPY or APPEND.
?Too many parameters The 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 already been initialized.

(continued on next page)

Table D–1: General Error Messages (Cont.)

Message and Meaning
?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 cannot contain wildcard characters in the queue-name.
?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 with the /REBUILD qualifier of the MOUNT command. If you are nonprivileged, have your system manager rebuild the disk.

D.3 LINK Error Messages

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 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 D–2: Task Builder Messages for RSX-Based Link

Message and Meaning
?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.
?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 the system manager.
?Overlay tree has too many levels You were already 7 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 the 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.

(continued on next page)

Table D–2: Task Builder Messages for RSX-Based Link (Cont.)

Message and Meaning
?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.

Table D–3: LINK.SAV Messages for RT11-Based Link

Message and Meaning
?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.

D.4 BATCH Error Messages

The following messages are generated by the batch facility.

Table D–4: BATCH Error Messages

Message and Meaning
?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 is 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.
?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.

(continued on next page)

Table D–4: BATCH Error Messages (Cont.)

Message and Meaning
?Unmatched parentheses An opening (left) parenthesis appears in a command, but an accompanying closing parenthesis is not found.
?Unmatched quotation marks Quotation marks (") and apostrophes (') must be paired in a control statement.

Appendix E
Disk Device Sizes

Table E–1 lists the device cluster size and device size (in 512-byte blocks) for each disk that RSTS/E supports. All values are in decimal.

Table E–1: Disk Device Sizes

Disk	Device Cluster Size	Device Size
RX50	1	800
RF11	1	1024 times number of platters
RS03	1	1024
RS04	1	2048
RK05	1	4800
RK05F	1	4800 per unit; 2 units for each drive
RL01	1	10220
RL02	1	20460
RD51	1	19391
RK06	1	27104
RK07	1	53768
RC25	1	50902 per unit; 2 units per spindle
RP02	2	40000
RP03	2	80000
RM02	4	131648
RM03	4	131648
RP04	4	171796
RP05	4	171796
RA80	4	237208
RM80	4	242575
RP06	8	340664
RA60	8	400175
RM05	8	500352
RA81	16	888012

Index

A

B

C

D

E

F

G

H

M

N

Q

R

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