RSTS/E
System User's Guide

Order No. AA–5133C–TC

December 1981

This manual describes the functions and use of the RSTS/E Version 7.1 operating system for the non-privileged user.

OPERATING SYSTEM AND VERSION:	RSTS/E	V7.1
SOFTWARE VERSION:	RSTS/E	V7.1

digital equipment corporation, maynard, massachusetts

First Printing: December 1976
Revised: December 1981

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.

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

Contents

Chapter 5 A Cookbook for Working with Disks and Tapes

Chapter 6 System Utility Programs

Chapter 7 The Batch Processing Program: BATCH

Appendix A Error Messages

Figures

Tables

Preface

What work will you do with a RSTS/E system? The answer depends on what tools you have available. Some of these tools are provided by DIGITAL when you buy a system. Others may have been done by people where you work or from other companies.

Part I of this manual presents an overview of the tools provided by DIGITAL — the hardware and software making up the RSTS/E computer system. Parts II and III describe some specific software tools called utility programs and the batch processor.

Conventions

Throughout this manual, symbols and other conventions are used to represent keyboard characters or aid in the presentation of information. The symbols and conventions used are as follows:

(RET)	The (RET) symbol represents a carriage-return/line-feed combination.
(LF)	The (LF) symbol represents a line-feed character.
^	The circumflex preceding an uppercase character indicates a control character.
color	Color-highlighted information in examples is typed by the user.
type print	As these terms are used in the manual, the user types and the system prints.
uppercase lowercase	As used in examples of format, information that must be typed as shown is in uppercase; information that is supplied by the user is in lowercase.

Chapter 1
Introduction

One of the best ways to find out about RSTS/E is to use it. The first thing you will likely encounter is a hardware device called a terminal. Figure 1–1 shows people working at the two general types of terminals: screen and hardcopy. Basically, you type commands and instructions to the computer on the keyboard of the terminal. The system shows you the commands you typed, and possibly the results of the commands, on the screen or on the paper listing of the hardcopy terminals. (Some commands may produce results on other hardware devices, such as a disk or line printer.)

Figure 1–1: Terminals: Your Link to a RSTS/E System

1.1 Getting a Project-Programmer Number and Password

Each time you use RSTS/E at a terminal, you perform a procedure called "logging in." You identify yourself to the system as a valid user by typing two codes: a project-programmer number and a password.

The first thing you should do, then, is talk to your system manager to get your own number and password. "System manager" is a term used to refer to the person at your site who is responsible for seeing that the system is maintained and used properly. This person will likely assign you a project-programmer number and let you choose your own password. He or she must then make sure this number and password are entered into the RSTS/E system.

The project-programmer number is really two numbers, separated by a comma. For example:

```
100,155
```

The first number, called the project number, is generally used to indicate a group of users with a common activity or interest. For example, the 100 in the number above could be used at your site to indicate all users in a certain department, 102 could indicate another department, and so forth.

The second number, called the programmer number, is generally assigned to indicate you alone. For example, you would be the only person using the number 155 on your project on the RSTS/E system.

The password is an alphanumeric code that you choose or that the system manager assigns you. When you type it during the log-in procedure, it is not printed on the paper or displayed on the terminal screen. This secrecy prevents unauthorized persons from learning your password and thus gaining illegal access to the system. To preserve the security of the system, do not write down your password; memorize it as soon as it is assigned.

1.2 Logging In

Once you have a project-programmer number and password, you can log in at a terminal. First, make sure that the terminal is "on-line" (connected to the RSTS/E system). There is usually a knob, switch, or light indicating the terminal's current status as ON–LINE or LOCAL. Switch it to ON–LINE. If your terminal is connected to the RSTS system by a telephone line and modem, you then perform whatever dial-up procedure is necessary.

Once the terminal is on-line, there are several ways to do the log-in procedure. We discuss the easiest way here; other possibilities are described in Section 6.8.

The easiest way to start log-in is to type HELLO and press the RETURN key. The RSTS/E system will respond with the prompt:

```
User:
```

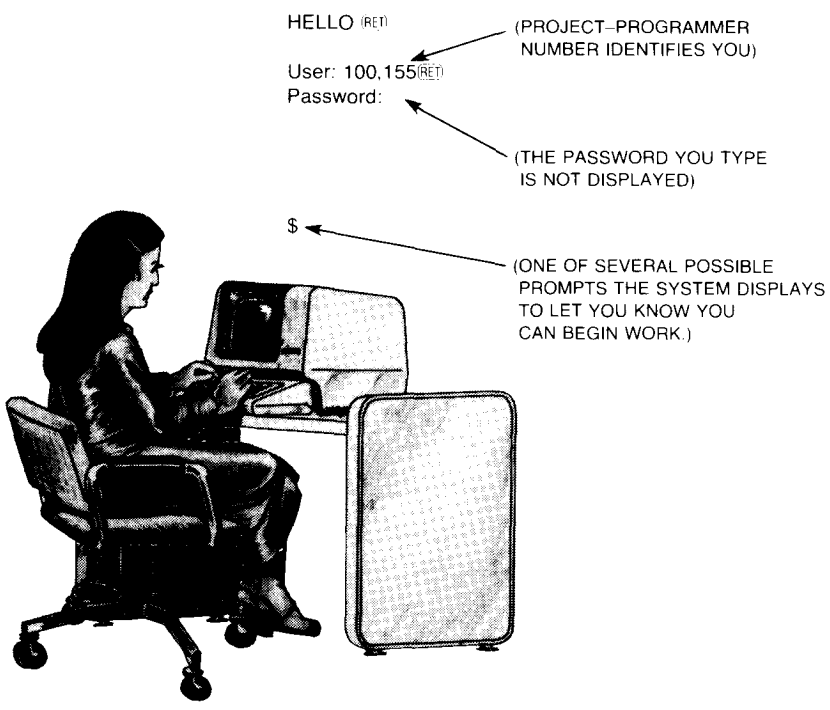
You type your project-programmer number followed by the RETURN key. The system then prompts:

```
Password:
```

You type your password, followed by RETURN. Remember that the password is not displayed at your terminal.

If the codes you typed are acceptable, you are logged in and the system prints the daily message (if any) and a prompt at the terminal.

Figure 1–2: Logging into the System



The daily message, written by the system manager, contains information on operation schedules, any changes or additions to the system, and so forth. For example:

```
HELLO(RET)
User: 100,242(RET)
Password: password(RET)                    (what you type is not displayed)

17-FEB-82

   TO ALL USERS--
   RSTS/E TIMESHARING HOURS WILL BE FROM
   9:30 AM TO 7:30 PM TODAY. FOUR NEW
   TERMINALS ARE AVAILABLE IN THE SYSTEM
   ROOM FOR GENERAL USE.
```

If the codes you entered are not valid, the system prints:

```
INVALID ENTRY - TRY AGAIN
```

You can try again to log in. After five unsuccessful attempts to log in, the system prints the message ACCESS DENIED and ends the log-in procedure.

1.3 Command Language Environments

After you log in, the system is ready to accept your typed commands. You will see a prompt displayed at your terminal — this means you can begin working. The prompt displayed can be any one of those shown below. It indicates which of four possible command language environments you are in.

$	The DIGITAL Command Language (DCL) environment
Ready	The BASIC–PLUS environment
>	The RSX environment
.	The RT11 environment

Some commands — such as RUN to run a computer program — are common to all environments. Other commands are available in only one environment, although the things you can do are similar throughout the command languages. For example, you might use the DELETE command in the DCL environment to delete some work you have done. In the BASIC–PLUS environment, you might use the UNSAVE command. In other environments, you could run a program called PIP to delete your work.

The commands available in the DCL environment (such as DELETE) are English-language commands, as opposed to program names. You may find DCL easier to use. DCL commands also offer the advantage of being similar to commands used on other DIGITAL systems including that on the VAX computer system, called VMS. So if you change from a RSTS/E to a VMS system, or vice versa, you will be able to use many of the same commands.

The BASIC–PLUS environment offers different advantages, particularly useful if you are learning how to write computer programs in the BASIC–PLUS language. The RSX and RT11 environments emulate the commands available on DIGITAL's RSX and RT11 operating systems for the PDP–11 computer. If you have worked with either of these systems, you may feel more "at home" in these environments.

In any case, the prompt that you see after you log in simply shows the environment that the system manager at your site has chosen as the most useful to the largest number of people. If you wish, you can switch to another. We tell more about the command environments in Chapter 3; you can then make your own choice.

1.4 Your "Job"

Your job, at least as far as the RSTS/E system is concerned, is all the work you do at your terminal from the time you log in to the time you log out. The RSTS/E system assigns you a job number when you log in. This number identifies the work you do, for the system's internal "bookkeeping" functions.

The system balances all the work being done on the system using this "job" concept. Each job gets a certain amount of time to use the computer all to itself — called a "time slice." Since the computer can do a lot of work in just a few microseconds, it can seem to you at your terminal that you have the computer all to yourself. That is the central idea of the RSTS/E system: a lot of jobs can share the computer's time and resources at the same time.

The job number appears in many reports produced from the information kept by the system for each job. You can see one such report by typing the command sequence below; it works in all the command environments. What you type is shown in red; the resulting prompts and displays are shown in black.

```
RUN $SYSTAT(RET)
SYSTAT V7 RSTS V7 *SYSNAME*
Output Status to?(RET)
```

SYSTAT is the name of a utility program; it is described in detail in Section 6.16. The dollar sign is a sort of "nickname" given to an account on the system; it simply tells where the program can be found. Thus, with the first line above, you tell the system what program to run and where to find it. The system loads the program into memory from disk and runs it. The running SYSTAT program displays a header identifying itself and the system, and asks where you want the status report to be sent. By typing just the RETURN key, you accept the default: your terminal.

The sample below shows the first section of a typical SYSTAT report. It contains information about items that we haven't discussed yet, but notice some that we have: the job number in the first column, and the project-programmer number in the second column. The fourth column shows the names of programs that people are running on your system. The highlighted line illustrates how you might appear in the report; note that the program running in this case is SYSTAT.

```
RSTS V7       *SYSNAME* status at 14-Nov-81, 10:10 AM Up: 13:48:40
Job    Who     Where     What     Size     State     Run-Time     RTS
 1    [OPR]    Det      ERRCPY   10/31K   SR D25        27.0    ...RSX
 2    [OPR]    Det      OPSRUN   16/31K   SL          1:23.4    BASIC
 3    [OPR]    Det      QUMRUN   16/31K   SL D24      3:02.2    BASIC
 4    [OPR]    Det      SPLIDL   16/31K   SL D26         0.1    BASIC
 5    [OPR]    Det      BATIDL   13/31K   SL D27         0.1    BASIC
 6    [OPR]    Det      BATIDL   13/31K   SL D29         0.1    BASIC
 7    [OPR]    Det      EVTLOG   22/31K   SL            19.3    ...RSX
 8    [OPR]    Det      NPKDVR    9/31K   SL          1:55.4    ...RSX
```

(continued on next page)

```
 9     [OPR]    Det      SYSMAN    8/31K    SL              50.6    BASIC
10      2,253   KB42     NONAME    2/31K    ^C              18.6    BASIC
11  -   1,203   KB37     ...ED2   20/31K    RN               2.2    ...RSX
12     [OPR]    P1J8     NONAME    2/31K    ^C            6:05.2    BASIC
13      2,210   KB32     ...ED2    7/31K    KB  A10         31.1    ...RSX
14      5,227   KB44     PIP      21/31K    KB            1:10.4    ...RSX
15     [OPR]    P0J8     MOM      14/31K    SL               1.6    BASIC
16      6,224   KB28     NONAME    2/31K    ^C               0.1    BASIC
17      5,249   KB30     PIP      16/31K    KB              43.4    RT11
18      5,248   KB29     SYSTAT   21/31K    KB               2.5    ...RSX
19      2,245   KB26     SEND     17/31K    KB               2.2    ...RSX
20     [OPR]    P2J15    SRCCOM   16/31K    RN               2.2    RT11
21    120,223   KB51     NONAME   21/31K    KB               4.5    ...RSX
22     30,225   KB25     MYEDT    19/31K    RN              51.4    ...RSX
23     [SELF]   KB31     SYSTAT   17/31K    RN  Lck          5.7    ...RSX
24      26,13   KB16     NONAME    2/31K    ^C  A12          1.9    BASIC
25      5,204   KB23     MYPROG   21/31K    KB               7.8    ...RSX
```

1.5 Logging Out

When you are done with your work, you log out of the system. As with logging in, there are several ways to do this. The simplest is to type:

```
BYE(RET)
```

The system asks you to confirm that you want to log out. Type Y for "yes".

```
Confirm: Y(RET)
```

The system then displays some information about the session you just completed, and logs you off the system.

```
Saved all disk files; 964 blocks in use, 36 free
Job 23 User 100,155 logged off KB31 at 14-Nov-81 11:28 AM
System RSTS V7 *SYSNAME*
Run time was 59.5 seconds
Elapsed time was 1 hour, 44 minutes
Good morning
```

Other ways to log in and log out are described in Section 6.8 (the LOGIN utility) and 6.9 (the LOGOUT utility).

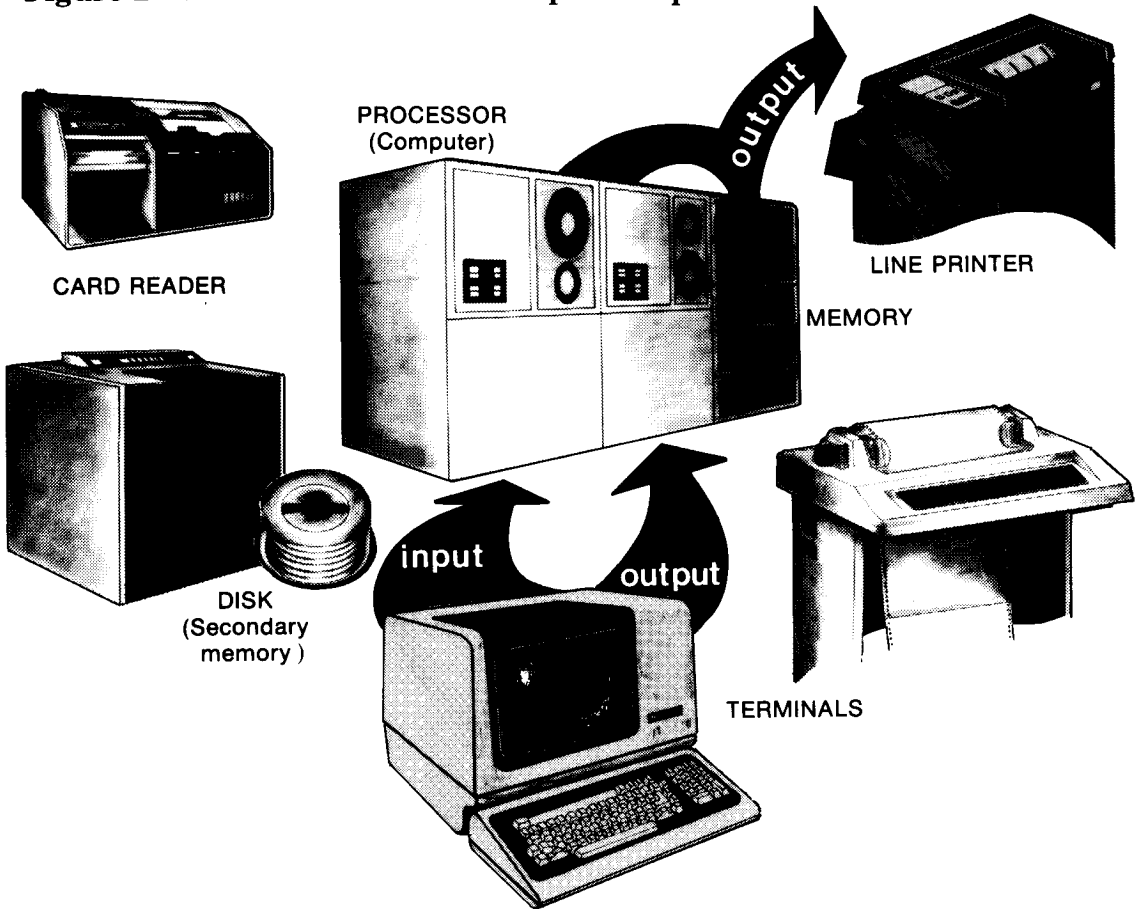
Chapter 2
The Tools Available

The terms "hardware" and "software" distinguish between the two general types of tools available on computer systems to help you with your work. Hardware means an array of equipment; software means a set of computer programs — instructions to the computer.

2.1 Hardware

You have already been exposed to one hardware device: a terminal. There is likely some place at your site where the rest of the equipment is located. Or, you may be at a remote site, with the central computer lab in another building, city, state, or country.

Figure 2–1: The Processor and Input/Output Devices



In any case, the terminal is your link to the "processor": the hardware device that does the computing on your system. The other devices are called "input/output" devices. They store instructions and data for input to the processor, and store or display the results (output) of the computations.

2.1.1 The Processor

The RSTS/E system operates on DIGITAL's PDP–11 series of computers — from the PDP–11/23 to the PDP–11/70. All of these processors (11/23, 11/24, 11/40, 11/44, 11/45, 11/60, and 11/70) offer the same instruction set. That is, the "machine language" that these computers process is basically the same, and they all process instructions and data stored as 16–bit words.* The differences in these processors involves what is called their "cycle time" — the time it takes to execute one computer instruction — and the amount of memory they can access. Computer memory is a storage medium; it can be accessed by the processor much more quickly than other information storage devices. Thus, memory is where instructions and data are stored during the time they are being processed. The "larger" PDP–11s, with faster cycle time and more memory, can get work done more quickly.

If you are interested in details about the capacity of the computer processor at your site, see the *PDP–11 Processor Handbook* for your system.

2.1.2 The Public Disk Structure

Users share computing time on the RSTS/E system, with each being allotted a "slice" of the processor's time. Users may share another system resource as well: the array of devices. One set of devices (disks) is shared by a number of users. This shared set of devices is called the public disk structure, because it is always accessible to all users and because the system treats it as a unit. On some systems, it may include many separate disk drives. One of these, the system disk, contains the system code, language processors, and, possibly, the library of system programs, some of which are described in Chapter 6 of this manual. The other disk drives (the public disks) contain information created by users — programs and data stored in units called "files." (The system disk too may contain user data directories and files in addition to its system information.)

When you are logged into the system and working with a disk file, you are usually unconcerned about which disk in the public structure happens to contain that file. The particular disk is chosen by the system according to current time-sharing needs. Each of the disks contains a master list of users' accounts, and, for each account, a list of all files stored under that account. The system, by using these lists, is able to locate your file when you request it from your terminal.

---

* The size of a computer word — the unit in which any computer processes instructions — is one of the factors determining the "size" of a computer. The 16–bit word establishes the PDP–11 computer, in the general terms of the industry, as a "minicomputer."

2.1.3 Private Disks

A good deal of system file activity — such as creation, access, editing, and deletion — takes place on the public disk structure. Since it is the largest constantly available medium of file storage, it is generally the busiest. But not all the disks used on a system need be in the public structure. Some of them may be private disks, disk packs or cartridges that belong to a single user account or perhaps to a few user accounts, in the sense that these accounts alone are on the disk(s). Only if a private disk already contains an account can files be created under that account on the disk. Without one of these private disk accounts, you can read or edit a private disk file, but only if its protection code permits.

For example, assume that a private disk mounted on drive DL3: belongs only to the members of a specific project group, to those users with project number 200. In such a case, the users who hold account numbers [200,30], [200,31], [200,32], [200,33], and so forth may all create files on private disk DL3:. If you have account [210,33], however, you cannot create files on DL3:, although, protection codes permitting, you can access files already created on that disk.

From your point of view, then, one important difference between a private disk and one on the public structure is that, as long as you can log in, you can always create a file on the public disk, whereas you can do so on a private disk only if you have an account number there. On both types of disk, file protection codes govern your read and write access to existing files. Another difference is that a private disk can be mounted or dismounted while the system is running, whereas a public disk must always be mounted during time sharing. This difference, obviously, is of special concern to the owner(s) of the private disk and to the system manager, who, it should be noted, determines which disk drives on the system are for public and which are for private use.

2.1.4 Assignable Devices

Disks, public and private, are generally not assignable. That is, you cannot, except in rare and special cases, request one for your exclusive, temporary use. Even a private disk is usually shared by a number of users. To satisfy the frequent need for input and output media devoted to your work alone, RSTS/E provides an assortment of assignable devices. DECtapes and magnetic tapes, for example, can be assigned by you for your own input or output; line printers and paper tape punches can be assigned for your output. Their assignability depends, of course, on their physical availability: obviously, if a DECtape or a line printer is being used, you cannot immediately assign it to yourself; should you try, the system will print an appropriate message at your terminal (?DEVICE NOT AVAILABLE, for example).

Within this class of assignable devices, there are degrees of accessibility. A DECtape, for example, with a single directory and no accounts, permits access to all the files it contains. A magnetic tape, on the other hand, which contains account numbers associated with its files, permits access to any

files if you know their account numbers. Paper tape punches and line printers contain no files and therefore impose no such restrictions on access. If nobody else is using a punch or printer, and if the system manager has not restricted its availability, you are free to assign it. And anyone may use an unoccupied terminal.

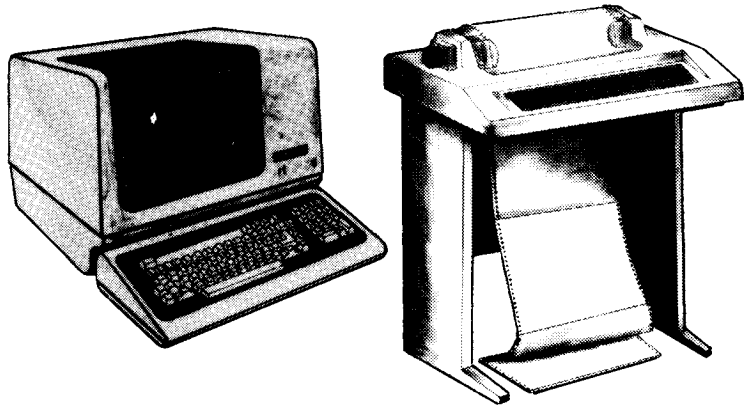
2.1.5 A List of RSTS/E Devices

The following list contains brief descriptions of devices available on RSTS/E systems, including their general functions, advantages, and disadvantages. Following the full name of each device are two items: the abbreviated name, or the specification, by which it is known to the system, and its I/O function (input and output, input only, or output only).

Devices are also classified as file-structured or non-file-structured; a file-structured device can store files, while a non-file-structured device cannot. However, a file-structured device can be treated by the system as non-file-structured. This capability allows you to bypass some of the limitations of file-structured devices.

The devices are listed according to degree of legibility: those specifically intended for human reading appear first; next come those which, with some effort, are humanly decipherable; finally come those which only the computer can read.

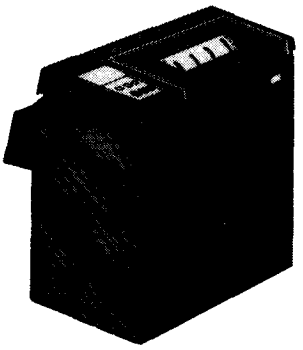
The terminal is a non-file-structured device. In addition to human readability, its most significant advantage is that it is interactive; that is, it allows you to communicate with the system and thus to control your job while it is running. Another advantage of the terminal is its operational similarity to a common off-line, non-file-structured, input-only device: the typewriter.



TERMINAL KB: or TT: or TI:
(input and output)

A disadvantage of the terminal is its inconvenience for output of large amounts of data. Compared with a line printer, a hard-copy terminal prints quite slowly. And though a video terminal may print rapidly, it may not produce a paper copy that you can remove and read at your own pace.

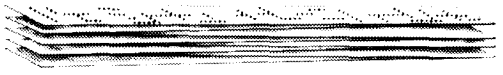
The line printer is a non-file-structured device. Like the terminal, it produces human-readable output, but at much greater speed. This speed is its most important advantage over the terminal; the line printer is the fastest available device for producing hard copies of information.



LINE PRINTER LP:
(output)

The line printer's disadvantages arise from its singular purpose: simply to output information in the form of a human-readable hard copy (a listing). Its output, therefore, unlike that of tapes and disks, cannot be read by the computer; there is no way to recycle one of its file listings through the system. Therefore, if the printed file has been deleted from the machine-readable device (disk, DECtape, and so forth) on which it resided, you have no way to edit it by computer. And, if that deleted file was a program, you cannot run it. Another disadvantage of the line printer is its inability to output more than one user's data concurrently, in the manner of the disk. Before assigning a busy line printer, you must wait for it to finish its current job.

Paper tape is a non-file-structured device. It has three advantages over other machine-readable media: it is inexpensive, easily shipped or mailed, and can be deciphered by a person who knows its punched code.



PAPER TAPE PR: and PP:
(input and output)

Most of paper tape's disadvantages are inherent in its physical makeup. Holes punched in paper cannot be erased or satisfactorily repaired: thus, a paper tape must always contain the same data; changing or editing requires a new tape. Also, paper is a low-density storage medium: a good deal of it is required to hold information in any form — even as tiny perforations. And paper is easily torn or creased.

In addition to its other disadvantages, paper tape requires two devices, one for input and one for output: namely, the reader and the punch. These are described below:

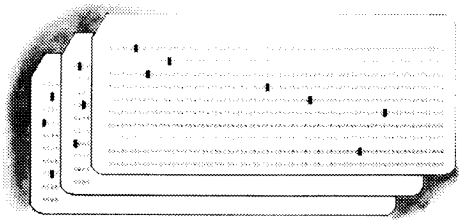
PAPER TAPE READER PR: (input)

The paper tape reader is a non-file-structured device. Its advantages are that it is compact and performs automatic input that is faster than user input from a terminal. Its input speed, however, is much slower than that of DECtape, magnetic tape, and disk.

PAPER TAPE PUNCH PP: (output)

The paper tape punch is a non-file-structured device. Its important advantages and disadvantages for output are the same as those of the paper tape reader for input.

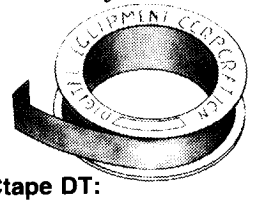
Cards are non-file-structured. Like paper tape, they are machine readable but humanly decipherable. Because they are inexpensive, easy to mail individually or in small quantities, and can be coded or annotated with a pencil, they are widely used to gather data for public and commercial operations: school registrations, consumer billings and surveys, and so on. Cards can be prepared off-line (by keypunch), thus conserving system time and resources.



CARDS CR: and CD:
(input and output)

The disadvantage of cards is their susceptibility to damage. Composed of card paper, they also share all the disadvantages of paper tape. For full usability, cards require two additional devices: the CARD READER (CR: and CD:) and CARD PUNCH.

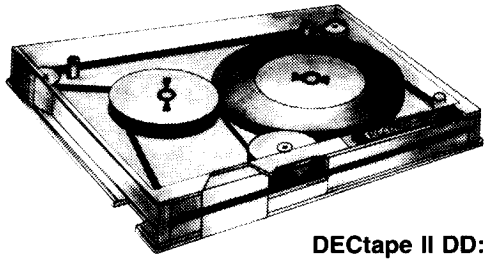
DECtape is a file-structured device. Its significant advantages over paper storage devices are its far greater I/O speed, its higher density of data storage, and its reusability. Unlike paper tape or cards, it can be erased, edited, and rewritten. And compared with the paper media, it involves less handling, because it requires only one additional device to fulfill its I/O capability: the DECtape drive, which performs both input and output. DECtape, because it has a directory structure, also allows you to change files in place, and therefore requires less manipulation by the system than does a magnetic tape. Because of its small size, DECtape is easily handled, carried, and stored. It is physically stronger and less sensitive to climate than cards, paper tape, and magnetic tape.



DECtape DT:
(input and output)

DECtape's small size presents one disadvantage: it cannot store as much data as a magnetic tape or disk. Also, its storage density is less than that of a disk. And, because DECtape is a magnetic storage device, it cannot be humanly decoded.

DECtape II is a non-file-structured device. Except for its non-file-structured property, it has the same advantages and disadvantages as the previously described DECtape. That is, I/O speed, high density of data storage, reusability, and in place file modification are all advantages of DECtape II. And, as with DECtape, DECtape II has the disadvantage of lower storage density compared to disk or magnetic tape.

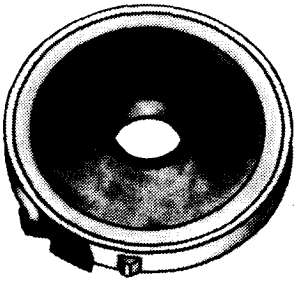


DECtape II DD:
(input and output)

Magnetic tape (also called magtape) is not a true file-structured device. Unlike DECtape, it has no directory; it does, however, contain an individual account associated with each file on the tape. It shares with DECtape the following advantages over paper storage devices: greater speed, higher

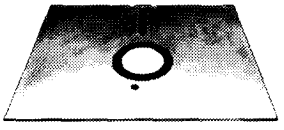
density, reusability, and less handling. Also, it requires only one other device for both input and output: the magnetic tape drive. The magnetic tape's size and shape make it convenient for storing files off-line.

Magnetic tape shares two disadvantages with DECtape: its speed and density are not as great as those of the disk. And, as noted, it is more susceptible to damage than DECtape, and requires more system manipulation. Also, magnetic tape is a sequential medium: to reach a specific file on the tape, the system must first read all the files preceding it. And a specific magnetic tape file cannot be deleted without also deleting all the files that follow it.



MAGNETIC TAPE: MT:, MS:, or MM:
(input and output)

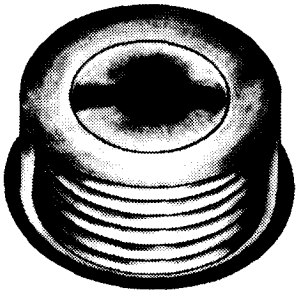
The flexible diskette is a non-file-structured device. Diskettes share with DECtape and magnetic tape the speed, density, reusability, and handling advantages that paper storage devices lack. The flexible diskette, also called a floppy disk, resembles a 45 RPM record. It consists of a thin, flexible surface on which data is recorded. This surface is encased in a plastic envelope with a slot for a read/write head and a drive spindle hole in the center. These physical characteristics make the diskette easy to store and ship.



FLEXIBLE DISKETTE DX: or DY:
(input and output)

Because data can be stored only on one side of the diskette, its storage capacity is less than that of disk, DECtape, or magnetic tape. Also, because it is a magnetic device, it is not humanly decodable. A flexible diskette drive is required to read and write the diskette.

The disk is a file-structured device. Of all devices, it is fastest, highest in density, largest, most reliable, and most durable. Like DECtape, flexible diskette, and magnetic tape, it is reusable. And it far surpasses those devices in the number of accounts and files it can hold. For input and output, the disk requires one additional device: the disk drive. The disk involves less human handling than any other device. For all these reasons, it is chosen for the RSTS/E public structure.



DISK SY:, DF:, DS:, DK:, DL:, DM:, DP:, DR:, or DB:
(input and output)

The disk's obvious disadvantages are its large size, heaviness, and high cost. These make it less practical for off-line storage, for travel, and for private ownership. And, because the disk is a magnetic device, it is not humanly decodable.

2.1.6 Device Names: Physical and Logical

Physical Device Names

In the foregoing list of RSTS/E devices, each device's specification, the name by which it is known to the system, appears beside the device's full

name. For DECtape, for example, the system's name is DT: or DD:. This specification, also called a physical name, is generally followed by a decimal unit number, which, in the case of a storage medium, is the number of the drive on which the medium is mounted. This number distinguishes devices of the same kind according to their physical locations on the system.

For example, three users may be simultaneously creating files on three DECtapes, physically named DT0:, DT1:, and DT2:. These physical device names separate the three DECtapes — from the point of view of each user and of the system.

To elaborate on this example, one of these three users, the current "owner" of DT0:, is creating on that device a text file named NANCY.TXT. From the system's point of view, that file's more specific name, or file specification, is DT0:NANCY.TXT — the file name and type, preceded by the physical name of the device on which the file resides. The standard colon (:) serves not only to identify the device name, but also to separate and to distinguish it from the file name.

A device name in a file specification performs the important function of identifying a file unequivocally. For example, assume that you are working on a file contained on DECtape whose specification is DT1:FIRST.BAS and that you have assigned the device DT1: (see Section 4.3.1). Furthermore, you already have on your account an existing file of the same name — not on DECtape but on the public disk structure, a resource you have not been using because you have been working on the DECtape instead. Now you close your DECtape file and return to the public structure, without giving up your ownership of the DECtape DT1: via the DEASSIGN command (see Section 4.3.2). Thus, although you are sharing the public disk structure with a number of current users, you still own the DECtape DT1:, because you have not revoked your original device assignment command, ASSIGN DT1:.

Sometime later, you wish to reopen and edit the file DT1:FIRST.BAS—that is, the FIRST.BAS that resides on your private DECtape, not the FIRST.BAS that resides on the public structure. If you were to forget the physical device name DT1: in specifying the file, and simply type FIRST.BAS, you would summon not the DECtape but the disk file, and with it ample possibility for confusion. The system retrieves the disk file FIRST.BAS because the system always assumes the public disk structure as the default device.

Protect yourself against this sort of confusion by being careful when specifying duplicate file names. The system, fortunately, cannot be so confused: it always remembers files not only by their names and types, but by their host devices as well. Thus, while you may have created, at various times, a number of files on devices other than public disks, the RSTS/E system would recognize these files by the following specifications: DT2:DATA3.REP, DT3:INDEX.001, MT1:ARTHUR.NAA (a magnetic tape file), and DK5:PRIVAT.GRP (a private disk file).

Logical Device Names

Any RSTS/E device can have, in addition to its physical name, an assigned logical name. You can give to a device a name of your own choosing. This name, like a file name, can contain from one to six alphanumeric characters, without embedded nulls, tabs or spaces, and including the standard colon separator (:). By using the ASSIGN command, you may, for example, assign the logical name INDEX: to DECtape DT3:. This logical name will be recognized by the system until you DEASSIGN the logical name.

Logical names are used to describe the nature of the information residing on a device, and therefore may be easier for you to remember than the physical name with its unit number.

But there are more specific reasons for logical names. The fundamental advantage of a logical name is that, unlike a physical name, it does not depend upon where (on what drive) the medium is mounted. Therefore, you can choose a logical name without regard to what device drives may be available at some future time. For example, if you write a BASIC–PLUS program that, at several points, accesses a specific magnetic tape for statistics, you can refer to this tape by the logical name STATS. The advantage of STATS over a physical name is this: if you were to specify a physical name such as MT1: for the magnetic tape, the success of your program, as written, would depend on the availability of tape drive unit 1 at the time the program runs. What if the drive were in use or inoperative at the time? If another drive, say unit 2, were available, you could use it, but would first have to edit your program accordingly, changing each occurrence of "MT1:" to "MT2:" — a tedious procedure. So by assigning the magnetic tape the logical name STATS before running the program, you ensure the system's recognition and access of that tape whether it is mounted on drive unit 1, 2, 3, and so forth.

A similar use of logical names is to enhance the efficiency of a batch job, an operation that does not require terminal interaction. Typically, such a job is "programmed" by one user for later execution by another. For instance, assume that you create a large batch job in the morning to be run by an operator at night, when there are fewer users on the system. The batch "program" or control file accesses a magnetic tape. If you refer to the tape by a logical name in the batch control file, and inform the operator of this name, the operator need not be concerned about the availability of a specific tape drive. Any drive will do, since the system recognizes the logical name once the operator assigns it.

Note that the system always recognizes and accepts a device's physical name, whether or not a logical name has been assigned to that device. Thus, you can specify a DECtape running on drive unit 2 and assigned the logical name STAR as DT2: or as STAR. Also, a physical device name preceeded by an underscore causes the system to access that device, regardless of any prior assignment. Some logical names, at the system manager's discretion, can be made system-wide: known to, and usable by, all other users. One such system-wide logical name is SY:, which designates the

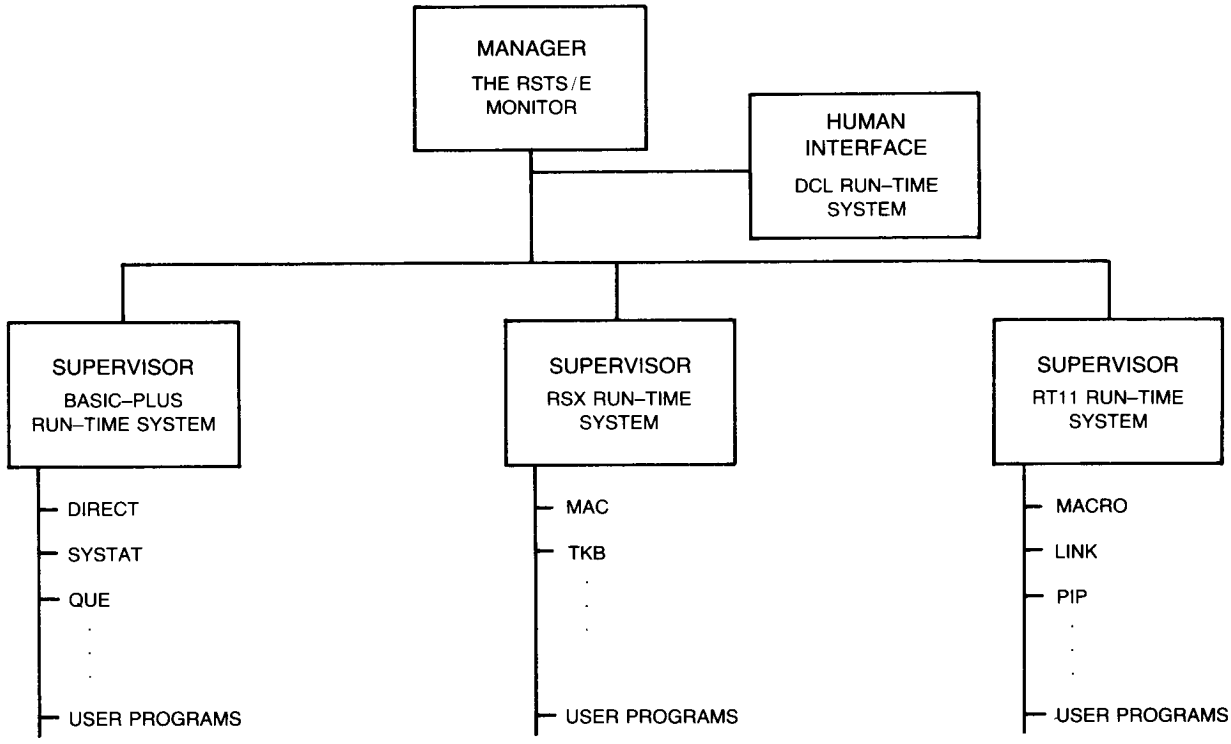
public disk structure. Since SY: is a logical name for a set of devices, it need not have a unit number, and would probably be confusing if it did. SY0:, however, is always the logical name for the system disk.

2.2 Software

We assume here that you are new to computers or at least new to the RSTS/E system. In either case, the remainder of this chapter will introduce you to some concepts of RSTS/E software. Detailed descriptions of RSTS/E software components appear later in this manual, and in the other RSTS/E manuals.

To begin, it may be useful to you to take your first look at the software available in terms of a human organization, such as a department in a company. An analogous "organization chart" is shown in Figure 2–2.

Figure 2–2: Software Organization on RSTS/E



As shown, the "manager" of the software organization is the monitor. It controls the activity for the rest of the programs making up the system. Section 2.2.1 describes the five functions of the monitor in more detail.

The run-time systems are the "supervisors" in our system organization. Each program you run on a RSTS/E system executes under control of a run-time system. They provide support needed, literally, at run time — such as a loader to load your program from disk into memory for execution (Section 2.2.2).

The remaining "worker" programs provide more direct help to users. Most of the help is in the area of program development. That is, most of the software that DIGITAL sells as a part of, or an adjunct to, a RSTS/E system is intended to help you write programs to do other work on the system. This type of software is called "systems software." It is provided to help you write what is called "applications software."

These "worker" programs — the systems software provided by DIGITAL — fall into some general categories that parallel the steps in program development. We discuss these steps, and the software available, in Sections 2.2.3 through 2.2.10.

2.2.1 The Monitor

The monitor is the heart of the software available on the system. It makes RSTS/E what it is — a resource-sharing and time-sharing system. The monitor's capabilities fall into five categories, discussed below.

1. Scheduling. This function is what makes RSTS/E a time-sharing system. The part of the monitor that handles scheduling oversees the jobs to be run, ensuring that each job gets its time-slice for execution in the processor. If a job needs more memory than is currently available, the scheduler handles what is called "swapping." This involves transferring jobs that are currently in memory to disk, and vice versa, so that each job gets its turn.

2. Memory Management. Computer memory is one of the critical resources that need to be shared by users. Regardless of how you use the system, you are executing a set of computer instructions — a program which must be in computer memory before it can be run. The memory management code in the monitor keeps track of where each job is in computer memory and flags those jobs eligible for "swapping" to disk when they are not being executed.

3. Input/Output Services. The part of the monitor that handles this function maintains directories for users on the public and private disk structures. It allocates the disk space needed by each job to store and retrieve data to and from disk. It maintains these areas according to "accounts"; that is, by project-programmer number. For example, as part of your work, you may create a "file" of instructions or data. The input/output services section of the monitor enters this file in a directory identified by your project-programmer number. Thus, the next time you log in, the file can be easily identified as yours by the project-programmer number in the directory entry.

4. Device Management. This part of the monitor helps you get control of and use the various devices available on your system. The software consists of programs called "drivers"; this code transfers data between the devices and computer memory. This data transfer can occur independently of computer processing, so the device management code keeps information that is useful to the scheduler. For example, if you are typing data at your terminal, the terminal driver can simply buffer

this information — store it in what are called "small buffers" — until you type some terminating character such as a RETURN key. The processor can be doing other work while this is going on.

5. Utility Functions. These are a wide variety of general-purpose functions performed by the monitor. For example, consider the device "nickname" LB: for "library" files. Users can also specify their own such names — called user logical names to distinguish them from the physical names such as MT: or DK:. This part of the monitor keeps track of logical names (who is using which names, by project-programmer number), and the devices the logical names refer to.

   Many of the utility functions provided are available to BASIC–PLUS and MACRO programmers, as described in the *RSTS/E Programming Manual* and the *RSTS/E System Directives Manual.*

2.2.2 The Run-Time Systems

Run-time systems are the "supervisors" in our analogy. They provide an environment for either people or programs to work in, and sometimes both.

Environments for People

In Chapter 1, we talked about command language environments. The part of a run-time system that analyzes user-typed commands is called a "keyboard monitor." Four of the run-time systems available with RSTS/E include a keyboard monitor; these are the BASIC–PLUS, DCL, RSX, and RT11.

The system manager at your site has selected one of these as your "default keyboard monitor." This determines the prompt you see after you log in, as described in Chapter 1. You can also switch to a different keyboard monitor. The program that lets you do this is called SWITCH. The keyboard monitor you switch to is called your "job keyboard monitor" if it is different than the default keyboard monitor. We give an overview of the commands available with each keyboard monitor in Chapter 3. The BASIC–PLUS commands are described in detail in the *BASIC–PLUS Language Manual*; DCL commands are described in the *RSTS/E DCL User's Guide.*

Environments for Programs

The BASIC–PLUS, RSX, and RT11 run-time systems also provide an environment for programs. These run-time systems include one or more of the following:

1. Loader. All of the run-time systems mentioned above provide code that loads a program from disk into memory and starts its execution.

2. Interpreter. The BASIC–PLUS run-time system, for example, executes any BASIC–PLUS program by interpreting an internal representation (called "translated code") of the program's source statements.

3. Object Library. Some run-time systems include a library of routines for programs.

4. Emulator. The RSX and RT11 run-time systems include code that emulates directives handled by DIGITAL's RSX–11M and RT–11 Operating Systems for the PDP–11 computer. These directives can be used by MACRO assembly language programs on RSTS/E systems. The directives are described in detail in the *RSTS/E System Directives Manual.*

Note that when you type a command that runs a program, the run-time system that the program runs under may be different than the run-time system that processed your command. When you type a RUN command, the keyboard monitor code in the run-time system passes control to the RSTS/E monitor. The monitor determines which run-time system the program should be run under, and passes control to the "loader" in that run-time system. After the program is loaded and executed, control again passes back to the monitor, which returns control to your job keyboard monitor.

Thus, you need not worry about being "in" the run-time system for the program you want to run. The system handles this for you.

The BASIC–PLUS Interpreter

The BASIC–PLUS run-time system provides an interesting feature where the distinctions we have made between environments for people and for programs are somewhat blurry. This feature is called an "interpreter."

Perhaps the best way to explain what the interpreter does is by example. Suppose that you are "in" the BASIC–PLUS keyboard monitor (you see a "Ready" prompt) and you type the following statement at your terminal.

```
PRINT 2+2
```

The PRINT statement is part of the language recognized by BASIC–PLUS. BASIC–PLUS translates the statement into its own internal language and executes code that displays the result — 4 — immediately. If you typed a line number in front of the statement, BASIC–PLUS would translate the statement and store it for later execution by the interpreter. You can save a series of statements as a complete program for later execution. If you made an error that the interpreter recognized, it would display an error message at once.

As described in later sections, there are other languages with which you can write programs on RSTS/E systems, but they do not use an interpreter. You use different kinds of tools, called editors, compilers or assemblers, and linkers to write and translate your program to a form that can be executed by the computer.

The advantage of an interpreter is that it can be highly interactive. Rather like a human interpreter, BASIC–PLUS allows you to carry on a "conversation" with the computer. You see the results — either the result of computation or an error message — at once.

2.2.3 Writing the Source Program: Help from a Text Editor

To code a program, you choose a programming language suited to your particular problem and write down the steps needed to solve your problem in this language. The tool available to help with the process of "writing down" is called a text editor.

As the name implies, a text editor allows you to create and edit text; the text you type at your terminal is stored on disk in a machine-readable format called ASCII.* You can also use the text editor to insert more text, delete existing text, move text from one place in your program to another, and any of the general processes that fall in the category of "editing."

Once you are satisfied with the text, you can store it as a "file" on disk. You name a file, and can thus refer to it again. In this form, your code is called a "source program": the source, or the beginning, of the solution to the problem.

There are several text editors available for RSTS/E systems. The easiest to use is EDT; it is described in the *EDT Editor Manual.* You can type commands to EDT to work with a file of text, or — if you have a DIGITAL VT100 or VT52 terminal — you can use EDT in "screen mode." In screen mode, your text file is displayed on the terminal screen and you move the cursor on your terminal screen to places where you wish to insert or modify text.

Another editor provided with RSTS/E is called TECO. TECO is a powerful text editing program; however, its commands are more complex. DIGITAL does not support the TECO software. The main implication of non-support is that if you find an error in the software, DIGITAL assumes no responsibility to fix it. However, TECO can be very useful; you can even do programming with it. (TECO is another example of a run-time system containing an interpeter.) TECO is described in the *PDP–11 TECO User's Guide.*

Another program that may be useful to you when preparing text is RUNOFF. RUNOFF is a text formatter; you may find it helpful if you are using a text editor to prepare text such as letters or documents. RUNOFF has commands that let you specify left and right margins, justify text, specify spacing between and indentation of paragraphs, and so forth. It is described in the *RSTS/E RUNOFF User's Guide.*

---

* ASCII stands for "American Standard Code for Information Interchange." This code is one of several that fall into the category "machine-readable." ASCII is the code used by the PDP–11 computer.

2.2.4 Translating to an Object Program: Help from a Compiler or Assembler

In the previous section, we mentioned choosing a computer language to write your source program. Besides the BASIC–PLUS interpreter, the software that provides these languages comes in two general categories: compilers and assemblers. Both take your source program and convert it to the machine language that can be recognized and executed by the processor. The difference is twofold, involving the intent of the language and its relationship to machine language.

Compilers provide a language that is suited to solving general types of problems. For example, FORTRAN, which stands for FORmula TRANslation, is a language suited to solving mathematical or scientific problems. COBOL, which stands for Common Business Oriented Language, is a language suited for solving business problems. An assembler provides a language that is harder for people to use and understand. It simply provides mnemonics — short codes — for the instructions recognized and executed by the processor. The advantage to using an assembly language lies in its flexibility; you can write code that can be executed very efficiently by the computer.

In any case, both compilers and assemblers provide one service: translation from a human-understandable language to the instructions that can be recognized and executed by the computer. You compile or assemble your source program to form what is called an "object program." Again, the result of the process is a "file" — a unit that can be stored and retrieved for the next step in the process.

One assembly language is available for PDP–11 computers, called MACRO. On RSTS/E, two MACRO assembler programs are available, however. One, called MACRO, translates assembly-language programs to code that will run under the RT11 run-time system. Another assembler, called MAC, translates assembly-language programs to code that will run under the RSX run-time system. The MACRO language is described in the *MACRO Language Reference Manual.* Information about running MAC on RSTS/E systems is given in the *RSTS/E Programmer's Utilities Manual.* Using MACRO is described in the *RSTS/E RT11 Utilities Manual.* Additional information about MACRO programming is given in the *RSTS/E System Directives Manual* and the *RMS–11 MACRO Programmer's Reference Manual.*

BASIC–PLUS is another language available on RSTS/E systems, although it is handled by an interpreter, rather than a compiler, as described in Section 2.2.2. The BASIC–PLUS language is a version of BASIC; it is described in the *BASIC–PLUS Language Manual.* Other information useful to BASIC–PLUS programmers is given in the *RSTS/E Programming Manual.*

Other languages are optionally available; they may or may not be on your system. These include:

BASIC–PLUS–2 This language is similar to BASIC–PLUS; the main difference is that it is translated by a compiler rather than an interpreter. This has its advantages and disadvantages. A BASIC–PLUS–2 program will run somewhat faster than a BASIC–PLUS program. This is due to the fact that, once translated, the BASIC–PLUS–2 program is machine-language code, rather than the intermediate code that must be processed by the BASIC–PLUS interpreter every time the program is executed. On the other hand, BASIC–PLUS–2 does not function interactively. Unlike BASIC–PLUS, you cannot type a language statement and have it executed (and analyzed for errors) immediately.

The BASIC–PLUS–2 language is described in the *BASIC–PLUS–2 Language Reference Manual*; information on how to use the compiler on RSTS/E systems is given in the *RSTS/E BASIC–PLUS–2 User's Guide.*

FORTRAN–IV As mentioned above, the FORTRAN language is widely used in the solution of mathematical or scientific problems. The FORTRAN–IV compiler translates programs to be executed under the RT11 run-time system. The FORTRAN–IV language is described in the *PDP–11 FORTRAN Language Reference Manual.* Information about how to use the compiler on RSTS/E systems is given in the *RSTS/E RT11 Utilities Manual.*

FORTRAN–77 The FORTRAN language is also available to RSTS/E users through the FORTRAN–77 compiler. It translates programs to be executed under the RSX run-time system. Again, the language is described in the *PDP–11 FORTRAN Language Reference Manual.* Information about how to use the compiler on RSTS/E systems is given in the *FORTRAN–77 User's Guide.*

COBOL–81 The COBOL language is widely used in solving business problems. The COBOL–81 compiler translates programs that will run under the RSX run-time system. COBOL–81 also has a Language Manual and User's Guide.

DIBOL DIBOL stands for DIGITAL Business Oriented Language. The DIBOL language is described in the *DIBOL Language Reference Manual.* Information about how to use the compiler on RSTS/E systems is given in the *DIBOL User's Guide.* The DIBOL compiler translates programs to run under the RSX run-time system.

2.2.5 Building an Executable Program: Help from a Linker

All of the compilers and assemblers listed above produce object code. The object code needs final touches from a linker before it can be executed, however.*

The reason is that you seldom write a program as one unit. It is easier to work with programs that are written as modules — programs and subprograms — that you can separately design, code, debug, and maintain. Even if you code your program as one unit (one main program with no separately assembled or compiled subprograms), every language compiler translates some source statements or instructions as calls to subroutines kept in libraries. For example, all the compilers generate calls to library subprograms to perform I/O or do mathematical calculations.

Linkers combine these separate modules, resolving references to addresses (translated program and subprogram names, statement labels, and variable names) that cross module boundaries.

Another necessary service that linkers provide is a means to construct overlays. The amount of memory from which programs can be executed is limited on PDP–11 computers to around 32,000 words. On RSTS/E systems, there are further limitations: the run-time systems take space in this 32,000–word area, for example.

If your program is too large to fit in the space available, you must specify how you want it overlaid — such that sections of code and data can be called into memory at different times (the new sections "overlaying" the old).

There are two linkers available on RSTS/E systems: the Task Builder (TKB) and LINK. The Task Builder links programs to run under the RSX run-time system and its derivatives. In other words, if you used MAC, BASIC–PLUS–2, COBOL–81, FORTRAN–77, or DIBOL, you use the Task Builder to link your programs. The RSTS/E Task Builder is described in the *RSTS/E Task Builder Reference Manual.*

LINK works with programs to run under the RT11 run-time system. That is, if you used MACRO or FORTRAN–IV, you use LINK. LINK is described in the *RSTS/E RT11 Utilities Manual.*

A linker is not necessary for BASIC–PLUS programs; the BASIC–PLUS run-time system provides the support that other languages provide in the form of libraries of subroutines. And, instead of using overlays, you "chain" from one program to another, as described in the *RSTS/E BASIC–PLUS Language Manual.*

---

* The code produced by BASIC–PLUS is executed by the interpreter. Thus, you do not need to link BASIC–PLUS programs.

2.2.6 Finding Errors: Help from Debugging Aids

There are two programs to help you find and correct errors in your program, both called ODT, for Octal Debugging Tool. They are similar in one respect: you work with instructions and data in octal. So, you would use these tools only if you are familiar with "machine language"; that is, with computer instructions and data in the form of octal numbers. If you are, the ODTs can be very handy for debugging. You can change instructions and data without reassembling or recompiling your program.

One ODT is a program; you type RUN $ODT to access this Octal Debugging Tool. With it, you can display, in octal, the contents of locations in memory, a file, or a device. You can also change the value of these locations. This ODT is described in the *RSTS/E System Manager's Guide.*

The other ODT is an object code file that you link into your program under the RSX run-time system. Then, when the program is run, you can examine and change the contents of the program, similar to the first ODT we described. The main difference is that you can set what are called "breakpoints" in your program. When you set breakpoints, you define locations where you want execution to stop. You can thus "step through" your program, examining and changing values within the program as you go. This "stepping through" can make it easier to pinpoint where an error is occurring in your program. You can then change instructions and data, and test the change without recompiling or reassembling. This ODT is described in the *IAS/RSX–11 ODT Reference Manual*, currently included as a part of RSTS/E documentation set.

If you use the BASIC–PLUS language, you will find many aids to debugging within the language itself. For example, it contains a STOP statement that allows you to set breakpoints in your program. You can then use the PRINT statement to display values while the program is stopped, and other language statements to change the values if you wish. In addition, there is a utility program available for cross-reference listings, to help you see where each variable value in your program is used. See the *BASIC–PLUS Language Manual* for further discussion of these features.

2.2.7 Making Routines Available: Help from Librarians

If you write code that other people can use in their programs, you may want to make it generally available on your system by putting it in a "library." A library is a collection of compiled or assembled (object code) modules that can be linked with other programs. The object-code modules provide general-purpose capabilities that people would otherwise have to code over and over. To prevent such "reinventing the wheel," RSTS/E both provides libraries and allows users to create their own libraries.

There are two methods for creating libraries on RSTS/E systems, relating again to the two run-time systems on RSTS/E that make it possible to code programs in languages other than BASIC–PLUS: RSX and RT11. The two librarians are called LBR (for RSX–based code) and LIBR (for RT11–based

code). Both librarians organize separate files of object code into object library files, so that individual modules can be easily located and linked into the programs that call the library routines.

Thus, if your code is written in MACRO (using the MAC assembler), BASIC–PLUS–2, COBOL, DIBOL, or FORTRAN–77, you would use the LBR librarian to make it available to others. LBR is described in the *RSTS/E Programmer's Utilities Manual.*

If you use MACRO (with the MACRO assembler) or FORTRAN–IV, you would use the LIBR librarian to make it available to others. LIBR is described in the *RSTS/E RT11 Utilities Manual.*

BASIC–PLUS programs do not need a librarian because there is no linking process with BASIC–PLUS programs. Hence, there is no need to organize routines such that a linker can easily find them. As mentioned in the previous program, you chain to other programs in BASIC–PLUS, simply referring to them by name.

2.2.8 Work While You Go Home: Help from the BATCH Processor

As mentioned before, RSTS/E is a time-sharing system that allows many users to do work at the same time. Some types of work done on computers is repetitive, however, and does not really require that you pay much attention while it is being done. For example, if you are writing computer programs with compiler languages, you will go through many repetitions of the compile-and-link process as your program reaches completion.

With the RSTS/E BATCH processor, many users can submit requests to handle such work. The term "batch" evokes the image of an avid worker assigned a batch of work that is accomplished by working through the batch, one piece at a time. This is just what happens: BATCH runs as a separate job on RSTS/E systems. You can submit work to be done, and go on to other work at your terminal, or submit work to be done overnight.

Chapter 7 of this manual describes the BATCH processor and tells how you submit work to it.

2.2.9 General-Purpose Tools: The Utilities

Three sets of general-purpose programs called utilities are provided with RSTS/E systems to help you with your work. One set, of universal use on RSTS/E systems, is described in Chapter 6 of this manual. These programs deal with the display and transfer of information; one example is the SYSTAT program introduced in Chapter 1, and described further in Chapter 6.

Most of the other utility programs described in Chapter 6, both those that transfer and those that display information, work with files. If you are not familiar with the term "file" as it relates to computer systems, its meaning is very similar to the general one. Most programs place the information

they generate on storage devices in units called files, just as you would put work relating to a particular topic in a file in your desk. When you create a program with EDT, for example, EDT produces a file containing the statements you type. When you compile or assemble a source program, the compiler or assembler produces a file with the object code output.

The files you create are stored in your "account" — identified by project-programmer number — usually on the public disk structure. With the general-purpose utility PIP (Peripheral Interchange Program) you can copy files you create from disk to tape or some other storage media, and vice versa. You can use the DIRECT utility to list the names and sizes of files.

The introduction to Chapter 6 gives a complete overview of the general-purpose utility programs available on RSTS/E systems. The utilities are described in alphabetical order in remaining sections in Chapter 6.

The other two sets of utility programs provide help with coding programs to run in the RSX or RT11 environments. Information needed to run the MAC assembler and the LBR librarian (for programs that run in the RSX environment) is given in the *RSTS/E Programmer's Utilities Manual.* This manual also describes a program called PAT that lets you "patch" object modules. You can make changes to your program by creating an assembly-language patch file with the desired change. You then assemble this file and input it, along with the original file, to PAT, which produces the changed object code.

The *RSTS/E RT11 Utilities Manual* describes similar programs for the RT11 program environment: the MACRO assembler, the LINK linker, the LIBR librarian, and a patch program also called PAT.

Chapter 3
More About Command Environments

As mentioned in Chapter 1, RSTS/E offers four command environments for you to work in: DCL, BASIC–PLUS, RSX, and RT11. Each offers its own advantages. The system manager at your site has selected a default keyboard monitor (Section 2.2.2) that determines the prompt that you see — and the corresponding command environment you are in — when you first log in. The prompts and their corresponding environments are:

$	DIGITAL Command Language (DCL) environment
Ready	BASIC–PLUS environment
>	RSX environment
.	RT11 environment

You can switch to another if you like, however. Section 3.1 describes how you switch from one command environment to another.

Sections 3.2 through 3.5 give an overview of the commands available in the DCL, BASIC–PLUS, RSX, and RT11 environments to help you choose. Section 3.6 describes another command facility available on RSTS/E systems: the Concise Command Language (CCL) facility.

Table 3–1 lists the commands available in each environment.

Table 3–1: Commands by Environment

System-Wide Commands	HELLO	RUN		BYE
CCL's (also system wide)	As determined by your system manager; the following are typical: MOUNT	ATTACH	PIP	HELP DCL etc.
Keyboard Monitor Commands (environment-dependent)	DCL	BASIC–PLUS	RT11	RSX
		COMMON COMMANDS		
	ALLOCATE ASSIGN APPEND BASIC CCL	ASSIGN DEASSIGN EXIT MOUNT REASSIGN	ASSIGN DEASSIGN EXIT MOUNT REASSIGN	ASSIGN DEASSIGN EXIT MOUNT REASSIGN
	COBOL	**SPECIFIC COMMANDS**		
	COPY CREATE DEALLOCATE DEASSIGN DELETE DIBOL DIFFERENCES DIRECTORY EDIT $EOD $EOJ FORTRAN HELP INITIALIZE LINK LOGOUT MACRO MOUNT PRINT RENAME REQUEST SET SHOW SUBMIT TYPE	APPEND CATALOG CCONT COMPILE CONT DELETE EXIT KEY LENGTH LIST LISTNH NEW OLD RENAME RUNNH SAVE SCALE TAPE UNSAVE Plus all immediate-mode statements	B/E/D CCONTINUE CLEAN CONTINUE CLOSE DATE ERROR GET INITIALIZE LIB LOCK MONITOR PPN R REENTER RN RU SHUTUP SIZE START TIME UNLOCK VERSION	DISMOUNT UNSAVE SHUTUP

3.1 Switching Between Command Environments

The system manager at your site has selected a default keyboard monitor, based on what he or she thinks will be useful to the most people at your site. You can switch to another, however. A command sequence that will work from any command environment, is:

```
RUN $SWITCH
Keyboard Monitor to switch to? Kbmon<RET>
```

You type RUN $SWITCH. The SWITCH program prints the question, and you respond with "kbmon" any of the following keyboard monitor names:

```
DCL
BASIC
RT11
RSX
```

The system then transfers control to that keyboard monitor, establishing it as your "job keyboard monitor." That is, you can type commands to the keyboard monitor you switched to until you switch again, or log off the system.

The SWITCH program is also documented in Chapter 6.

3.2 DCL Commands

As the commands below illustrate, the DCL (Digital Command Language) environment provides a set of useful commands that are easy to understand and use. For a complete discussion of the DCL command syntax and all the commands available, see the *RSTS/E DCL User's Guide.*

ALLOCATE

Reserves a device for your use alone. The device cannot be used by anybody else until you issue a DEALLOCATE command or log off. You can also specify a logical name by which you want to refer to a device.

ASSIGN

Assigns a logical name to a device, a project-programmer number, or both.

APPEND

Appends one or more files to another file.

BASIC

Allows you to enter the keyboard monitor for BASIC–PLUS.

CCL

Lets you use a RSTS/E Concise Command Language (CCL) command in the DCL environment.

COBOL

Compiles a COBOL–81 source program file, producing an object file.

COPY

Copies files. You can copy files between devices, between accounts (with limitations), or between systems (if your system is part of a network connected by DECnet).

CREATE

Creates a file and then fills it with data that you type at your terminal.

DEALLOCATE

Releases a device that you have previously reserved with ALLOCATE.

DEASSIGN

Deassigns a previously assigned logical name.

DELETE

Lets you delete files from disk and DECtape devices. For systems that use DECnet, you can also delete files at a remote system, provided you have access to the files on that system. Variations of the command allow you to delete jobs submitted for print or batch processing with the PRINT or SUBMIT commands.

DIBOL

Compiles a DIBOL source language file, producing an object code file.

DIFFERENCES

Compares two text files and produces an output file that lists the information that appears in one file but not in the other.

DIRECTORY

Displays the names, sizes, and other characteristics of files.

DISMOUNT

Informs the system that you are about to physically dismount a tape or disk.

EDIT

Lets you create a text file or edit an existing text file. EDIT will use either the EDT or TECO editor.

$EOD

Used in a batch stream (batch control file) to indicate "end of data".

$EOJ

Used in a batch stream (batch control file) to indicate "end of job". It logs your job off the system.

FORTRAN

Compiles a FORTRAN source program file, producing an object code file. Options allow you to select either the FORTRAN–IV or FORTRAN–77 compiler.

HELP

Displays helpful information about DCL commands at your terminal.

INITIALIZE

Zeroes a magnetic tape, to prepare it for writing new information to the tape.

LINK

Produces an executable program file, using object files you specify as input. The LINK command options identify what source language you used, and an overlay structure, if desired.

LOGOUT

Logs you off the system.

MACRO

Assembles a MACRO–language source program file, producing an object code file. Options allow you to select either the MAC assembler (for a program to run under the RSX run-time system) or the MACRO assembler (for a program to run under the RT11 run-time system).

MOUNT

Informs the system that a disk or magnetic tape has been physically mounted.

PRINT

Prints files on the line printer.

RENAME

Assigns a new name to a file.

REQUEST

Displays a message that you type at the operator's console.

RUN

Executes a program that you specify.

SET HOST

If your system has DECnet, you can use the SET HOST command to log in to another node in the network.

SET QUEUE

Allows you to change the status of a file that is queued to a line printer or for the BATCH processor.

SET TERMINAL

Lets you set certain parameters for your terminal.

SHOW

Variations of this command allow you to display current settings for your terminal, the status of the system, and other information.

SUBMIT

Submits a file to be executed as a batch job.

TYPE

Displays a file at your terminal.

3.3 BASIC–PLUS Keyboard Monitor Commands

The BASIC–PLUS command environment provides commands useful to the BASIC–PLUS programmer on RSTS/E systems. As mentioned in Chapter 2, the BASIC–PLUS interpreter lets you type in a program using BASIC–PLUS statements that are translated and stored in your job area. Many of the BASIC–PLUS commands (which are distinct from language statements) let you list, compile, store, and otherwise work with the BASIC–PLUS program you create. Note, however, that BASIC–PLUS has a very powerful feature (called "immediate mode") that allows you to type many language statements as though they were commands.

The BASIC–PLUS commands are listed below. For a complete description, see the BASIC–PLUS Language Manual.

APPEND

Includes the contents of a previously saved source program into the current program.

CAT or CATALOG

Displays the file directory for your account. Unless another device is specified following the term CAT or CATALOG, the public disk structure is the assumed device.

CCONT

Can be used by privileged users only. The command operates the same as the CONT command, but detaches the job from the terminal.

COMPILE

Allows you to store a translated version of your BASIC program. The file is stored on disk with the current name and the file type .BAC. Or, you can specify a new file name and type.

CONT

Allows you to continue execution of the program currently in memory following the execution of a STOP statement.

DELETE

Allows you to remove one or more lines from the program currently in memory. After the word DELETE, type the line number of the single line to be deleted or two line numbers separated by a dash (–) indicating the first and last line of the section of code to be removed. Several single lines or line sections can be indicated by separating the line numbers, or line number pairs, with a comma.

EXIT

Returns you to the job keyboard monitor.

KEY

Reenables the echo feature on the terminal after a TAPE command. Enter with the LINE FEED or ESCAPE key.

LENGTH

Displays the length of the current program in memory, in 1K increments.

LIST

Displays the program currently in memory at your terminal. You can list the entire program currently in memory, or one or more lines of that program.

LISTNH

Does the same thing as LIST, except that it does not print a header containing the program name and current date.

NEW

Clears your area in memory and allows you to input a new program from the terminal. You indicate a program name following NEW or when the system requests it.

OLD

Clears your area in memory and allows you to recall a saved program from a storage device. You indicate a program name following the word OLD or when BASIC–PLUS requests it.

RENAME

Lets you change the name of the program currently in memory.

RUN

Lets you begin execution of the program currently in memory. Or, if the RUN command is followed by a file specification, including device and account if desired, you can run any program.

RUNNH

Executes the program currently in memory, but does not print header information with the program name and current date.

SAVE

Saves the source program currently in memory in a file on the system disk, under its current name with the file type of .BAS. Or, you can give a complete file specification (including device and account).

SCALE

Sets the scale factor to a designated value or displays the values currently in effect if no value is designated.

TAPE

Disables the echo feature on the terminal; this is useful if you are using a terminal with a paper-tape reader.

UNSAVE

Deletes the file specified.

3.4 RT11 Keyboard Monitor Commands

The RT11 Run-Time System has a keyboard monitor. If you switch control to RT11, a dot (.) prompt is displayed, and you can type commands. The standard RSTS/E commands ASSIGN, BYE, DEASSIGN, HELLO, and MOUNT can be used. In addition, the RT11 keyboard monitor provides the commands described below.

B/E/D

The commands B, E, and D let you examine memory locations.

B <address>

sets a base address from which offsets in E and D commands are computed. Initially, the base is 0.

E <offset>

prints the contents of location <address> + <offset>.

D <offset> <value>

replaces the contents of location <address> + <offset> with <value>.

Any address from 0 to 177777 is valid for these commands; however, addresses outside your current job image area (as determined by the SIZE command) will give an error.

CCO[NTINUE]

Resumes the execution, as a detached job, of a program that was stopped by a CTRL/C. (Privileged users only.) If, for any reason, the program cannot be restarted, the message "?Can't continue" is displayed on the terminal.

CLEAN

Rebuilds the SAT (Storage Allocation Table) of a disk pack; functions the same as the CLEAN command of UTILTY (see the *RSTS/E System Manager's Guide*). The CLEAN command can be used only by privileged users; the format is:

CLEAN dev:

where dev: is the device designation of the disk.

CO[NTINUE]

Resumes execution of a program stopped by a CTRL/C. If, for any reason, the program cannot be restarted, the message "?Can't continue" is displayed on the terminal.

CL[OSE]

Closes all open channels. Tentative files are made permanent.

DATE

Displays the current date on the job's terminal.

ER[ROR]

Prints the RSTS/E error message text associated with the last encountered error, if there was one. For example, any failure on file lookup will give a message similar to "?Fil not fnd?". Typing CTRL/C to return to RT11's dot prompt (.) and then typing ERR will print the full RSTS/E error message for the error encountered; for example, "Can't find file or account".

GE[T]

Loads an RT11–executable file (with a file type of .SAV) into memory, but does not execute it. (The START command is used to run programs loaded with GET.) If no project-programmer number is specified, the program is assumed to be in the user's account. No automatic expansion of the job image area occurs with GET; you must use SIZE if necessary.

Loading for GET differs from R or RUN in that certain low-core areas are preserved rather than loaded from block 0 of the .SAV file.

IN[ITIALIZE]

Resets all RT11 conditions, except for logical assignments, to the state they are in when RT11 is first entered as the job keyboard monitor.

LIB

The command:

LIB [proj,prog]

sets the project-programmer number specified as the account to be searched with an "R" command. The default is [1,2]. This account is also scanned if a file lookup (see .LOOKUP directive, *RSTS/E System Directives Manual*) cannot find the specified file on the current account. Typing LIB without a project-programmer number disables the file lookup scanning in a library account.

LOCK

Denies access to a disk pack by nonprivileged users; that is, programs running under nonprivileged accounts cannot open any files on the disk. The command is available only to privileged users; it works the same as the LOCK command in UTILTY (*RSTS/E System Manager's Guide*). The format is:

LOCK dev:

where dev: is the device designation for the disk.

MO[NITOR]

Exits to the job keyboard monitor. This will be the default keyboard monitor unless you have run the SWITCH utility to set a job keyboard monitor.

PPN

The command

PPN [proj,prog]

sets an account to be searched for files opened for input under the RT11 run-time system (see .LOOKUP directive, *RSTS/E System Directives Manual.*) If a file specification is given with no project-programmer number, and none is specified at offset PPN in the scratch pad area (see the *RSTS/E System Directives Manual*) the RT11 run-time system first checks the current user account for the file. If this fails, it then searches the account specified with a PPN command. If this fails, it then searches the library (either the system library, or as set by the LIB command). If no PPN command has been given for this job, or if PPN search has been disabled by issuing PPN without a project-programmer number, only the user's account and the library are searched. If LIB has been typed without a project-programmer number, then only the user's account is searched.

R

Runs a program from the library account (either the system library or one defined with the LIB command). The file can be associated with any run-time system. For example, R LINK has the same effect as RUN $LINK.

The R command will automatically expand the user job image area, if necessary.

RE[ENTER]

Reenters or restarts a program already in memory whose execution has been halted. Bit 13 in the Job Status Word (see the *RSTS/E System Directives Manual*) must be set to allow a restart. If it is set, execution begins at the address specified by the contents of word 40 in the low 1000 bytes of virtual memory. All channels are reset (see .SRESET directive, *RSTS/E System Directives Manual*) unless the chain bit (bit 8 of the Job Status Word) is set. If restart is not possible, the message "?No restart" is displayed on the terminal.

RN

Loads the file named (from the user's account if no project-programmer number is given) and begins execution. The program may be associated with any run-time system. If necessary, the user job image area is automatically expanded for the program to be loaded and run. The RN and R commands are identical except for the accounts searched for the file when no explicit project-programmer number is given.

RU

RU is the same as a GET followed by a START. You must use SIZE to expand the user job image area, if necessary; no automatic expansion occurs with RU.

RUN

This command is passed on by the RT11 run-time system to the RSTS/E monitor — it works the same in any command environment.

SH[UTUP]

Only privileged users can execute this command. If (1) one and only one job is running on the system, (2) logins are disabled, (3) no disks except the system disk are mounted, and (4) no files are open on the system disk, then the SHUTUP command logs the current job off the system and bootstraps the initialization code after the job is logged off.

SI[ZE]

Sets the size available for an RT11 program, in K words. When the RT11 run-time system is entered, the size is set to 2K words. To see the current size, type SIZE without a value.

You must use SIZE to insure enough memory before typing a GET or RU command. The job image area is automatically expanded for R and RN commands, from information carried in the file itself. However, a size given with SIZE overrides any computed value (for R and RN) if the computed value is smaller than the SIZE value.

ST[ART]

Starts a program loaded with GET. The format of the command is:

ST[ART] [addr]

where addr is an even octal number specifying the address where program execution is to begin. If addr is omitted or 0, execution begins with the address in location 40 in the low 1000 bytes of memory.

TI[ME]

Displays the current time of day at the job's terminal.

UNLOCK

Unlocks a disk pack, so that programs running under nonprivileged accounts can open files on the disk. Works the same as the UNLOCK command of UTILTY (*RSTS/E System Manager's Guide*), and can be used only by privileged users. The format is:

UNLOCK dev:

where dev: is the device designation of the disk to be unlocked.

VE[RSION] VERSION command, RT11

Prints the current emulator version number.

3.5 RSX Keyboard Monitor Commands

The RSX run-time system has a keyboard monitor. In addition to the standard RSTS/E commands ASSIGN, BYE, DEASSIGN, MOUNT, REASSIGN, and RUN, the RSX keyboard monitor accepts and processes the commands listed below.

DISMOUNT

The DISMOUNT command prepares a device for dismounting, so that you can remove a disk pack or magnetic tape from a device. The form of the command is:

DISMOUNT dev:

UNSAVE

The UNSAVE command deletes a specifically named file from your account. The format of the command is:

UNSAVE filename.type

(The RSTS/E file specification is described further in Chapter 4.)

SHUTUP

Only privileged users can execute this command. It shuts down the system if no other jobs are running on the system, all disks are dismounted, all other run-time systems are removed, and no logins are allowed.

3.6 CCL Commands

RSTS/E allows you to run certain programs (typically, the system library programs described in Chapter 6) by typing a unique system command called a Concise Command Language (CCL) command. The number of programs that can be run by CCL commands is decided by the system manager, although DIGITAL initially supplies CCL commands for some programs.

The chief advantage of CCL commands is that they let you call a program with one brief command (by typing PIP, for example, rather than RUN $PIP). CCL commands also allow you to type one brief command, rather than initiating the dialogue normally provided by most system library programs. You can type QUE MYFILE.DAT, for example, and queue the file MYFILE.DAT to be listed at the line printer.

Chapter 4
Working with Data: Files and Devices

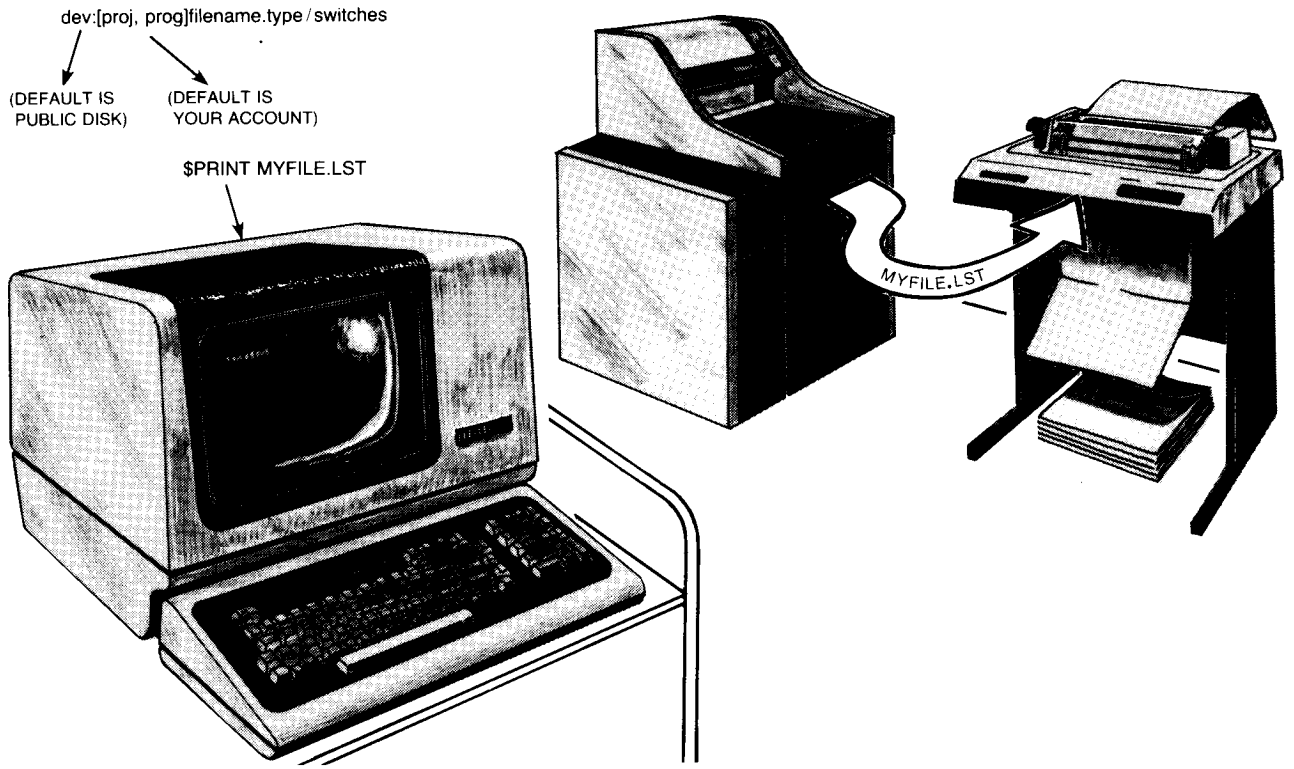
Much of the specific information you will need to use the RSTS/E system is given in other manuals. There are some features of RSTS/E, however, that are system-wide. For example, many commands in the environments discussed in Chapter 3 require that you identify a file to serve as either input or output to the command. Information that is useful to you in this system-wide context is discussed in this chapter.

4.1 The RSTS/E File Specification

As mentioned previously, much of the software you will work with produces and processes data organized as files. The RSTS/E system provides a consistent way of identifying files that takes all the variables — device, owner, name, and protection — into account. The general form for a RSTS/E file specification is:

```
device:[project,programmer]filename.type/switches
```

Figure 4–1: The File Specification



If you leave out parts of a file specification, the software will assume certain "defaults." Table 4–1 shows the defaults for the various parts. The following sections discuss the parts of the file specification in detail.

Table 4–1: System Defaults for Parts of the File Specification

Part	System Default
device:	If you do not specify a device as part of the file specification, the system assumes the public disk structure.
[proj,prog]	The project-programmer number identifies a file on tape or disk (not for other devices). If you do not specify a particular account for a file on tape or disk, the system assumes your account.
filename.type	The system does not assume a default for filename. The default for type varies, depending on the program you are running.
/switch	The defaults for switches depend upon the way your system has been defined by the system manager.

4.1.1 device: The Device Designator

If you do not specify a device designator, the system supplies the public structure by default. For non-file-structured devices (such as line printer), only the device designator need be specified; the system ignores any file name, type, or account number that you specify.

A device designator, or device specification, consists of 2 alphabetic characters optionally followed by a decimal unit number, and always terminated by a colon (:). The alphabetic characters are generally an abbreviation of the device's generic name, and the unit number uniquely identifies the device itself or the drive on which it is currently mounted.

Table 4–2 lists and explains the RSTS/E device designators.

Table 4–2: RSTS/E Device Designators

Designator	Device
DF:,DS:,DK:,DL:,DM:, DP:,DR:,DB:, or SY:	RSTS/E public disk structure.
SY0:	System disk (the unit that was bootstrapped).
DF0:	RF11 disk (all platters).
DS0: to DS7:	RS03 /RS04 fixed-head disk units 0 through 7.
DK0: to DK7:	RK05 disk cartridge units 0 through 7.
DL0: to DL3:	RL01 /RL02 disk cartridge units 0 through 3.
DM0: to DM7:	RK06 /RK07 disk cartridge units 0 through 7.
DP0: to DP7:	RP02 /RP03 disk pack units 0 through 7.
DR0: to DR7:	RM02 /RM03/RM05 disk units 0 through 7.
DB0: to DB7:	RP04 /RP05 /RP06 disk pack units 0 through 7.

(continued on next page)

Table 4–2: RSTS/E Device Designators (Cont.)

Designator	Device
PR:	High-speed paper tape reader.
PP:	High-speed paper tape punch.
CR:	CR11 punched or CM11 mark sense card reader.
CD:	CD11 punched card reader.
MT0: to MT7:	TE10 /TU10 /TS03 magnetic tape units 0 through 7.
MM0: to MM7:	TE16 /TU16 /TU45 /TU77 magnetic tape units 0 through 7.
MS0: to MS3:	TS11 magnetic tape units 0 through 3.
LP0: to LP7:	Line printer units 0 through 7.
DT0: to DT7:	TU56 DECtape units 0 through 7.
DD0: to DD7:	TU58 DECtape II unit 0 through 7.
KB:	Current user terminal.
KBn:	Terminal n in the system.
TTn:	Terminal n in the system (synonym for KBn:).
TI:	Current terminal (synonym for KB:, the terminal that initiated the job).
DX0: to DX7:	RX01 /RX02 flexible diskette units 0 to 7.

NOTE

You can reference LPn:, DTn:, DXn:, KBn:, MMn:, MTn:, MSn:, and DDn: where n is between 0 and the maximum number of such units on the system. LP:, DT:, DX:, MM:, MT:, MS:, and DD: are each the same as specifying unit 0 of the related device.

If your system has only TS11 tape units, the designators MS: and MT: are synonymous. If your system has only TE16, TU16, TU45, or TU77 tape units, the designators MM: and MT: are synonymous. On systems which have the CD11 card reader, the designator CR: is synonymous with CD:.

If you attempt to specify a device or type of device not available on the system, the error message ?NOT A VALID DEVICE is printed.

Logical Device Names

A device may be identified either by a physical device name (i.e., a device designator) or by a logical device name; logical device names are associated with devices by the ASSIGN command (see Section 4.3.6) or by the system manager. A logical device name consists of 1 to 6 characters terminated by a colon. Examples of logical device names are DTA:, DTFACT:, MTMINE:, MYPACK:, DK5:. A project-programmer (account) number may be associated with a logical name.

NOTE

When you specify a device name, physical or logical, the system seeks the device by the following procedure: 1) the system determines if the device name is specific to one job; 2) if it is not, the system determines if it is a system-wide logical name or disk pack identification label; 3) if the device name is neither job-local nor system-wide, the system determines if it is a physical device designator.

An underscore before a logical device name suppresses interpretation of the name as a logical name. For example, if you had, for whatever reason, assigned the logical name MT0: to a disk, you could specify _MT0: when you wanted to refer to magnetic tape unit 0.*

System-Wide Logical Names

At the system manager's discretion, a system-wide logical name may be associated with a device or a device and an account on the device. This name, once assigned, may be used by all jobs on the system. For example, a manager associates the logical name CUP: (for Commonly Used Programs) with account [3,4] on the RK05 disk cartridge DK3:. Subsequently, you can execute a program FOO that is under account [3,4] on DK3: by typing the RUN command:

```
RUN CUP:FOO
```

As a result of this command, the system searches for an executable file named FOO under account [3,4] on RK05 disk unit 3.

You can override the default account associated with a system-wide logical name by specifying another account to the right of the logical name. For example, if your account is [200,210] and you wish to run the program OTHER from account [200,240] on disk pack SCRACH: type the following command:

```
RUN SCRACH:[200,240]OTHER
```

As a result, the system searches the disk pack SCRACH: for an executable file named OTHER under account [200,240] rather than under any account associated with the logical name or under your own account [200,210]. Thus, the account appearing to the right of the logical name overrides the default account.

---

* Utilities provided with the RT11 emulator (such as MACRO, LINK, and PIP) will not always process the underscore as described above, due to the way RT11 is emulated on RSTS/E systems.

NOTE

If the system manager has associated an account with the system-wide logical name, the placement of the device name and account is significant. If, in the preceding example, you were to type:

```
RUN [200,240]SCRACH:OTHER
```

the system would generate the ?ILLEGAL FILENAME error.

On the other hand, the manager may associate a system-wide logical name with a physical device name only, without specifying an account on that device. In this case, the default account is that of the job accessing the device. If, for example, the disk pack DP1: has been assigned the system-wide logical name SCRACH:, type the command:

```
RUN SCRACH:MYFILE
```

to cause the system to look only under your account on DP1: for the program MYFILE.

4.1.2 [proj,prog] The Project-Programmer or Account Number

If you do not specify a project-programmer code, the system assumes your number by default; that is, the owner of the file is assumed to be the current user. This code is meaningful only for disk and magnetic tape files; it has no significance for files on DECtape or on non-file-structured devices. The [proj,prog] code consists of two decimal numbers between 0 and 254, separated by a comma, and enclosed within square brackets[ ]. These numbers have the following meanings:

proj — represents a project group consisting of users with a common task or interest.

prog — represents an individual user in that project group.

RSTS/E uses project-programmer codes for two general purposes: 1) to identify files according to their owners, and 2) to apply the files' protection codes to individual users and to groups of users (see Section 4.1.4.5). Note that a project-programmer number (PPN) is identical in use and format with a User Identification Code (UIC) described in some RSTS/E utility and optional software documentation.

Special Account Characters

In most cases, you can specify one of the special nonalphanumeric characters listed below for a project-programmer number. Each of these characters designates an account. The $, for example, designates the system library account [1,2]. Thus, a file specification containing a $ denotes a file stored in that account. The usual application of $ is in calling a system library program — the command RUN $PIP, for example.

The special account characters and the accounts they designate are as follows:

Character	Account
$ (ASCII 36)	[1,2], system library account.
! (ASCII 33)	[1,3], determined by manager. (Not used in the DCL environment.)
% (ASCII 37)	[1,4], determined by manager.
& (ASCII 38)	[1,5], determined by manager.
# (ASCII 35)	[proj,0].
@ (ASCII 64)	assignable account.

The # character allows each project on the system to have its own library of files common to all members of that project.

Assignment and use of the @ character are explained in Section 4.3.6.5. If you specify the @ character without having previously assigned it to an account, the ?ILLEGAL FILE NAME error is printed. Note that the @ character can also be used in an indirect command file specification.

4.1.3 filename.type The File Name and Type

For file-structured devices, each file is assigned a file name and type.

The file name is a string of one to six alphanumeric characters, without embedded nulls, tabs, or spaces. It is the only element of a file specification without a delimiting mark of punctuation. The file name may also include or consist of the wildcard character ?. Or, it may consist of only the wildcard character * (see Section 4.1.3.1).

The file type is a string of one to three alphanumeric characters without embedded nulls, tabs, or spaces and is preceded by a dot (.). Usually, the type denotes the kind of file: .BAS, for example, indicates a BASIC–PLUS source file; .CTL, a program control file; .TMP, a temporary file (see Table 4–3 for common RSTS/E file types and their meanings). Note that a temporary file (i.e., an explicit .TMP file type) can be deleted when the system reboots, when the disk is cleaned, or a similar situation occurs. A type may include, or consist of, the wildcard character ?. Or, it may consist of only the wildcard character * (see Section 4.1.3.1).

Table 4–3: RSTS/E File Types

Type	Meaning
.B2S	BASIC–PLUS–2 source file.
.BAC	BASIC–PLUS compiled output file.
.BAS	BASIC–PLUS source file.
.BAK	BAcKup file created by various programs.
.CBL	COBOL source file.
.CMD	indirect CoMmanD file for running a system program.
.CRF	Cross ReFerence listing file.
.CTL	BATCH ConTroL file.
.DAT	DATa file.
.DIF	FILCOM output file.
.DIR	DIRectory file.
.DOC	RUNOFF output file.
.FOR	FORTRAN source file.
.HLP	system program HeLP text file.
.LIB	Resident LIBrary file.
.LOG	BATCH output LOG file.
.LST	LiSTing file created by a system program.
.MAC	MACro source file.
.MAP	MAP file produced by a linker.
.MLB	MACRO LiBrary file used by the MACRO Assembler.
.OBJ	compiled MACRO, COBOL, FORTRAN, or BASIC–PLUS–2 program.
.ODL	Overlay Description Language input file.
.OLB	Object module LiBrary file used by the Task Builder.
.PMD	Post-Mortem Dump.
.RNO	RUNOFF source file.
.RTS	Run-time system.
.SAV	Executable program under the RT11 run-time system.
.SIL	Save Image Library.
.SRT	SORT–11 file.
.STB	Symbol TaBle file produced by the Task Builder.
.SYS	SYStem file, usually for internal use.
.TMP	TeMPorary file created by a system program (deleted when you log off or if the disk is CLEANed).
.TSK	Executable program under the RSX run-time system.
.TXT	ASCII TeXT file.

A null file type, in which only the dot is specified, or a blank file type, in which case the dot and file type field are both omitted from the file designation, are permitted.

4.1.3.1 Wildcard Specifications — Many of the RSTS/E system library programs, when requiring a file specification, allow you to specify the wildcard characters * (asterisk) and ? (question mark). The * character replaces an entire field, while the ? character replaces a character within a field.

4.1.3.2 The * Wildcard — The * (asterisk), replacing a file name, type, or both (one * for each), denotes all file names, all types, or all file names with any types. For example:

FILE.* denotes all files with file name FILE and any type; for example, FILE.DAT, FILE.TXT, FILE.8, FILE.9B.

*.TYP denotes all files with TYP types; for example, DATA.TYP, MYFILE.TYP, FB10.TYP, DD.TYP.

. denotes all files; for example, all the files listed as examples above.

4.1.3.3 The ? Wildcard — The ? (question mark), in any position of either the file name or type, denotes any alphanumeric character appearing in that position. The following examples illustrate:

FILE.TY? denotes all files with file name FILE and a type consisting of TY, or of TY and any other alphanumeric character; for example, FILE.TY1, FILE.TY2, FILE.TYF, FILE.TYE, FILE.TY.

FILE??.TYP denotes all files with type .TYP and a file name consisting of FILE and any other two alphanumerics, including trailing blanks; for example, FILE01.TYP, FILE54.TYP, FILE3B.TYP, FILE3.TYP.

FILE??.E?? denotes all files with a file name consisting of FILE and any other two alphanumerics and a type consisting of E and any other two alphanumerics (trailing blanks are included); for example, FILE54.TYP, FILE01.ERA, FILEJQ.E91, FILE91.EJQ.

FI?.TYP denotes all files with type .TYP and a file name consisting of FI and any other alphanumeric (trailing blanks are included); for example, FI1.TYP, FIL.TYP, FI7.TYP, FIG.TYP.

4.1.3.4 The * and ? Wildcards Combined — The * and ? wildcards may be intermixed in a file specification. The following examples show such mixtures and their meanings:

FILE??.* denotes all files with any type and a file name consisting of FILE and any two alphanumerics; for example, FILE60.DAT, FILE75.DAT, FILEZX.TXT, FILES5.B, FILEB6.AM.

*.TY? denotes all files with a type consisting of TY and any other alphanumeric; for example, MYFILE.TYP, YRFILE.TYP, MCR.TYE.

4.1.4 /switch File Specification Switch

A file specification switch may be included as the final element of the specification string. Six options are possible: /PROTECT, /FILESIZE, /CLUSTERSIZE, /POSITION, /MODE, and /RONLY. These switches specify, respectively, the disk size — in blocks — to which the file is pre-extended, the minimum number of contiguous disk blocks forming a cluster, the specific block number on disk at which the file is to be placed, the read/write mode in which the file's data is passed to the device driver and the protection to be given to the file. Note that the switches /FILESIZE, /CLUSTERSIZE, and /POSITION are used for disk files.

If a file specification switch is not at the end of the file specification, is missing a colon, or is not a valid form, the system prints the error message ?ILLEGAL SWITCH USAGE. For example, either of the following specifications will produce the switch usage error:

```
ABC/SI:100 [1,2]
```

or

```
ABC/SIQ
```

If an argument is missing, or contains an illegal character, the system generates the ?ILLEGAL NUMBER error.

4.1.4.1 /PROTECT Switch — The /PROTECT switch establishes a protection code for the file. The /PROTECT switch consists of 1) a slash (/), 2) PROTECT or a minimum abbreviation of PROT, 3) a colon (:), and 4) a number specifying the protection code. Use a pound sign (#) before octal numbers, and a decimal point (.) to terminate decimal numbers.

The protection code is a string of one to three digits*. This string determines the file's degree of protection on two levels: the actions — reading, writing, and deleting — against which it is protected, and the user or class of users against whom it is protected. There are three such user classes, which the system recognizes by project-programmer numbers as follows:

1. The individual user (owner)
   who is recognized by his or her programmer number; for example, [200,25].

2. The user's project group
   which is recognized by the user's project number; for example, [200,25], [200,57], [200,70].

---

* You can specify a protection code enclosed in angle brackets, as <nnn>, but this capability may not be in future releases of RSTS/E, so the /PROTECT switch is recommended.

3. All other users on the system
   who are recognized by the existence of valid project-programmer numbers; for example, [225,60], [250,35], [254,10].

Thus, two variables — read /write privileges and class of user — determine protection. Degrees of protection for data files are enforced by the following individual codes; typically, a file's total protection code is the sum of the desired combination of individual codes. These individual codes and their meanings are listed in Table 4–4.

Table 4–4: File Protection Codes

Code	Meaning
1	read protection against owner.
2	write protection against owner.
4	read protection against owner's project group.
8	write protection against owner's project group.
16	read protection against all others who do not have owner's project number.
32	write protection against all others who do not have owner's project number.
64	executable program.
	Individual codes added to the compiled protection <64> have meanings different from those of the data file protection codes above. These compiled codes follow:
1	execute protection against owner.
2	read and write protection against owner.
4	execute protection against owner's project group.
8	read and write protection against owner's project group.
16	execute protection against all others who do not have owner's project number.
32	read and write protection against all others who do not have owner's project number.
128	program with temporary privileges or file with protected data.

In accordance with the codes in Table 4–4, therefore, a data file with protection 60 — the usual system default — is protected against reading, writing, and deleting by all users except its owner: $60 = 4 + 8 + 16 + 32$.

Note that a file that has a protection code of 128 is overwritten with zeros when it is deleted.

NOTE

When the system overwrites a protected file with zeros, all other file processing requests are deferred until the overwrite operation is completed. It is recommended that PIP be used to delete large protected files (see Section 6.11.6.18).

4.1.4.2 /FILESIZE Switch — The /FILESIZE switch allows the creation of a disk file of the specified size, in blocks, before any read /write operations are performed. Thus, /FILESIZE reserves space on the disk for data to be placed in the file. This switch consists of 1) a slash (/), 2) any one of the unique abbreviations shown in the list that follows, 3) a colon (:), 4) an optional pound sign (#) for octal conversion of the argument n, 5) the argument n, a decimal number which indicates the number of blocks in the pre-extended file, and 6) an optional trailing decimal point (.) to ensure that n is interpreted as a decimal number.

The argument n specifies the length in disk blocks to which the file is pre-extended. That is, it reserves a specified portion of disk space for the file. The value of the /FILESIZE argument is dependent on the type of file. If the file's protection marks it as executable, the argument n must be in the range of 0 to 65535. If the capability to create large files is present on your system and the file is not marked as executable, the argument n can be in the range of 0 to 2**23–1 (assuming that the disk is large enough). Note that you cannot run an executable file that is larger than 65535 blocks. An attempt to pre-extend such a file beyond block 65535 returns a ?PROTECTION VIOLATION error.

The /FILESIZE switch may be minimally abbreviated to /FI or to /SI. The following list shows its valid forms:

```
/FI:[#]n[.]
/FIL:[#]n[.]
/FILE:[#]n[.]
     .
     .
     .
/FILESIZE:[#]n[.]
```

or

```
/SI:[#]n[.]
     .
     .
     .
/SIZE:[#]n[.]
```

4.1.4.3 /CLUSTERSIZE Switch — The /CLUSTERSIZE switch establishes the minimum cluster size for a disk file; a cluster is a number of contiguous blocks taken together as a unit. The /CLUSTERSIZE switch is especially useful for large files; specifying a large cluster size speeds up random access to the data and prevents such files from crowding or filling your directory, whose size is limited. RSTS/E permits cluster sizes of 1, 2, 4, 8, 16, 32, 64, 128, or 256 blocks.

The /CLUSTERSIZE switch consists of 1) a slash (/), 2) CLUSTERSIZE or a minimum abbreviation of CL, 3) a colon (:), 4) an optional minus sign (–) to specify a negative cluster size (explained in the next paragraph), 5) an optional pound sign (#) for octal interpretation of the argument n, 6) the

argument n specifying the cluster size in blocks, and 7) a decimal point (.) to ensure decimal interpretation of n. The following list shows the valid forms of the option:

```
/CL:[-][#]n[.]
/CLU:[-][#]n[.]
/CLUS:[-][#]n[.]
     .
     .
     .
/CLUSTERSIZE:[-][#]n[.]
```

Specifying a negative cluster size avoids certain errors associated with disk devices. A negative cluster size causes the system to use the absolute value of the cluster size, if the device on which the file is created allows that value. If the absolute value is less than the device cluster size at which the file is to be created, the system uses the device cluster size instead of returning an error. For example, assume you are accustomed to creating a file with a cluster size of 2 on an RK05 disk cartridge, which is a system scratch disk. The scratch disk, however, temporarily changes to an RP06 disk pack, which has a device cluster size of 8. Your customary cluster size of 2, therefore, would be illegal on the RP06. But by specifying the negative cluster size of –2, you guarantee that the file creation will not fail because of the RP06 disk's cluster size restriction.

4.1.4.4 /POSITION Switch — The /POSITION switch allows you to specify the location of a file on disk. The option consists of 1) a slash (/), 2) POSITION or a minimum abbreviation of PO, 3) a colon (:), and 4) the argument n, a decimal number that specifies the desired placement of the file.

The argument n indicates a device cluster number on the disk. The device cluster numbers vary from disk to disk; see the *RSTS/E System Generation Manual* for the table of device sizes. If the value of n in /POSITION:n is 0, no placement is performed and the system determines the location of the file on the disk. If the value of n is a nonzero number, the system attempts to place the first block of the file at the specified device cluster number on the disk. If the file can not be placed at that location, the system places the first block of the file at the first free cluster number that is greater than the specification. When a file is placed, it is marked as such by the monitor. Note that no error is returned if the system is unable to place the file at the specified location. To determine the actual location of the file on disk, use the /PO,/FU, or /S options of the DIRECT program (see Section 6.2).

For example:

```
DIR /PO
SY:[2,211]
RMSTST.B2S 18698
RMSTST.OBJ 18720
RMSTST.CMD 18721
RMSTST.ODL 18722
```

```
RMSTST.TSK 18701
RMSTST.MAP 18723
PROOF .BAS 18741
PFILE .DAT 18742
RDPLN1.RNO 18743
RDPLN1.DOC 18747
UBAK17.RNO 18751
UBAK17.DOC 18767
CHTST .BAS 6020
CHTST1.BAS 9255
CHTST1.B2S 16257
CHTST1.OBJ 16313
CHTST1.CMD 16433
CHTST1.ODL 16522
CHTST1.TSK 16562
```

The following list shows the valid forms of the /POSITION option:

```
/PO:n
/POS:n
/POSI:n
  .
  .
  .
/POSITION:n
```

Note that the position of a file on disk may change if you backup and restore the file with the BACKUP program.

4.1.4.5 /MODE and /RONLY Switches — The /MODE switch enables the passing of up to 16 (decimal) bits of information to the device driver at file open time. The meaning of these bits (if any) is device dependent, and determines the read /write mode for data transfer. For explanations of the bits, see the *RSTS/E Programming Manual.*

The /MODE switch consists of 1) a slash (/), 2) MODE or a minimum abbreviation of MO, 3) a colon (:), 4) an optional pound sign (#) for octal interpretation of the argument n, 5) the argument n specifying a mode setting between 0 and 32767 (decimal) inclusive, and 6) an optional decimal point (.) to ensure decimal interpretation of n. The following list shows the valid forms of the option:

```
/MO:[#]n[.]
/MOD:[#]n[.]
/MODE:[#]n[.]
```

The /RONLY switch enables setting of the read only MODE value for a disk file. The /RONLY option consists of 1) a slash (/), and 2) RONLY or a minimum abbreviation of RO. The following list shows the valid forms of the option:

```
/RO
/RON
/RONL
/RONLY
```

4.2 System-Wide Input and Output Control Characters for Terminals

The control characters described in this section are useful in all command environments. A character preceded by the word "control" (abbreviated as CTRL) is typed by holding down the CTRL key, typing the character (C, for example), and releasing both keys.

4.2.1 CTRL/C

Typing a CTRL/C causes RSTS/E to stop a program and return to the command environment.

Note that CTRL/C interrupts processing. For example, if you use CTRL/C to stop EDT when it is writing your file to disk, EDT will close the file and exit. All of the file that was not yet written is lost.

If your keyboard monitor is BASIC–PLUS or RT11, you can type CONT to cause some programs to continue execution. In general, though, you should not use CTRL/C unless you don't care that work may be lost.

Note that, unless disabled by the system manager, the terminal BREAK key operates as a CTRL/C combination.

4.2.2 CTRL/O

The CTRL/O combination suppresses output to the terminal until the next time CTRL/O is typed. When a program produces a large amount of output, you may not wish to wait for the printing of the complete information. CTRL/O enables you to monitor the output while not stopping it completely. Typing CTRL/O while output is occurring does not stop the computer's output; the terminal, however, does not display it. The second time CTRL/O is typed, the output is again displayed at the terminal. Output, however, does not resume at the point of the first CTRL/O, but at the point of the second CTRL/O; thus, some output may be skipped in the printing.

Unlike CTRL/C, CTRL/O does not terminate program output. It is useful to think of CTRL/O as a switch, whose first setting creates a condition and whose second setting releases the condition.

4.2.3 CTRL/R

The CTRL/R combination displays all buffered terminal input. For example, if you type a line that contains characters deleted by RUBOUT and you type CTRL/R prior to typing a line terminator, the system retypes the line with deleted characters removed. Thus on hard copy terminals, CTRL/R allows you to examine a corrected line before its transmission by a line terminator.

Note that the use of CTRL/R can be enabled or disabled with the TTYSET program.

4.2.4 CTRL/S and CTRL/Q

These two control characters work together on screen terminals. CTRL/S temporarily suspends output to the display terminal. It is used to examine the lines currently displayed before they are replaced on the screen by additional lines. Output can be resumed at the next character by typing the CTRL/Q combination. Note that the CTRL/S and /Q feature can be enabled or disabled with the TTYSET program (Section 6.15) or the DCL SET TERMINAL command.

4.2.5 CTRL/Z

Many programs on RSTS/E systems are designed to recognize CTRL/Z as stopping the program "gracefully." Check the descriptions for the various programs to see what CTRL/Z does. In the BASIC–PLUS environment, the CTRL/Z combination is used to mark the end of a data file. When data is input from the terminal, the CTRL/Z character marks the end of recorded data. The message ?END OF FILE ON DEVICE is printed by the system when a CTRL/Z is detected, unless an ON ERROR GOTO statement is used to enable a BASIC–PLUS routine to handle the error.

4.2.6 RETURN Key

The RETURN key is normally used to terminate a line and enter that line to the system. The RETURN key, when typed, usually echoes as a carriage return/line feed operation on the terminal.*

4.2.7 ESCAPE or ALTMODE Key

The ESCAPE key, like the RETURN key, is used to terminate the current line and causes the line to be entered to the system. The ESCAPE key, however, echoes on the terminal as a $ character and does not perform a carriage return/line feed.

On some terminals the ESCAPE key is replaced by the ALTMODE key, which performs the same functions.

4.2.8 CTRL/T

The CTRL/T combination generates a one-line status report on the current job. When you type CTRL/T, a report in the following format is printed. This is the same format as printed for all jobs in the SYSTAT report (Section 6.14).

```
33      KB37      NONAME+BASIC      ^C(OR)  2(16)K+16K      0.5(+0.5)
```

---

* Unless the terminal is in "RSX mode," as described in the *RSTS/E System Directives Manual.*

where:

33 is the current job number.

KB37 is the keyboard number of this terminal.

NONAME is the program or operation that is being performed by the current job.

+BASIC is the current run-time system.

^C(OR) is the current state of the job; in this case, the job is in CTRL/C state (keyboard monitor wait on channel 0).

The possible job states reported by CTRL/T include the following:

ss (ccx) is returned if the job is in an I/O wait state; where ss is the type of wait, cc is the channel number of the pending operation, and x is R for a read operation and W for a write. This form of job state report is shown in the example.

FP (fff) is returned if the job is in a file processor wait; where FP represents file processor and fff is one of the following functions:

CLS	close.
OPN	open an existing file.
CRE	create a file.
DLN	delete a file.
REN	rename a file.
DIR	directory look up.
UUO	process a UUO directive.
ERR	look up error message.
RST	close (reset) channel(s).
LOK	look up a file.
ASS	assign a device.
DEA	deassign a device.
DAL	deassign all devices.
CRT	create a temporary file.
CRB	create a binary (executable) file.
RUN	open a binary (executable) file for execution.
WIN	window turn for disk files.
EXT	extend an open disk file.
BYE	logout a user.
SND	send a message.
RCV	receive a message.
NET	DECnet function.
DSP	Monitor function.
???	not decodable.

ss fnn	where ss is the job state and fnn is a swap slot number (if the job is swapped out); this information is returned in SYSTAT format as described in Section 6.14.
2	the current program's size (in words) is 2K.
(16)K	the maximum program size (in words) is 16K.
+16K	the size (in words) of the run-time system is 16K.
0.5	the job has used 0.5 seconds of CPU time.
(+0.5)	the amount of elapsed time since the last CTRL/T.
-8	the current job's priority.

Note that if your system is part of a DECnet network, your system's local node name will appear between the job number and keyboard number.

To have effect, CTRL/T capability must be installed on the system during system generation and the TTYSET command SET CTRL/R (see Section 6.15) must be executed. (You can also use the DCL SET TERMINAL command.)

CTRL/T does not interrupt the job; it is a completely safe way to find out what is happening at your terminal.

4.3 ASSIGN, DEASSIGN, and REASSIGN Commands—For All Environments Except DCL

This section describes three commands that apply and adapt system resources such as devices and accounts to your individual needs. For example, you can reserve a device for you use alone, and release the reservation so that the device can be used by others.

Unlike the rest of this chapter, the functions described here are handled by different commands in the DCL environment. If you work primarily in the DCL environment, you will probably wish to ignore this section and refer to the *RSTS/E DCL User's Guide.*

4.3.1 Reserving a Device: The ASSIGN Command

The ASSIGN command reserves an I/O device for the use of one programmer (i.e., one job number).

To reserve a device for your exclusive use, type the command ASSIGN and an object, in this form:

```
ASSIGN dev:
```

where dev: is a device specification. (For a list of possible specifications, see Table 4–5.) If the device is available for use, the system prints the command environment prompt Ready, ., or >.

Following the prompt, you can perform I/O with the assigned device. (See the discussion of the PIP utility in Chapter 6.)

If the device is not available for use, the system prints the message:

```
?DEVICE NOT AVAILABLE
```

There are several reasons for a device's unavailability; it may be 1) opened or assigned by another user, 2) reserved for a specific operation (a terminal reserved for network communications), 3) restricted by the system manager for privileged use or hardware maintenance.

If the device is not configured on the system, the system prints the error message:

```
?NOT A VALID DEVICE
```

The following example illustrates successful assignment of line printer 0, but the assignment of the high-speed paper tape reader is unsuccessful because that device is unavailable.

```
ASSIGN LP:
READY

ASSIGN PR:
?DEVICE NOT AVAILABLE
```

Even if more than one job is logged into the system under a single account number, only the job (i.e., user) performing an ASSIGN (or DEASSIGN) is affected by that command. Devices reserved by a job remain under that job's control until the job releases the devices, changes accounts, or logs off the system.

Table 4–5: Assignable Device Specifications

Specification	Device
PR:	High-speed paper tape reader.
PP:	High-speed paper tape punch.
CR:	CR11 punched or CM11 mark sense card reader.
CD:	CD11 punched card reader.
MT0: to MT7:	TE10/TU10/TS03 magnetic tape units 0 through 7.
MM0: to MM7:	TE16/TU16/TU45/TU77 magnetic tape units 0 through 7.
MS0: to MS3:	TS11 magnetic tape units 0 through 3.
LP0: to LP7:	Line printer units 0 through 7.
DT0: to DT7:	TU56 DECtape units 0 through 7.

(continued on next page)

Table 4–5: Assignable Device Specifications (Cont.)

Specification	Device
DD0: to DD7:	TU58 DECtape II units 0 through 7.
KB:	Current user terminal.
KBn:	Terminal n in the system.
TTn:	Terminal n in the system (synonym for KBn:).
TI:	Current terminal (synonym for KB:, the terminal that initiated the job).
DX0: to DX7:	RX01/RX02 flexible diskette units 0 to 7.

NOTE

You can reference LPn:, DTn:, DXn:, KBn:, MMn:, MTn:, MSn:, and DDn: where n is between 0 and the maximum number of such units on the system. LP:, DT:, DX:, MM:, MT:, MS:, and DD: are each the same as specifying unit 0 of the related device.

If your system has only TS11 tape units, the designators MS: and MT: are synonymous. If your system has only TE16, TU16, TU45, or TU77 tape units, the designators MM: and MT: are synonymous. On systems which have the CD11 card reader, the designator CR: is synonymous with CD:.

4.3.2 Releasing a Device: The DEASSIGN Command

The DEASSIGN command releases an assigned device from your control back to the system's supply of available devices. Thus, DEASSIGN makes the device available for use by other jobs.

Issued with no device specification, DEASSIGN releases all of the job's assigned devices. For example:

```
DEASSIGN
```

releases all devices that you have assigned under the current job number. If you do not issue this command before logging out, the system itself performs a DEASSIGN during log out.

To release a specific device, type the DEASSIGN command followed by a device specification. For example:

```
DEASSIGN LP:
```

releases line printer unit 0.

Note that if your job has the device open on a channel, DEASSIGN won't suffice for making the device available to other jobs. For example, if you interrupt a program (written in BASIC–PLUS, say) which is writing output to the line printer by typing CTRL/C, and then type DEASSIGN LP:, your job still controls the line printer. (You have released any ASSIGNment of LP:, but its being left OPEN on some channel in your interrupted program still keeps LP: reserved to your job.) You can complete the release in such a case (still assuming BASIC–PLUS) by typing CLOSE #n% (where n% is the channel number on which LP: is OPEN), or by typing END (which closes all channels).

4.3.3 Transferring Control of a Device: The REASSIGN Command

The REASSIGN command transfers control of a device to another job. For example, if DECtape unit 1 is under control of the current job, the command:

```
REASSIGN DT1:8
```

transfers control of the DECtape to job number 8.

In performing this transfer between two jobs, the REASSIGN command also prevents a third job from gaining control of the device. In the foregoing example, job number 8 might be busy with an operation that prevents it from using DT1: immediately after reassignment. If job number 20, for instance, attempts to assign this DECtape before job number 8 is ready to use it, the attempt will be unsuccessful.

An attempt to reassign control of a device to a nonexistent job causes the system to print the %ILLEGAL NUMBER error. If the device is open or has an open file, the system generates the error ?ACCOUNT OR DEVICE IN USE. Before transferring control, you must close the device or close any files open on the device.

Magnetic tape users should note that the labeling format, density, and parity characteristics assigned to a tape unit are preserved in the REASSIGN command's transfer.

Magnetic tape users should note that the labeling format, density, and parity characteristics assigned to a tape unit are preserved in the REASSIGN command's transfer.

4.3.4 Changing the Default Protection Code: The ASSIGN Command

The ASSIGN command, followed by a /PROTECT:n switch, where n is a protection code, changes the default protection code which the system assigns to files created by the current job during time sharing. Usually, this default is 60. when you log into the system. To change it, type ASSIGN followed by a /PROTECT switch and the new code.

The following command, for example, changes the default protection to 40. and assigns that value to all files subsequently created under the job's control.

```
ASSIGN /PROTECT:40.
```

The default protection remains in effect until you assign another default protection or until you log off the system. Because the default protection is job-related, its assignment remains in effect when you change accounts. When you type a DEASSIGN command with no arguments, the assigned default protection reverts to the system's default. Note that the system always assigns a minimum protection code of /PROTECT:64. when a program is compiled.

4.3.5 Changing the Magnetic Tape Labeling Default: The ASSIGN Command

The magnetic tape labeling default is set by the system manager, and is system-wide. Though it normally remains in effect during the time-sharing session, it can be changed for an individual job.

RSTS/E supports two types of magnetic tape file labels; DOS and ANSI. DOS labels were first used by the DOS/BATCH–11 operating system. ANSI labels, where used in RSTS/E documentation, refer to the RSTS/E implementation of American National Standard X3.27–1978 – magnetic tape labels and file structure for information interchange.

To change the default, type the ASSIGN command followed by 1) the tape's physical name, and 2) either .DOS or .ANSI. For example:

```
ASSIGN MT0:.DOS
```

As a result of this command, the system reserves unit 0 for the current job and treats files on unit 0 as having DOS labels. Similarly, to change the default to ANSI labeling, type ASSIGN, the physical device name, and the new default:

```
ASSIGN MT0:.ANSI
```

Note the importance of the dot (.) character in each example. If it is omitted, the system assigns the logical name DOS or ANSI to the magnetic tape unit. The labeling default remains in effect when the device is reassigned to another job and when you change accounts.

The system program PIP can initialize (i.e., zero) either ANSI or DOS labeled magnetic tape (see Section 6.11.6.21). A magnetic tape, after being initialized, conforms to the system defaults in format and labeling unless the system default is overridden by a private user default.

NOTE

If the magnetic tape is labeled in other than the system default and the appropriate .ANSI or the .DOS option is not used, an error message is returned. The error message:

```
?BAD DIRECTORY ON DEVICE
```

appears when a directory listing is attempted. Note that you can use the DIRECT system utility program (see Section 6.2) to obtain a directory listing of the tape provided that it is written with ANSI or DOS labels under RSTS/E.

The DEASSIGN command automatically makes the magnetic tape unit return to the system's labeling default.

4.3.6 Assigning and Using Logical Names: The ASSIGN and DEASSIGN Commands

The logical names that you assign for devices, discussed in Section 4.1.1, do not depend on the physical device specifications. Thus, if you write a program referencing physical devices, you can give these devices logical names of your own choosing in the program. Before running the program, issue the ASSIGN command to associate your chosen names with the devices. This action makes the program independent of the devices' physical locations on the system.

To associate a logical name with a physical device, type the following form of the ASSIGN command:

```
ASSIGN dev:[(proj,prog)] logical name
```

where dev: is the specification of the physical device and the optional (proj,prog) is a user account (see Section 4.3.6.4). The logical name can be from one to six alphanumeric characters. A job can have a maximum of four logical name assignments at a time if only device names are specified. However, if one or more of the logical names are associated with an account, the job is limited to three logical name assignments. If you attempt to make more than the legal number of logical assignments, the system prints the error message ?ACCOUNT OR DEVICE IN USE. A logical association is unique to the job and is preserved during CHAIN operations. And because the logical assignment is job-related, it is also preserved when you change accounts. The command does not reserve the device but merely associates the logical and physical names.

4.3.6.1 Associating Multiple Logical Names with One Device — If you make two logical name assignments for the same device, the system recognizes both logical names as belonging to that device. For example:

```
ASSIGN MT1:A

ASSIGN MT1:B
```

As a result of these commands, the system associates both logical names A: and B: with magnetic tape unit 1.

If you associate two different devices with the same logical name, the system replaces the former logical assignment with the latter assignment. For example:

```
ASSIGN MT1:A

ASSIGN MT2:A
```

After execution of these commands, the system associates logical name A: with magnetic tape unit 2.

4.3.6.2 Associating a Valid Physical Name with a Device — If you associate a device with a logical name that is also a valid physical device name, the system recognizes the logical, and not the physical, assignment. For example:

```
ASSIGN MT0:MT4
```

The system subsequently associates the logical name MT4: with magnetic tape unit 0. The system makes this association only for the job that has assigned this logical name. When another job requests MT4:, the system attempts to access the physical device MT4:. Note that if you precede a device designator with an underscore, the system will access that device regardless of any logical name assignment (see Section 4.1.1).

4.3.6.3 Reserving and Releasing a Logically Named Device — To reserve a device for which a logical name exists, type the ASSIGN command, followed by the logical name and a colon. The command takes the following form:

```
ASSIGN logical name:
```

The system reserves the associated physical device if it is available. Note that the colon is required.

The following commands reserve magnetic tape unit 1 and associate a logical name with that device:

```
ASSIGN MT1:

ASSIGN MT1:ABC
```

As a result of these commands, a BASIC–PLUS statement of the form OPEN "ABC:FILE.TYP" AS FILE 1 in a subsequently executed BASIC–PLUS program attempts to open FILE.TYP on magnetic tape unit 1. Also, the subsequent use of ABC: in any system command refers to magnetic tape unit 1. An attempt to refer to a device by an unassigned logical name generates the error ?NOT A VALID DEVICE.

To release control of the physical device if a logical name is still in effect, type the following form of the DEASSIGN command:

```
DEASSIGN logical name:
```

Note that the colon is required. This command releases control of the physical device associated with the logical name. And any device, of course, can be released by issuing DEASSIGN with a physical name. Neither of these forms of the DEASSIGN command, however, cancels the association between physical device and logical name. To cancel the association between the physical device name and the logical device name, use the following form of the DEASSIGN command:

```
DEASSIGN dev:[(proj,prog)] logical name
```

4.3.6.4 Associating a Logical Name with a Device and Account — You can use the ASSIGN command to associate a device and a project-programmer number with a logical name. For example, you can associate the logical name LIB with account [2,10] on the RL01 disk cartridge DL2: with the following command:

```
ASSIGN DL2:[2,10]LIB
```

Following this command, you can run a program STAT that is under account [2,10] on DL2: with the command:

```
RUN LIB:STAT
```

This command causes the system to search for the compiled file STAT under account [2,10] on RL01 disk unit 2.

This capability is similar to the system manager (privileged) function described in Section 4.5; except that while the system manager's logical assignment is system-wide, your assignments are local and apply only to your job. That is, you can associate a logical name with a device and account, but other jobs cannot use the logical name to refer to the associated device and account. Other users can, however, make their own logical assignments and may use the same logical name. These local logical name assignments override system-wide logical names, but only for the job which executed the ASSIGN command.

NOTE

As described in Section 4.3.6, you are normally allowed to make a maximum of four logical assignments. However, if you include a project-programmer number in one of those assignments, you are allowed a maximum of three. Also, because of the way that the monitor stores logical assignments, you cannot make an account assignment if four previous logical assignments were made. Your job would have to deassign two of the previous assignments before an account assignment could be made.

4.3.6.5 Associating the Logical @ with a User Account — The ASSIGN command, followed by an account number, establishes a logical association between that account and the commercial at sign (@) character. For example, the command:

```
ASSIGN [100,101]
```

associates the @ character with the account [100,101]. The @ character, therefore, when used in subsequent commands and program statements, refers to account [100,101]. For example, the command DIR @ prints a directory of account [100,101]. Moreover, BASIC–PLUS statements such as:

```
OPEN "[100,101]FILE.TYP" AS FILE 1
```

can be shortened in the following manner:

```
OPEN "@FILE.TYP" AS FILE 1
```

If an account has not been logically assigned, an attempt to refer to it by the @ character generates the error ?ILLEGAL FILE NAME.

To cancel the association between the @ character and the account, type the DEASSIGN command in one of the following two forms:

```
DEASSIGN [100,101]
```

or

```
DEASSIGN @
```

The logical assignment remains in effect until you issue the DEASSIGN @ command, or until you log off the system, or until you make another assignment. Because the logical assignment of an account is job-related, it remains in effect even when you change accounts.

Chapter 5
A Cookbook for Working with Disks and Tapes

You may never have to work directly with magnetic tape and disk devices on your system. If you do, however, and you are not in the DCL command environment, you may find this chapter useful in "tying together" the multitude of commands and utility programs available on RSTS/E into one systematic set of operations for working with disks and tapes. The DCL command environment, while not as rich as using the procedures described here, is much simpler. See the *RSTS/E DCL User's Guide* for more information.

5.1 Disks

Each disk or DECpack cartridge on the system will have written on it a *pack identification label.* This label is not a physical name, because it is independent of the drive unit currently holding the pack.

First, you physically put the disk pack in the disk unit. To access files on the disk, you must then logically mount the disk pack. This establishes the association between the pack and its identification label. You do a logical mount with the MOUNT command.* The general form of the MOUNT command is:

MOUNT dev:packid

---

* The MOUNT command may be processed by the monitor or by the UMOUNT utility when MOUNT has been defined by the system manager as a Concise Command Language (CCL) command for the UMOUNT program. If MOUNT has been defined as a CCL, it will take precedence over the monitor's processing. The format described here is the same in either case, however.

Once this is done, you may want to ASSIGN the logical name you use in your own commands or program to the disk.

You can now perform whatever processing you wish. For example, you can transfer files to and from the disk pack using the PIP utility (Section 6.11), copy the entire disk pack using the COPY utility (Section 6.1), or display the contents of the disk pack using the DIRECT utility (Section 6.2).

Once you are finished with processing, you DISMOUNT the disk and remove the pack for storage. The example below shows mounting a disk pack labelled "UTFILS", assigning the logical name "MYPACK" to the disk, listing the files on the disk, copying all files in the user's account from the disk to the user's account on the public disk structure, and dismounting the disk.

```
MOUNT DP1:UTFILS

Ready

ASSIGN UTFILS:MYPACK

Ready

RUN $DIRECT
DIRECT V7 RSTS V7 SYSTEM
#MYPACK:
 Name .Typ    Size    Prot    Date    DP1:[2,196]
TRXWTL.TSK    332     < 60>   22-Oct-81
TRXDOC.TSK    524     < 60>   11-Dec-81
        .
        .
        .

Total of 4603 blocks in 70 files in DP1:[2,196]
#^Z

Ready

RUN $PIP
**.*=MYPACK:*.*
*^Z

Ready

DISMOUNT MYPACK

Ready
```

5.2 Tapes

As with disks, there are a variety of commands and routines for working with magnetic tapes. (Again, you may find DCL easier than the procedures described here.) The first concern is whether you want to use (1) a brand new tape or a "scratch" tape (one that has files written on it that are no longer needed), or (2) a tape that has files written on it that you want to access.

5.2.1 Brand New Tape or Scratch Tape

If you are using a brand new or a scratch tape, you need to initialize it to prepare it for processing. First, assign the device, naming the format (.DOS for Disk Operating System format or .ANSI for the American National Standards Institute format).* You then run PIP, giving a volume identification to the tape reel (if you selected ANSI format), zeroing the tape, and setting a density and parity, if desired. Density can be 800 or 1600 (bits per inch) and parity can be ODD or EVEN.* For example:

```
ASSIGN MTO:.ANSI

Ready

RUN $PIP
*MTO:MYTAPE/ZE/DENSITY:800/PARITY:ODD
Really zero MTO:MYTAPE.    /PARITY:ODD/DENSITY:800? YES
*(other processing, if desired)
*^Z

Ready

DEASSIGN MTO:
```

5.2.2 Tape Already Containing Files

To work with tapes that have already been initialized, it is simplest to use the MOUNT command. MOUNT is a CCL command that runs the system utility UMOUNT (Section 6.18).

MOUNT allows you to specify the following switches: /DOS or /ANSI, to specify the format of the tape, /DENSITY:n, /PARITY:par.* If you select /ANSI format, you also supply a volume identification immediately after the device designation. The MOUNT command also logically assigns the tape drive to your job.

Once the tape is logically mounted, you can run PIP, for example, to do the processing you want, and then use the DISMOUNT command with the /UNLOAD switch to rewind the tape and take it offline. The DISMOUNT also releases the assignment done by the MOUNT command. For example:

```
MOUNT MTO:MYTAPE/ANSI/DENSITY:800/PARITY:ODD

Ready

RUN $PIP
*MYFILE.TXT=MTO:MYFILE.TXT
*^Z

Ready

DISMOUNT MTO:/UNLOAD

Ready
```

---

* For any particular tape drive, a default format, density, and parity have been specified by the system manager. These system defaults will be used unless you change them, using the switches described and shown in the examples.

This example mounts an ANSI labelled tape, with odd parity and 800 bits per inch density, labelled MYTAPE on magnetic tape drive zero. PIP is then used to transfer the file MYFILE.TXT from the tape to the user's account on the public disk structure. Finally, the tape is dismounted and "unloaded"; that is, the tape is rewound and set offline so that no other user can access it.

Chapter 6
System Utility Programs

This chapter describes utility programs available on RSTS/E systems. Table 6–1 gives an overview of the utilities and the functions they perform. Table 6–2 lists the utilities that can be run by Concise Command Language (CCL) commands. CCL commands are defined by the system manager. For a full list of CCLs available on your system, see your system manager.

The utilities are described in following subsections, arranged in alphabetical order.

Table 6–1: Overview of System Utility Programs

Function	Program	Location
Account reporting	MONEY	6.10
Comparing files	FILCOM	6.3
Device, copying	COPY	6.1
Device, requesting	QUE	6.13
Device transfers	PIP	6.11
Directory, listing	DIRECT	6.2
Diskette; IBM transfer	FLINT	6.5
Disk; RT–11, DOS, and RSTS/E transfer	FIT	6.4
Disk, mounting private	UMOUNT	6.16
Dumping user memory	PMDUMP	6.12
Flexible diskette transfer	FLINT	6.5
Information on system programs	HELP	6.7
Logging in	LOGIN	6.8
Logging out	LOGOUT	6.9

(continued on next page)

Table 6–1: Overview of System Utility Programs (Cont.)

Function	Program	Location
Magnetic tape, mounting private	UMOUNT	6.18
Message to manager	GRIPE	6.6
Quota report	QUOLST	6.14
Status report on system	SYSTAT	6.16
Switch to another keyboard monitor	SWITCH	6.15
Terminal settings	TTYSET	6.17

Table 6–2: CCL Commands that Run System Utility Programs

Program	Command	Function of Command
DIRECT	DIR[ECTORY] DIR[ECTORY] filespec DIR[ECTORY]/option(s)	Displays standard directory of your account. Displays directory of specified file. Displays directory according to option(s) specified.
FLINT	FLI[NT]	Runs the FLINT dialogue.
HELP	HELP topic	Lists information on system programs and resources (use, command syntax, etc.).
UMOUNT	MOU[NT] device:pack id label/option(s)	Logically associates the specified disk pack with the specified identification label, and applies the specified option(s).
	DIS[MOUNT] device:pack id label	Logically dismounts the disk pack on the specified drive, and removes the specified pack identification label from system's table of logical names.
PIP	PIP PIP [command line]	Runs the PIP program. Executes [command line] as a standard PIP command.
TTYSET	SET [command line]	Executes [command line] as a standard TTYSET command.
QUE	QU[EUE] [command line]	Executes [command line] as a standard QUE command.
SYSTAT	SY[STAT] SY[STAT]/option(s)	Displays standard system status report. Displays system report according to option(s) specified.
SWITCH	SW[ITCH] SW[ITCH] kbmon	Switch to another keyboard monitor.
QUE SUBMIT	SU[BMIT] file spec/option	Submits a job to the Batch processor.

6.1 Copying Between Devices: The COPY Program

All of the information on a DECtape, magnetic tape, or RK05 disk can be copied by using the COPY system program and the /FC (fast copy) switch.

To run COPY, type:

```
RUN $COPY
```

COPY prints a pound sign (#) when it is ready to receive instructions. To print a help message, type /HE. To copy a device, respond in the format shown below.

```
#new device=old device/FC
```

For example:

```
#DT1:=DT2:/FC
```

In this example, all of the data, programs and directories are copied, block by block, from the DECtape on unit 2 to the DECtape on unit 1. The original information on unit 2 is still intact.

Notice that all of the information on a device is copied; individual files cannot be specified.

Only magnetic tapes, DECtapes, and RK05 disk packs should be copied. In addition, information can be copied only to different units of the same device. That is, COPY can be used to copy a DECtape to another DECtape or a magnetic tape to another magnetic tape, but not to copy a DECtape to magnetic tape or vice versa. If an attempt is made to copy the information on one type of device onto another type of device, the error message ?MUST HAVE SAME TYPE DEVICES is given. Do not use COPY to transfer a bootable magnetic tape between tapes of different densities. The bootstrap block on the tape works in only one density.

If COPY encounters a bad block on the old device, it aborts the operation and prints the following error message:

```
?DATA ERROR ON DEVICE dev:
```

where dev: is the device containing the bad block.

Note that COPY is not equipped to detect bad blocks on the new (output) device. Hence, COPY is not recommended for backing up data on devices that allow bad blocks, such as RK06 and the RP series of disks.

Finally, if the same unit number of a device is specified twice, the error message ?MUST HAVE DIFFERENT UNIT NUMBERS is given.

When the information is successfully copied from one device to another, the program terminates, and the system prints the command environment prompt.

To verify that the information on a device unit has been copied properly, issue the /VE (verify) option to the COPY program in either of the following ways:

```
#DT1:=DT2:/FC/VE
```

or

```
#DT1:=DT2:/NC/VE
```

Since verification is performed block by block, it requires as much time as the copy operation. Therefore, it is important to understand the difference between the commands. The first command, which includes the /FC switch, copies information from DT2 to DT1 and then verifies that the information was copied correctly. The second command, which must include the /NC (no copy) switch, does not copy information; it only verifies that the information on both DECtapes is identical.

The two sample commands are the only allowable forms for verifying. If you type a command string omitting the /FC or /NC option, the error message ?/FC OR /NC MUST BE SPECIFIED is returned.

If you specify both the /FC and /NC options in the same command line, the error message ?CANNOT SPECIFY BOTH /FC AND /NC is returned.

As with the /FC option, individual files cannot be specified with the /VE option; all of the information on a device is verified.

As the verification is performed, the first message COPY prints is:

```
BEGINNING VERIFICATION PASS
```

If all of the information has been copied correctly from one device to another, the next message COPY prints is:

```
VERIFICATION COMPLETE 0 BAD BLOCKS
```

If, however, the information has not been copied correctly, COPY prints the decimal number of blocks in which inconsistencies appear. For example:

```
#DT1:=DT2:/FC/VE
BEGINNING VERIFICATION PASS
THE FOLLOWING BLOCKS ARE BAD:
17
31
89

VERIFICATION COMPLETE 3 BAD BLOCKS
```

The /BL switch, which specifies blocksize, speeds up copying or copies magnetic tapes written with nonstandard record sizes. To speed up copying, specify a larger blocksize. For example:

```
#DK0:=DK1:/BL:2048/FC
```

This command causes the disk on drive DK1: to be copied to the disk on drive DK0: in 2048–byte blocks, rather than the default 512–byte blocks. If you specify an illegal blocksize, the error message ?ILLEGAL BLOCK SIZE is returned. If you specify a blocksize that is too large, the error message ?MAXIMUM MEMORY EXCEEDED is returned.

To specify other than standard density and parity settings when copying magnetic tape, issue the /DENSITY:d and /PARITY:p options. These may be minimally abbreviated to /DE:d and /PA:p. In these options:

d Can Be:	**p Can Be:**
800	ODD
DUMP	EVEN
1600 (phase-encoded)	

If you use a density or a parity that does not appear in this list, the error message ?ERROR IN SPECIFYING DENSITY (or PARITY) is returned.

The following example illustrates /DENSITY and /PARITY:

```
#MT0:/DENSITY:800/PARITY:EVEN = MT1:/DENSITY:DUMP/FC
```

COPY reads tape unit 1 at 800 bits per inch dump mode and writes unit 0 in even parity at 800 BPI.

Several error messages are returned for general command errors. If you misplace or mistype an option, the program prints ?ERROR IN SPECIFYING SWITCHES. If you specify an illegal or nonexistent device, the program prints ?ILLEGAL DEVICE. If you specify anything besides a device in the input or output specification, the program prints ?ILLEGAL INPUT (or OUTPUT) SPECIFICATION. Finally, if you type a command string that the program cannot decode, it prints ?SYNTAX ERROR IN COMMAND STRING.

6.2 Listing Directory of Files: The DIRECT Program

The DIRECT program lists file-related information from a disk directory. The benefits of DIRECT are increased speed and more options compared with other methods of listing directories. DIRECT opens your directory as a file and reads information by immediately accessing the blocks in the directory. Since the program follows pointers through a disk directory, incorrect information may be printed if the pointers are changed during program execution. For example, if a file is opened on the current account by another user after the directory operation is begun, then the information printed may be incorrect.

You run DIRECT by typing the RUN $DIRECT command or by using the CCL command (explained later in this section). When DIRECT runs as a result of the RUN command, it prints a header line and the # character, which acts as a prompt. You can then type a command to DIRECT. If you run DIRECT by the CCL command, include the command to DIRECT in the CCL command.

The general format of the command is:

```
output=input/option(s), input/option(s),...
```

Output is optional and can be a device specification or a disk file specification. If output is not given, the = character is optional and DIRECT prints output at your terminal. If a file type does not appear with an output file name, DIRECT appends .DIR unless you force a null extension by specifying the . character with the file name. Input can be any number of full disk file specifications. The full disk file specification on input can include a device, file name, type and project-programmer number. If a device is not given, DIRECT uses the public structure and denotes it by the SY: specification.

The file name, type, and project-programmer fields can contain * and ? characters to denote wildcard specifications. If no file specification is given or if an * character is given as the file specification, DIRECT processes all files in the directory. DIRECT applies the default interpretations shown for the following specifications for a given directory.

User Types:	Program Interprets As:	Meaning:
null	*.*	All files
*	*.*	All files
*.	*.	All files with null file types
.	*.	All files with null file types
.TYP	*.TYP	All files with type TYP
FILE	FILE.*	All files with file name FILE

With a file specification, you can specify one or more options. If no options appear, DIRECT proceeds as if you had specified /DI. Table 6–3 lists and describes the options.

Table 6–3: DIRECT Options

Type	Format	Meaning
Individual	/NA	List file names only.
	/TY or /EX	List file names and types of each file.
	/SZ /SI	List file name, type, size (in blocks) of each file; prints C if contiguous, P if file is protected (non-deletable), and L if file is located (placed). The /SI option cannot immediately follow the DIRECT CCL command (DIR/SI); /SZ is allowed.
	/AL	List file name, type, and number of blocks allocated to file. The number of blocks listed is always a multiple of the cluster size. If the cluster size is 4, and the actual length of a file is 5 blocks, the blocks allocated is shown as 8 blocks.
	/OP	List file name, type, open status, and access count. Note that the designation U is appended to the access count if the file is opened in UPDATE mode; W is appended if the file allows others with write access. The access count is printed as two numbers separated by a slash. The first number indicates the number of times the file is currently open in any mode but read regardless; the second number indicates the number of times the file is currently open in read regardless mode.
	/RT	Print the name of the run-time system that created the file.
	/PR	List file name, type, and file protection code.
	/LA	List file name, type, and date of last access or update, depending on disk initialization.
	/DA	List file name, type, and date of creation for the file.
	/TI	List file name, type, date and time of day when file was created.
	/CL	List file name, type, and file cluster size.
	/SU	List only summary data to include number of designated files and total number of blocks occupied by designated files.
	/PO	List file name, type, and the position of the file on disk (i.e., the device cluster number of the file's first block). Refer to Section 4.1.4.4.
	/SA	List file name, type, symbolic file attributes, and caching

(continued on next page)

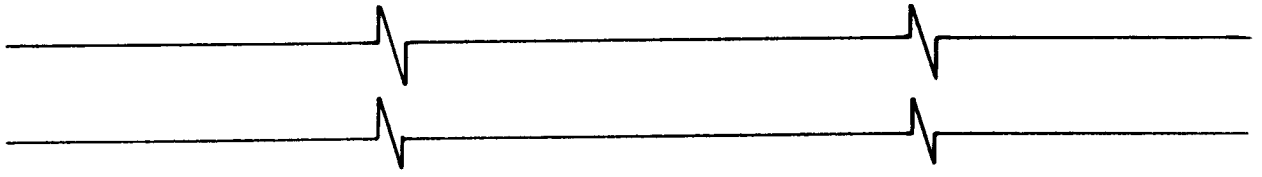
Table 6–3: DIRECT Options (Cont.)

Type	Format	Meaning
	/AT	status. Caching is indicated by one of the following entries: CACHE:ON:RAN; file will be automatically cached randomly when open. CACHE:ON:SEQ; file will be automatically cached sequentially when open. CACHE:OFF:SEQ; file will not be automatically cached, but if cached, it will be cached sequentially.
	/OA	List file name, type, and a series of octal numbers that represent file attributes.
Aggregate	/BR /F	List file names and types with a brief summary message.
	/FU	List headings, file names, types, size, protection code, date of last access, date of creation, time of creation, cluster size, associated run-time system, file position, open status, and summary.
	/DI:S /LI:S /S	List all relevant data to include headings, file names, types, size, protection code, date of last access, date of creation, time of creation, cluster size, associated run-time system, file position, symbolic attributes, open status (if possible to print on the same line), files marked for deletion (see /MD), and summary. (Called slow directory.)
	/DI /LI null	List most important data to include heading, file name, type, size, protection code, creation date, and summary.
	/MD	List all files and flag files marked for deletion by printing an * after the file name. Tentative files are marked for deletion until closed.
	/N	Print only the entries that do not match the file specification given.
General	/HD	Print heading at top of columns on the listing.
	/W	List data across the width of a line rather than one item per line. Useful with large directory listings and individual options. When used alone, /W prints file names and types across the width of a line with a summary message.
	/HE	Print the file DIRECT.HLP which describes the DIRECT program.
	/BK	List the directory for the specified device in reverse order. As a result, the files at the end of the directory appear at the beginning of the listing. If /BK is used to list files in the public structure and multiple disks are in the public structure, the listing reflects reverse order for each disk. DIRECT cannot list directories containing more than 200 files in reverse order.

Consider the following example:

```
RUN $DIRECT
DIRECT V7  RSTS V7  Timesharing
#/DI:S
 Name .Typ     Size    Prot    Access      Date       Time    Clu  RTS     Pos  Op/rr

INTRO .COB        3    < 60> 27-Feb-81 27-Feb-81 02:21 PM    4 ...RSX 12651  0/0
FOO   .CMD        1    < 60> 11-Mar-81 11-Mar-81 11:13 AM    4 ...RSX 12652  0/0
MEMO  .           4    < 60> 26-Jun-81 26-Jun-81 10:18 AM    4 ...RSX 12654  0/0
UG1   .          28    < 60> 19-Jul-81 19-Jul-81 04:12 PM    4 ...RSX 12655  0/0
UG2RMS.          73    < 60> 21-Jul-81 21-Jul-81 03:05 PM    4 ...RSX 12664  0/0
UG2   .          59    < 60> 21-Jul-81 21-Jul-81 03:34 PM    4 ...RSX 12683  0/0
UG    .         138    < 60> 22-Jul-81 19-Jul-81 04:12 PM    4 ...RSX 12698  0/0
SIGNOF.           2    < 60> 23-Jul-81 23-Jul-81 04:52 PM    4 ...RSX 12736  0/0
UG3   .          46    < 60> 24-Jul-81 24-Jul-81 01:12 PM    4 ...RSX 12749  0/0
BILLY .           1    < 60> 31-Jul-81 31-Jul-81 04:32 PM    4 ...RSX 12763  0/0
FCSJMP.LST       31    < 60> 10-Aug-81 10-Aug-81 02:52 PM    4 ...RSX 12766  0/0
 RF:VAR=95   FO:SEQ    USED:31:218      RECSI:95       CC:IMP
FCSVEC.LST       31    < 60> 10-Aug-81 10-Aug-81 02:54 PM    4 ...RSX 12774  0/0
 RF:VAR=96   FO:SEQ    USED:31:454      RECSI:96       CC:IMP
T1    .           7    < 60> 10-Aug-81 10-Aug-81 04:54 PM    4 ...RSX 12782  0/0
UG4   .           5    < 60> 13-Aug-81 13-Aug-81 04:11 PM    4 ...RSX 12784  0/0
RUG4  .           6    < 60> 13-Aug-81 13-Aug-81 04:11 PM    4 ...RSX 12786  0/0
RUG5  .          10    < 60> 17-Aug-81 17-Aug-81 11:27 AM    4 ...RSX 12793  0/0
UG5   .          10    < 60> 17-Aug-81 17-Aug-81 04:07 PM    4 ...RSX 12796  0/0
EDTINI.EDT        1    < 40> 18-Aug-81 18-Aug-81 10:46 AM    4 ...RSX 12804  0/0
OLUG6 .           5    < 60> 18-Aug-81 18-Aug-81 03:38 PM    4 ...RSX 12805  0/0
UG6   .           1    < 60> 18-Aug-81 18-Aug-81 03:44 PM    4 ...RSX 12807  0/0
RUG6  .           1    < 60> 18-Aug-81 18-Aug-81 03:44 PM    4 ...RSX 12808  0/0
UGSNIP.          14    < 60> 18-Aug-81 18-Aug-81 04:51 PM    4 ...RSX 12809  0/0
RUGSNP.          15    < 60> 18-Aug-81 18-Aug-81 04:51 PM    4 ...RSX 12813  0/0
PARITY.           2    < 60> 25-Aug-81 25-Aug-81 03:13 PM    4 ...RSX 12817  0/0
RTKB1 .          44    < 60> 28-Aug-81 28-Aug-81 04:12 PM    4 ...RSX  5037  0/0
```



```
RTKB2 .          82    < 60> 28-Sep-81 28-Sep-81 10:10 AM    4 ...RSX  4147  0/0
TKB10 .BAK       63    < 60> 28-Sep-81 28-Sep-81 12:03 PM    4 ...RSX  1179  0/0
TKB10 .          66    < 60> 28-Sep-81 28-Sep-81 12:35 PM    4 ...RSX   359  0/0
TKB11 .          69    < 60> 28-Sep-81 28-Sep-81 12:45 PM    4 ...RSX  6862  0/0
RTASK .         636    < 60> 28-Sep-81 28-Sep-81 12:47 PM    4 ...RSX  4647  0/0
MAIL12.TMP       2    < 60> 28-Sep-81 28-Sep-81 03:18 PM    4 ...RSX   982  0/0
SEND12.TMP       1    < 60> 28-Sep-81 28-Sep-81 03:19 PM    4 ...RSX  3391  0/0
DIREXM.          25    < 60> 28-Sep-81 28-Sep-81 03:21 PM    4 BAS4F   6983  1/0W

Total of 3317 blocks in 148 files in SY:[3,156]
```

In this example, you run the DIRECT program and select the /DI:S option. This option causes the program to list all of the relevant information on the files in your account. This information is printed under the following headings:

Name.Typ	the file name and type.
Size	the size (in blocks) of each file.
Prot	the protection code.
Access	the date of last access.
Date	the date of creation.
Time	the time of creation.
Clu	the file cluster size.
RTS	the file's run-time system.
Pos	the position of the file on disk (the device cluster number of the file's first block).
Op/rr	the number of times the file is currently open (Op) in any mode except read regardless; and the number of times the file is currently open in read regardless mode (rr).

Note that those files which have attributes contain a description of the attributes on the line following the file. Refer to Section 6.2.2 for the meaning of the listed attributes.

To list a directory at the line printer, specify the device designator and options in the command. For example:

```
*LP0:=*.BA?/F/W
```

The command lists file names and types of all files with BA as the first 2 characters of the file type. DIRECT formats the data across the width of the line printer paper.

To list a directory of a specific file, type the file name and type (with options) in the command.

To list directories of several accounts and place them in a disk file, specify the project-programmer field and the disk file specification in the command. For example:

```
*PROJ=[120,*]/DI
```

DIRECT creates the file PROJ.DIR in the current account and writes, to the file, directory listings of all accounts with project number 120.

An error encountered in a command causes DIRECT to print a message followed by the # character. You must retype the entire command correctly. The messages are described in Table 6–4.

Table 6–4: DIRECT Program Error Messages

Message and Meaning
?DEVICE NOT DIRECTORY STRUCTURED Device specified does not use a directory for file access.
?DIRECTORY OF dev:[n,m] IS EMPTY DIRECT finds the account [n,m] on device dev: contains no entries.
?DISK PACK IS NOT ON–LINE The pack or cartridge referred to is either not mounted or is off-line.
?ILLEGAL FILE NAME <filename> The file specified by <filename> contains a logical device name which you have not reserved by the ASSIGN command.
?ILLEGAL INPUT FILE SPEC <file spec> The file specification indicated by <file spec> generates the error ?ILLEGAL FILE NAME.
?ILLEGAL SWITCH text File specification contains an undefined switch indicated by text.
?INVALID DEVICE SPECIFICATION The device specification is invalid or the device referred to does not exist on the system.
?NO DIRECTORY FOR [n,m] ON dev: DIRECT cannot find an account for user account [n,m] on the device dev: or else DIRECT encounters a protection violation.
?NO HELP AVAILABLE The file DIRECT.HLP is not in the system library account.
?NO SUCH FILE AS <file spec> ON [n,m] DIRECT cannot find the requested file indicated by <file spec> in the account [n,m].
?OUTPUT FILE MUST BE IN THE USER'S AREA DIRECT does not create an output disk file in another account if you are not privileged.
?TOO MANY FILES FOR INVERTED DIRECTORY LISTING DIRECT limits use of the /BK option to accounts with less than 200 files.

6.2.1 DIRECTORY as a CCL Command

The following commands show some useful methods of requesting listings with the standard CCL command DIRECTORY or its abbreviation DIR. To list the file names and file types of all files in the current account, type the following command:

```
DIR /W
```

DIRECT prints the listing at your terminal and lists the information across the width of the page. To list at another device the file names, types, sizes, protection codes and creation dates for certain files in the current account, type the following command:

```
DIR LP1:=*.BAS
```

DIRECT prints the listing of all BASIC–PLUS source files on line printer unit 1. The READY message indicates DIRECT has completed printing. To obtain a reverse listing, type the following command:

```
DIR /DI/BK
```

DIRECT prints, at the current terminal, directory and summary information in reverse chronological order. To print the file DIRECT.HLP on the line printer, type the following command:

```
DIR LP1:=/HE
```

6.2.2 File Attributes

File attributes are file description data required by RMS–11 software. RMS–11 is used by COBOL and BASIC–PLUS–2, for example. The attributes of a given file are recorded in the User File Directory (UFD) entry for that file. Some of the more important file attributes are:

- file organization
- record type
- record size
- file size
- location of the next free byte within the file

You can list the attributes of one or more files with the DIRECT program options /SA, /AT, /OA, and /S (see Section 6.2) or with the PIP program directory options :OA, :SA, :S, and :AT (see Section 6.11). The options list the file's attributes symbolically or octally on the line below the file name. The symbolic attribute fields are defined as follows:

RF — is record format and, if given, the maximum record length. The possible formats are UDF (undefined), FIX (fixed), VAR (variable), VFC (variable / fixed control), and STM (stream).

FO — is file organization; SEQ (sequential), REL (relative), and IDX (indexed).

USED	is the number of blocks currently in use followed by the number of bytes being used in the last block.
RECSI	is the size in bytes of the largest record in the file.
CC	is carriage control; IMP (implied) or FOR (FORTRAN).
NOSPAN	is printed if records are not allowed to span block boundaries.
BK	is printed if the file organization is relative or indexed and indicates the bucket size in blocks.
EX	if the default is not used, indicates the file type quantity.
HS	if the default is not used, indicated the fixed header size.

For example:

```
DIR /SA
SY:[2,211]
RMSTST.B2S
RMSTST.OBJ
 RF:VAR=128    FO:SEQ   USED:7:22      RECSI:128    CC:IMP
PROOF .BAS
PFILE .DAT
 RF:FIX=56     FO:IDX   USED:8:0       RECSI:56     CC:IMP
RDPLN1.DOC
CHTST .BAS
CHTST1.BAS
CHTST1.B2S
CHTST1.OBJ
 RF:VAR=128    FO:SEQ   USED:2:396     RECSI:90     CC:IMP
CHTST1.CMD
CHTST1.ODL
CHTST1.TSK
 RF:FIX        FO:SEQ   USED:8:0       RECSI:512
FILE2 .BAS
FILE  .DIF
FILE1 .BAS
FIL   .DIF
INSRT .MAC
 RF:STM        FO:SEQ   USED:3:234     RECSI:63     CC:IMP
INSRT .OBJ
 RF:VAR        FO:SEQ   USED:1:144     RECSI:42     CC:IMP
INSRT .LST
 RF:VAR        FO:SEQ   USED:5:456     RECSI:94     CC:IMP
UBAK01.RNO
 RF:STM        FO:SEQ   USED:121:159   RECSI:121    CC:IMP
UBAK01.DOC
 RF:STM        FO:SEQ   USED:134:377   RECSI:80
RDPLN1.RNO
 RF:STM        FO:SEQ   USED:26:164    RECSI:79     CC:IMP
RDPLN2.RNO
 RF:STM        FO:SEQ   USED:24:475    RECSI:110    CC:IMP
RDPLN2.DOC
 RF:STM        FO:SEQ   USED:26:286    RECSI:80
```

The OA options lists the file's attributes as a series of 6–digit octal numbers below the file name. For example:

```
DIR /OA
SY:[2,211]
RMSTST.B2S
RMSTST.OBJ
 001002 000200 000000 000007 000000 000007 000026 000000 000200
RMSTST.CMD
RMSTST.ODL
RMSTST.TSK
 000001 001000 000000 000222 000000 000223
RMSTST.MAP
 000002 000167 000000 000211 000000 000211 000354
PROOF .BAS
PFILE .DAT
 001041 000070 000000 000007 000000 000010 000000 000001 000070
```

Normally, such a listing shows seven 6–digit octal numbers (called words) for each file that possesses attributes. RMS–created files show nine or ten words. The additional words contain information pertinent to RMS–11. Table 6–5 describes these words and their meanings.

For example, suppose a directory listing contains an entry as follows:

```
FILE .DAT        7   < 60> 15-Sep-81  15-Sep-81  02:10 PM   8  RSX   0
 001041 000070 000000 000010 000000 000007 000000 000001 000070 000010
```

The first word, 001041, indicates:

000001 – the file has fixed length records
000040 – the file has indexed organization
001000 – carriage control is implied

The second word, 000070, indicates that the number of bytes in each record is 70 (octal) (56 decimal).

The third and fourth words, 000000 and 000010, indicate that the file size is 10 (octal) blocks (8 decimal).

The fifth and sixth words, 000000 and 000007, indicate that the first 7 blocks in the file are in use.

The seventh word, 000000, indicates that no bytes are in use in the seventh block.

The eighth word, 000001, indicates that the bucket size is 1 block and the fixed header length is 2 bytes.

The ninth word, 000070, indicates that the record size is 70 (octal) bytes (56 decimal).

The tenth word, 000010, indicates that the file is allocated 10 new blocks (8 decimal) each time it is extended.

Table 6–5: File Attributes

Word	Meaning
1	Low order digit – Record Format 000000 – undefined 000001 – fixed length records 000002 – variable length records 000003 – variable with fixed control 000004 – stream format records Second digit – File organization 000000 – sequential organization 000020 – relative organization 000040 – indexed organization Third and fourth digit – Record attributes 000400 – FORTRAN carriage control 001000 – implied carriage control 004000 – no span Fifth and sixth digits – Not used
2	The size in bytes for fixed-length records or the size of the largest record for variable-length records.
3 and 4	A two-word integer that represents the file size. Word 3 is high order, word 4 is low order.
5 and 6	A two-word integer that represents the number of blocks that are currently in use for data. Word 5 is high order, word 6 is low order. Note that words 3 and 5 are non-zero only if the file is larger than 65535 blocks.
7	The number of bytes in use for the last block used. Note that the size of a file with n full blocks can be described as n blocks with 512 characters in the last block or as n + 1 blocks with 0 characters in the last block.
8	Low byte The bucket size in blocks in the range of 1 to 32. If this byte is 0, the bucket size is 1 block. High byte The fixed header size in bytes in the range of 1 to 255. If this byte is 0, the header size is 2 bytes.
9	The maximum record size in bytes. If the file contains fixed-length records, this word contains the record length. If this word is 0, no maximum size is set for the file.
10	The number of blocks to which the file is extended by default. That is, the content of this word represents the number of blocks that are added to this file each time the file is extended.

File attribute data is processed by RMS–11 and all languages that use RMS–11. BASIC–PLUS and FORTRAN IV do not use RMS–11 and therefore do not correctly process files that have attributes. Thus, programs that are written in BASIC–PLUS or FORTRAN IV, except those that are especially programmed to handle file attributes, cannot be used with files produced by RMS–11. FORTRAN–77 does use file attributes but does not use RMS–11.

System programs that do handle file attributes correctly include the BACKUP package, the Spooler package, and PIP.

The system program PIP (see Section 6.11) is the only program generally used that preserves file attributes and run-time systems in disk-to-disk transfers within one system. When PIP copies a disk file to a non-disk device, it performs file format conversions, either automatically or as you specify through switch specifications. PIP can also perform file format conversions on a copying operation from a non-disk device to a disk file, and create the appropriate file attributes in the process.

To move files between systems using magnetic tape as the transportation medium, you have the following alternatives:

- to use the RSTS/E programs BACKUP or PIP for transfers between two RSTS/E systems.
- to use the RMS BACKUP package for transfers between a RSTS/E system and a system in the RSX–11 family, when the files involved are produced by RMS–11.

More information on RMS–11 and file attributes can be found in the *RMS–11 User's Guide*. More information on the program PIP can be found in Section 6.11 of this manual.

6.3 Comparing Files: The FILCOM Program

The FILCOM (File Compare) system program allows you to compare two ASCII files, line by line, and identify differences. When the program runs, you specify the files you wish to compare and, optionally, one or more switches that direct that comparison.

To invoke the FILCOM program, type:

```
RUN $FILCOM
```

If the invocation is successful, FILCOM prints a two line identification header as follows:

```
FILCOM V7  RSTS V7  TIMESHARING
FILE COMPARISON PROGRAM 30-Sep-81 11:59PM
```

Following the identification header, FILCOM prints a prompt:

```
OUTPUT TO <KB:> ?
```

In answer to the prompt, you can type one of three responses.

1. The device designator of the peripheral device to which FILCOM writes the comparison data.

   If you only specify a device or type a RETURN key to accept the user terminal as a default, FILCOM prints an additional series of prompts. These prompts request the files you wish compared and allow you to set the parameters of the comparison. These prompts are described in Section 6.3.1. Note that you can also specify a switch, or switches, as described in Table 6–6 in answer to this prompt. These switches provide an alternative method for setting comparison parameters.

2. A single command line that contains all of the information that FILCOM needs to make the file comparison.

   The command line takes the form:

   ```
   output=input1,input2/sw
   ```

   If you specify the full command line, FILCOM does not issue any other prompt and begins the file comparison. Use of the command line allows you to specify wildcards in the file specifications and to specify one or more optional switches. Section 6.3.2 describes the command line specification and Table 6–6 describes the switches.

3. A command line that contains an indirect command file name. The indirect command file is an ASCII text file that you create with PIP or an editor. The file must contain all of the information that FILCOM needs to make the file comparison.

The command line takes the form:

```
OUTPUT TO <KB:> ?@filename
```

where filename represents the indirect command file. The indirect command file can contain a single line command as described in the previous response 2, or it can contain multiple line answers to the FILCOM prompts as described in response 1. If the command file contains answers to the FILCOM prompts, each prompt must be answered in sequence as described in Section 6.3.1. Note that you cannot nest another indirect command file within the original response. If FILCOM encounters an at sign (@) in the command file, it processes it as an assignable account specification.

After a file comparison is complete, FILCOM returns to the initial prompt for additional input. To exit from FILCOM, type CTRL/Z in response to the OUTPUT TO prompt.

6.3.1 FILCOM Prompts

If you answer the initial FILCOM prompt with a device designator or a RETURN key, the FILCOM program prints a series of additional prompts. These prompts request the following information:

1. The files you wish to compare.
2. The number of successive lines in those files that constitute a match.
3. Whether FILCOM will consider BASIC–PLUS continuation lines as part of a numbered line.
4. Whether FILCOM will compare blank lines.

Note that you can specify one or more of the switches described in Table 6–6 in response to FILCOM's initial prompt. You can use a switch specification to override the appearance of prompts and to provide additional instructions to the FILCOM program.

Following the initial prompt, FILCOM prints requests for the filenames of the ASCII files to be compared. For example:

```
OUTPUT TO <KB:> ? RET
INPUT FILE #1? SORT1.BAS
INPUT FILE #2? SORT2.BAS
```

In this command series, you specify that the file SORT1.BAS is to be compared to SORT2.BAS and the result of that comparison is to be printed at your terminal. When you specify the files to be compared, include the file type. Note that you can use the wildcard specification * and ? to designate the input files. The use of wildcards is described in Section 6.3.3.

The prompt that follows the input file requests asks you the number of lines in the file that will constitute a match. As FILCOM compares the lines in a file, it prints the differences until it comes to a series of lines that are identical. The number of lines that constitute a series is determined by your answer to this prompt. For example:

```
OUTPUT TO <KB:> ? SORT.DIF
INPUT FILE #1? SORT1.BAS
INPUT FILE #2? SORT2.BAS
HOW MANY TO MATCH <3> ? RET
```

In this command series, you specify the creation of a file named SORT.DIF that contains the result of a comparison between the files SORT1.BAS and SORT2.BAS. FILCOM examines the files and records the differences until it finds three lines to match. The RETURN key response accepts the default of three lines.

Following the HOW MANY TO MATCH prompt, FILCOM asks if you wish it to consider BASIC–PLUS continuation lines as part of a numbered program line. For example:

```
OUTPUT TO <KB:> ? RET
INPUT FILE #1? SORT1.BAS
INPUT FILE #2? SORT2.BAS
HOW MANY TO MATCH <3> ? RET
BASIC+ LINES <NO> ? Y
```

In this command series, you specify that FILCOM is to consider continuation lines as part of a numbered program line when it compares SORT1.BAS and SORT2.BAS. Note that the default for this prompt, <NO>, causes FILCOM to compare only single lines. Consider the following program line:

```
100 FOR J=10. TO 15.    &
\   PRINT 3.14*J/2.     &
\   NEXT J
```

If you answer the BASIC+ LINES prompt with Y (for YES), FILCOM compares all three multi-line statements (FOR, PRINT, and NEXT) as a single line. If you accept the NO default, FILCOM only compares the first line (FOR J = 10. TO 15. &) in SORT1 to a single line in SORT2.

The last prompt in the series asks if FILCOM is to compare blank lines. For example:

```
OUTPUT TO <KB:> ? SORT.DIF
INPUT FILE #1? SORT1.BAS
INPUT FILE #2? SORT2.BAS
HOW MANY TO MATCH <3> ? 5
BASIC+ LINES <NO> ? Y
BLANK LINES <NO> ? RET
```

In this command series, you specify that FILCOM is to compare SORT1.BAS and SORT2.BAS and print the differences in SORT.DIF. You instruct FILCOM to match five successive lines, consider BASIC–PLUS continuation lines, and to ignore the presence of blank lines. That is, if FILCOM encounters two blank lines in SORT1 and five blank lines in SORT2, it is not considered a difference.

Section 6.3.4 contains examples of FILCOM comparisons.

6.3.2 FILCOM Single Command Line

In response to its initial prompt, FILCOM accepts a single command line that specifies all files and comparison parameters.

This command line has the following format:

```
OUTPUT TO <KB:> ? output=input1,input2/sw
```

where:

output — is one of the following:

1. A complete RSTS/E file specification containing device, project-programmer number, file name, and type designators.
2. A RETURN key which specifies the default user terminal. If you specify a RETURN key, FILCOM prints the prompts as described in Section 6.3.1.
3. A wildcard specification as described in Section 6.3.3.
4. No specification. If you do not specify a file or equal sign, but rather an input specification only, output is printed on your terminal.

= — required only if a file specification is given for output. Note that an equal sign with no output specification causes FILCOM to output the result of its file comparison to a file on the public structure. FILCOM creates this file in the current account with a file name taken from the first input file and a .DIF file type.

input1, input2 — the two files that FILCOM is to compare separated by a comma. You can describe these files with full file specifications, file names and types, or wildcards as described in Section 6.3.3.

/sw — one or more optional switches as described in Table 6–6.

Table 6–6: FILCOM Switches

Switch	Meaning
/MA[TCH][:n]	specifies the number of lines in the file that will constitute a match (see Section 6.3.1). If you do not specify the number of lines (n), the default is three.
/BL[ANK] /NOBL[ANK]	specifies whether FILCOM will consider blank lines in its comparison (see Section 6.3.1). The default is not to consider blank lines (/NOBL).
/BA[SIC] /NOBA[SIC]	specifies whether FILCOM will consider BASIC–PLUS continuation lines as part of a numbered program line (see Section 6.3.1). The default is not to consider continuation lines (/NOBA).
/PA[TCH]	specifies that FILCOM is to create the output file as a patch file. You can use a BASIC–PLUS APPEND command (see the *RSTS/E BASIC–PLUS Language Manual*) to include the patch file in a succeeding FILCOM operation to cause the resulting output to be equivalent to the second input file.
/AP[PEND]	causes FILCOM to open the output file for APPEND (see above). If this switch is not specified, FILCOM deletes any existing output file of the same name prior to writing differences. With the /AP switch, you can cause FILCOM to write differences into an existing output file.
/SU[MMARY]	causes FILCOM to list the total number of differences found in the files without listing individual lines.
/LI[MIT][:n]	specifies the maximum number of lines that are allowed to differ before FILCOM stops examining the file. If you include the /LIMIT switch but do not specify a limit in n, the default is 60 differing lines.
/CO[MPARE][:m:n]	converts tabs to the appropriate number of spaces and causes FILCOM to compare characters starting in column position m and continuing for n characters. If you include the /COMPARE switch but do not specify m or n, FILCOM defaults m to column position 1 and n to 72 characters.

The use of these switches is not restricted to single line commands. That is, you can specify any of these switches in answer to the OUTPUT TO prompt, regardless of whether you specify the full FILCOM command line in that prompt.

If you do not specify any switches, FILCOM assumes that three lines constitute a match, blank lines will not be compared, and BASIC–PLUS continuation lines will not be considered part of a numbered program line (i.e., /MA:3/NOBL/NOBA). However, if you specify the /PATCH switch, FILCOM does compare blank lines and does consider BASIC–PLUS continuation lines. That is, a /PA specification overrides the normal defaults and sets /BA and /BL.

If you specify a switch in illegal format or a switch that is not one of the legal set, the following error message is printed:

```
ILLEGAL COMMAND LINE /x
```

where /x is the illegal switch.

If you specify the /LIMIT switch, FILCOM ceases to compare the files when a set number of lines are found to be different during a single compare. You can specify the number in the switch or accept the default of 60 lines. Once the limit is exceeded, FILCOM stops comparing the files and prints the following message:

```
LIMIT x REACHED ON LAST COMPARE, SKIPPING REST OF FILE
```

where x is the /LI specification or the default 60. Note that no limit is set unless you specify the /LI switch.

The /COMPARE switch is useful for comparisons of lines of text where the characters beyond a certain column need not be considered. For example, a program that contains comments beginning in column position 80 can be compared and the /CO switch used to ignore the comments. That is, you specify /CO:1:79 to cause FILCOM to convert all tabs in the lines to the appropriate number of spaces (in order to properly count character positions) and compare each line from column 1 to column 79. FILCOM ignores the content of the lines from column 80 to the end of the line.

Consider the following examples.

```
OUTPUT TO <KB:> ? SORT.DIF=SORT1.BAS,SORT2.BAS
```

This command series causes FILCOM to compare SORT1 and SORT2 and report the differences in the file SORT.DIF. FILCOM sets the /MA:3, /NOBA, and /NOBL switches by default.

```
OUTPUT TO <KB:> ? SORT1.BAS,SORT2.BAS/LI:10
```

This command series causes FILCOM to compare SORT1 and SORT2 until the number of differing lines in a single compare exceed 10 or the comparison is complete. FILCOM creates a file named SORT1.DIF in the current account to store the result of the comparison. Note that the /MA:3, /NOBA, and /NOBL switches are set by default.

```
OUTPUT TO <KB:> ? SORT1.BAS,SORT2.BAS/BA/BL/SU
```

This command series causes FILCOM to compare SORT1 and SORT2 and print a summary of the differences on the user terminal. FILCOM is instructed to compare blank lines (/BL) and to consider BASIC–PLUS continuation lines (/BA).

6.3.3 Wildcards

You can specify FILCOM input files as full RSTS/E file specifications or as wildcard characters; FILCOM output files can contain a wildcard as the file name or type but not both. For output, you must specify a file name or type or default both fields as described in Section 6.3.2. The use of wildcard characters * and ? are documented in Section 4.1.3.1. The asterisk (*) wildcard replaces an entire specification field (i.e., file name, type, etc.). The question mark (?) wildcard replaces a character within a field. When wildcards are specified in a FILCOM command line, FILCOM bases its search for files on the first input file specification.

FILCOM uses wildcard specifications for input files in the following manner:

If the Specification Is:	FILCOM
.TYP,.XYZ	compares all files with the type .TYP and all files with the same file names and type .XYZ.
ABC.*,DEF.*	compares all files named ABC with any file type and all files named DEF and the same types.
[5,30]*.*,[5,31]*.*	compares all files in account [5,30] and all files with the same file names and types in account [5,31].
[5,30]*.*,[5,31]*.TYP	compares all files in account [5,30] and all files with the same file names and a .TYP type in account [5,31].
AFILE?.*,BFILE?.*	compares all files whose file name consists of AFILE and any character with any type and all files whose file name consists of BFILE and the same character and the same file type.
.DIF=.ABC,*.DEF	compares all files with the type .ABC and all files of the same file name with the type .DEF. As FILCOM compares each set of files, it creates a series of output files with the type .DIF. The output file names duplicate the file names of each .ABC file compared.

With wildcards, you can cause FILCOM to make multiple file comparisons based on a single command line specification. When such a method is used, FILCOM prints the differences found in each set of files in the standard manner (see Section 6.3.4). In addition, FILCOM prints a summary at the completion of the comparisons. The summary has the form:

```
x DIFFERENCES FOUND IN y FILES OF z TOTAL FILES COMPARED
```

where x is the total number of differences in all files, y is the number of files that contain differences, and z is the total number of files compared based on the wildcard specification.

Wildcard file names for the first input file can only be specified if the file resides on disk. If the first input file is not a disk file, FILCOM prints the message:

```
WILDCARD FILENAME ILLEGAL ON DEVICE x
```

(where x is the specified input device) and returns to the OUTPUT TO prompt. Note that the second input file does not have to reside on disk.

If a wildcard specification would cause FILCOM to create a file name for output that is syntactically incorrect, the program prints the following message:

```
ILLEGAL FILENAME x CREATED FROM y BY z
```

where x is the illegal file name that would have been created, y is the wildcard specification you made for the first input file, and z is the file specification for the second input file. For example:

```
OUTPUT TO <KB:> ? *.DIF=*.TYP,???OLD.TYP
```

would result in an output file name of A OLD.DIF if A.TYP existed as the first input file. Because spaces are not allowed in a file name, the error message is returned.

6.3.4 FILCOM Examples

Assume that the following two files are in your account on the system disk.

```
TEST1TEST2
10 REM A                    10 REM A
20 REM B                    20 REM B
30 REM C                    30 REM C
40 REM D                    70 REM G
50 REM E                    80 REM H
60 REM F                    90 REM I
70 REM G                    100 REM J
80 REM H                    110 REM 1
90 REM I                    120 REM 2
100 REM J                   130 REM 3
110 REM K                   140 REM N
120 REM L                   150 REM O
130 REM M                   160 REM P
140 REM N                   170 REM Q
150 REM O                   180 REM R
160 REM P                   190 REM S
170 REM Q                   200 REM T
180 REM R                   210 REM U
190 REM S                   220 REM V
200 REM T                   222 REM 4
```

(continued on next page)

FILCOM Examples (Cont.)

```
210 REM U                          224 REM 5
220 REM V                          230 REM W
230 REM W                          240 REM X
240 REM X                          250 REM Y
250 REM Y                          260 REM Z
260 REM Z
```

To compare these files, you can run FILCOM as follows:

```
RUN $FILCOM
FILCOM V7  RSTS V7  TIMESHARING
FILE COMPARISON PROGRAM  01-Sep-81 11:14 AM
OUTPUT TO <KB:> ? TEST1.BAS,TEST2.BAS
```

This command line causes output to your terminal and defaults the switches /MA:3, /NOBL, and /NOBA. The output appears as follows:

```
COMPARING: 1) [2,211]TEST1.BAS TO 2) [2,211]TEST2.BAS

********************
1) [2,211]TEST1.BAS
40 REM D
50 REM E
60 REM F
70 REM G
*****
2) [2,211]TEST2.BAS
70 REM G
********************
1) [2,211]TEST1.BAS
110 REM K
120 REM L
130 REM M
140 REM N
*****
2) [2,211]TEST2.BAS
110 REM 1
120 REM 2
130 REM 3
140 REM N
********************
1) [2,211]TEST1.BAS
230 REM W
*****
2) [2,211]TEST2.BAS
222 REM 4
224 REM 5
230 REM W


?3 Differences Found


OUTPUT TO <KB:> ? ^Z
```

FILCOM prints the lines from both files that do not compare followed by the first line that does compare. The first line in each group is the first line that differs and the last line in each group is the first line that is the same. In this example, the first group consists of lines 40 through 70 in TEST1.BAS. FILCOM printed line 40 because it had no match in TEST2.BAS. FILCOM continued to print TEST1 until it found three matching lines (as directed by the /MA:3 default), the first line of which was line 70.

As another example, consider the following two programs.

```
5     EXTEND
10    REM A
20    REM B
30    REM C  &
\     PRINT "LINE 30"
40    REM D
50    REM E  &
\     PRINT "LINE 50"
60    REM F
70    REM G
90    END
```

```
5     EXTEND
10    REM A
20    REM B
30    REM C
40    REM D
50    REM E  &
\     PRINT "LINE 50"
70    REM F
80    REM G  &
\     PRINT "LINE 80"
90    END
```

The following series of commands cause FILCOM to create a file named FIL.DIF.

```
RUN $FILCOM
FILCOM V7 RSTS V7 TIMESHARING
FILE COMPARISON PROGRAM 01-SEP-81 11:54 AM
OUTPUT TO <KB:>? FIL.DIF=FILE1.BAS,FILE2.BAS/BA/MA:2/SU
OUTPUT TO <KB:>? ^Z

READY
```

The file FIL.DIF contains a comparison of FILE1.BAS and FILE2.BAS. The specified switches instruct FILCOM to consider BASIC–PLUS continuation lines as part of a numbered program line and to print differences when more than two lines fail to match. The /SU switch cause FILCOM to print a summary of the differences in FIL.DIF. After completion of the compare, FIL.DIF contains the following information:

```
FILCOM V7 RSTS V7 TIMESHARING
FILE COMPARISON PROGRAM 01-SEP-81 11:54 AM

COMPARING: 1) [2,211]FILE1.BAS TO 2) [2,211]FILE2.BAS

?2 DIFFERENCES FOUND.
```

The following series of commands cause FILCOM to output its comparison of FILE1.BAS and FILE2.BAS to the user terminal. However, the number of lines that is needed to match is four and a full description of the differences is requested.

```
RUN $FILCOM
FILCOM V7 RSTS V7 TIMESHARING
FILE COMPARISON PROGRAM 01-SEP-81 11:54 AM
OUTPUT TO <KB:>? FILE1.BAS,FILE2.BAS/BA/MA:4

COMPARING: 1) [2,211]FILE1.BAS TO 2) [2,211]FILE2.BAS

********************
1) [2,211]FILE1.BAS
30        REM C  &
\         PRINT "LINE 30"
40        REM D
50        REM E  &
\         PRINT "LINE 50"
60        REM F
70        REM G
80        END
*****
2) [2,211]FILE2.BAS
30        REM C
40        REM D
50        REM E  &
\         PRINT "LINE 50"
70        REM F
80        REM G  &
\         PRINT "LINE 80"
90        END

?1 DIFFERENCE FOUND.

OUTPUT TO <KB:>? ^Z
```

6.3.5 FILCOM Error Messages

FILCOM can return the following errors and informational messages.

Message and Meaning

?MAXIMUM MEMORY EXCEEDED

The files being compared are significantly different, and FILCOM has used up all memory available to it for storing the differences. This error is particularly likely to happen if you said "YES" to the BASIC+ LINES question. Two things can help:

1. Use the /LIMIT switch to specify the maximum number of differing lines. This will avoid the error and allow you to see at least the first n lines of differing text.
2. Use the /NOBASIC switch (if you had been using the /BASIC switch). /BASIC can have the effect of making FILCOM's comparison lines significantly longer.

?A NULL LENGTH FILE

The file examined by FILCOM (the first or second input file) is empty.

?LIMIT n REACHED ON LAST COMPARE, SKIPPING REST OF FILE
The number of differing lines in a single compare (n) as specified in the /LIMIT switch is exceeded. FILCOM continues to the next set of files or prints the OUTPUT TO prompt.

?LINE TOO LONG AT LINE n
FILCOM attempted to input a line at program line number n that is greater than 255 characters. The program continues to compare the rest of the lines in the file.

?WILDCARD FILENAME ILLEGAL ON DEVICE x
You typed a wildcard specification for the first input file where that device was not a disk device.

?WILDCARD '*.*' ILLEGAL FOR OUTPUT FILENAME
FILCOM requires either a file name or type in the output file specification. A wildcard can be used for one of these fields (but not both) or both fields can be defaulted.

?OUTPUT FILENAME 'file.type' WOULD OVERWRITE AN INPUT FILE – SKIPPING COMPARE
During a check of the input and output file specifications, FILCOM discovered an output specification that would cause the overwriting of an input file. The operation is aborted.

?ILLEGAL COMMAND LINE x
The FILCOM command line or file specification contains illegal syntax.

?FILE HAS NON–ASCII STREAM ATTRIBUTE – CAN'T READ <filename>
The FILCOM program can only compare ASCII file data. If the file <filename> contains file attribute data that is not ASCII stream, this error is returned. The program continues to the next set of files or prints the OUTPUT TO prompt.

?INPUT FILE #n<filename> NOT FOUND
FILCOM was unable to find the specified input file. The message indicates the first or second input file (#n) and the file name.

?ILLEGAL FILENAME
The file name specification (input or output) is syntactically incorrect.

?ILLEGAL FILENAME x CREATED FROM y BY z
Based on the wildcard input file specification (y and z), FILCOM would have created a file name (x) of illegal syntax. This error is printed and the program continues to the next set of files or prints the OUTPUT TO prompt.

6.4 File Transfer Between Devices: The FIT Program

The system program FIT (File Transfer) allows you to perform file transfers between devices. The transfers that can be performed by FIT are:

1. Transfer files between disk and TU58 DECtape II or between disk and RX01 or RX02 flexible diskette. FIT also allows you to list the directory of a DECtape II or diskette.
2. Transfer files between RSTS/E disk and RT–11 disk or between RSTS/E disk and RT–11 DECtape. FIT also allows you to list the directory of an RT–11 disk.
3. Transfer files from a DOS disk to a RSTS/E disk. FIT also allows you to list the directory of a DOS disk.

TU58 DECtape II and flexible diskette files are maintained in the RT–11 directory structure format. Thus, DECtape II and diskettes are directly transferrable between RSTS/E and RT–11 systems.

To invoke the FIT program, you must be logged into the system and type the following command:

```
RUN $FIT
```

Or, if the system manager has installed FIT as a concise command language (CCL) command, you can type:

```
FIT
```

If FIT is successfully invoked, it prints an identifying message followed by a FIT> prompt. The prompt indicates that FIT is prepared to accept a command line (see Section 6.4.1). To cause FIT to print a help text file, type the /HE switch in response to the prompt. To exit from FIT, type CTRL/Z in response to the prompt, for example:

```
FIT>^Z

Ready
```

6.4.1 Transferring Files with FIT

To transfer a file with FIT, type a command line of the following form in response to the FIT prompt:

```
FIT>[output[/sw]=]input[/sw][/WATCH]
```

where output and input are file specifications and /sw is an optional switch that specifies the structure of the device on which the file resides (RSTS/E,

RT–11, or DOS). If you specify the optional /WATCH switch, FIT logs all file transfers on the terminal. Note that you can abbreviate the /WATCH switch to /W.

The input and output file specifications have the following form:

```
dev:[proj,prog]name.typ/PROT:n
```

where:

dev: is a device specification. If none is specified, the default is the public structure. The output device must not be a DOS disk. The input device can be any disk with a valid RSTS/E, RT–11, or DOS directory structure, or any TU56 DECtape, TU58 DECtape II, or flexible diskette with RT–11 directory structure.

proj,prog is a project-programmer number. If none is specified, the default is your current account. FIT ignores the project-programmer number if the device has an RT–11 directory structure. Wildcards can be used as described in Section 6.11.2.

name.typ is a file name and type. If none is specified for output, FIT uses the input file name and type. An input file name must be specified. Wildcards can be used as described in Section 6.11.2.

/PROT:n specifies a protection code. If none is specified on output, FIT assigns your current default protection code. FIT ignores any input file protection code specification.

The legal input devices for a FIT file transfer are as follows:

- Any disk with a RSTS/E directory structure
- RX01 or RX02 flexible diskette with an RT–11 directory structure
- TU58 DECtape II with an RT–11 directory structure
- Disk or TU56 DECtape with an RT–11 directory structure
- Disk with DOS directory structure

The legal output devices (except for DOS input) for a FIT file transfer are as follows:

- Any disk with a RSTS/E directory structure
- RX01 or RX02 flexible diskette with an RT–11 directory structure

- TU58 DECtape II with an RT–11 directory structure
- Disk or TU56 DECtape with an RT–11 directory structure
- Line printer, keyboard, etc.
- DECtape with a RSTS/E directory structure

If the input device is a DOS disk, the legal output devices are as follows:

Disk or DECtape with a RSTS/E directory structure
Line printer or keyboard

The switch specification /sw can be one of the following:

/RT11 The file is on an RT–11 directory structured device. When this switch is specified, FIT will ignore files with a .BAD file type on transfers and deletes. For squeeze (see /SQ switch, Section 6.4.2.2), when FIT finds a file type of .BAD, it will print a warning message and abort the squeeze.

/DOS The file is on a DOS format disk (legal only for an input file specification).

/RSTS The file is on a RSTS/E disk or device. This switch is illegal with flexible diskette and TU58 DECtape II.

These switches can be abbreviated to the first two letters (for example, /RT, /DO, and /RS).

If you do not specify a switch on the input or output specification, FIT attempts to determine the directory structure of the device. If the device does not contain a valid RSTS/E, RT–11, or DOS directory structure, FIT prints the following error:

```
?NOT A RECOGNIZED DIRECTORY STRUCTURE
```

If this error occurs on the output device, it indicates that the device has not been properly initialized. To initialize (or zero) an RT–11 structured device, use the /ZE switch as described in Section 6.4.2.2. To initialize any other device, use PIP as described in Section 6.11.

If the specified input file does not exist on the device, FIT prints the following error:

```
?CAN'T FIND FILE OR ACCOUNT
```

If you do not specify a switch on the output specification, FIT assumes a default directory structure based on the type of device. That is, if the device

is TU58 DECtape II or flexible diskette, FIT assumes RT–11 directory structure; if the device is disk or TU56 DECtape, FIT assumes RSTS/E directory structure.

If the output device has an RT–11 directory structure, the following errors are possible:

?DIRECTORY OVERFLOW

The output device has no space for directory entries. It may be possible for you to use the /SQ switch (see Section 6.4.2.2) to free more space for the directory. If this is not successful, you must delete some of the files on the device before more can be added. Note that if you intend to have many small files on the device, you should use the /N switch to allocate more than the default number of directory segments (see Section 6.4.2.2) when the device is initialized.

?DEVICE FULL

The output device has no free space for the file transfer. It may be possible for you to use the /SQ switch (see Section 6.4.2.2) to create enough contiguous free space for the additional files. If this is not successful, you will have to delete enough files from the output device to add the file(s) from the input device.

These errors are generated prior to the transfer of any data to the RT–11 structured device. If an error is encountered during a transfer, a RSTS/E error is reported and the transfer is aborted. Because the directory entry for the RT–11 file is not finalized until the transfer is complete, an error during transfer causes the space reserved for the file to be freed.

If an error is encountered during the open or transfer phase of a RSTS/E file operation, the appropriate RSTS/E error (see Appendix A) is generated and the transfer is aborted.

6.4.2 Maintaining RT–11 Format

In addition to transferring files to and from an RT–11 directory structured device, FIT can perform the following operations:

1. You can use the /ZE switch (see Section 6.4.2.2) to zero or initialize a device as an RT–11 device. This operation creates an RT–11 directory structure on the device and must be performed prior to transferring files to and from the device.
2. You can use the /LI switch (see Section 6.4.2.2) to obtain a directory listing of the RT–11 device.
3. You can use the /DE switch (see Section 6.4.2.1) to delete a file on an RT–11 device.

4. You can use the /SQ switch (see Section 6.4.2.2) to compress (squeeze) the data and directory of the device and allow more files to be stored on that device. This operation is useful if the available space on the device is fragmented.

Note that the RT–11 operating system requires that files used to contain bad blocks have a .BAD file type. The following procedure must be used to transfer files between RSTS/E and RT–11 systems on bad block disks:

1. A disk that contains bad blocks must be initialized on an RT–11 system before FIT can process it. The bad blocks must be located and contained in files with .BAD types.

2. Whenever the bad block disk is referenced in the FIT command line, the /RT switch (see Section 6.4.2.2) must be included with the file specification.

Using this procedure, you cause FIT to ignore any files with a .BAD type on transfer to and from RT–11 devices. FIT also ignores .BAD files on delete operations. During squeeze operations, FIT prints an error message and aborts the operation if it detects a .BAD file.

6.4.2.1 File Deletion on an RT–11 Format Device — To delete a file on an RT–11 directory structured device, enter the following command format in response to the FIT> prompt:

```
FIT>dev:filename.typ/DE[/WATCH]
```

where dev: identifies the device and filename.typ specifies the file to be deleted. Note that wildcards can be used in the file name and type specifications. FIT will ignore any request to delete a file with the file type .BAD when the /RT11 switch is specified. The /WATCH switch is optional and, if specified, causes FIT to log all file deletions at the terminal. The /WATCH switch can be abbreviated to /W. If the specified device is not RT–11 directory structured or if the file does not exist, FIT prints an error message. If an error is encountered by FIT as it accesses the device, a RSTS/E error is reported.

6.4.2.2 RT–11 Device Switches — You can perform zeroing, compressing, and listing operations on RT–11 directories by entering the following command format in response to the FIT> prompt:

```
FIT>dev:/sw[/sw][/sw]
```

where dev: indicates the device on which the operation is to be performed and /sw specifies the operation. The switches you specify can be one or more of the following:

/ZE [:m] Zero (initialize) the device and build an RT–11 directory structure. If :m is specified, m extra words of directory information for each file is added to the minimal RT–11 directory structure. If m is not specified, FIT adds 13 extra words by default to the directory structure. These words are used to store the run-time system name and file attributes. This switch can be abbreviated to /Z.

Note that RT–11 systems currently ignore these extra directory entries. However, if future releases of RT–11 attach significance to these words, a specification of /ZE:0 may be required for RT–11 files that are read by RT–11 systems. FIT currently ignores extra directory information unless there are 13 extra words (this may change in future releases). Therefore, you should only specify values of 0 or 13 in the /ZE switch.

NOTE

FIT will not prevent an RT–11 disk with bad blocks from being zeroed. Zeroing such a device will destroy the information it contained about bad blocks. An RT–11 bad block disk should never be zeroed under FIT, but all non-bad files may be deleted with a wildcard delete operation using the /RT11 switch.

/N:n Used with the /ZE switch to create n directory segments on the initialized device. Each segment consists of two 512–byte blocks. The value of n must be in the range of 1 to 31 and determines the size of the directory on the device (and, therefore, the number of files that can be stored on the device). The number of files that can fit on a device is determined by the following formula:

$$n*INT(507/(m+7))$$

where n is the number of directory segments and m is the number of extra words for each directory entry (that is, /N:n and /ZE:m, respectively).

Note that you may have to compress the device with the /SQ switch before the maximum number of files is reached.

If you do not specify n, FIT applies defaults based on the device type. These defaults are indicated in the following

table. The table also lists the maximum number of files allowed on a particular device corresponding to your specification in /ZE:n.

Device	Default Number of Directory Segments		Max. Number of Files	
	/ZE:0	/ZE:13	/ZE:0	/ZE:13
TU58 DECtape II	4	10	288	250
RX01/RX02 Diskette	4	10	288	250
RK05 disk	16	31	1152	775
RK06 disk	31	31	2232	775
RK07 disk	31	31	2232	775
RL01/RL02 disk	31	31	2232	775

/LI or /DI — List the directory of the device. The information listed includes file names, types, file size, protection codes, dates of creation, and position on the device. If the device has 13 extra words for each directory entry (/ZE:13, the default value), the run-time system name and file attributes are also printed. Those sections of the device which are free space are marked <UNUSED> in the listing. To list a specific directory, include a file name and type between the device specification and the /LI or /DI switch. Note that the file name and type specification can contain wildcards. The /LI switch can be abbreviated to /L.

/RT11 — Used with the /LI or /DI switches. /RT11 indicates that the device must have RT–11 directory structure. That is, if this switch is not present and the device is a RSTS/E or DOS disk, a directory of the disk is listed. If this switch is present and the device is not an RT–11 device, an error is returned. This switch can be abbreviated to /RT.

/SQ — Compress (squeeze) the device. This switch allows you to compress the files that are currently on the output device and make room for files that are to be transferred. If the /RT11 switch is also specified, FIT will first inspect the device to see if it contains any files with the file type .BAD. If so, FIT prints a warning message and aborts the squeeze operation. The space on RT–11 devices that is allocated to the directory or files can become fragmented. If this is the case, an attempt to add another file to the device can result in a ?DIRECTORY OVERFLOW or ?DEVICE FULL error. You can use the /SQ switch to compress the contents of the device and provide enough room for the additional files. Note that if a squeeze operation is interrupted, data may be lost on the device being squeezed. Thus, FIT ignores any CTRL/C typed at the keyboard during a squeeze operation.

Note that because of the manner in which RT–11 directory entries are made, you may need to compress the device to use the directory space even if files are added contiguously without deletions.

NOTE

RT–11 format DECtape (TU56) is much slower to access than RSTS/E format DECtape and thus should be used only when necessary to transfer between RSTS/E and RT–11 systems. For example, a FIT squeeze operation (/SQ) on an RT–11 DECtape can take up to an hour or longer.

6.4.3 DOS Disk Directory Listings

To list the directory of a DOS disk, enter the following command format in response to the FIT prompt:

```
FIT> dev:[proj,prog]filename.typ[/DO[S]] /L[I]
```

where dev: is the device designator of the DOS disk, [proj,prog] is the account number (UIC) of the directory to be listed, and filename.typ indicates the file(s) to be listed. Note that the project-programmer and filename.typ fields can contain wildcards. The switch /DI can be used in place of /L or /LI. If the /DOS switch is omitted, FIT lists the directory of the device whether it DOS, RSTS/E, or RT–11 directory structured. If the /DOS switch is present, an error is returned if the device is not a DOS disk.

The listing generated by this operation includes the following information: account numbers (UICs), filenames, file types, file length, protection codes, dates of creation, the first block numbers of the files, and whether or not the file is contiguous.

6.5 Diskette Transfer: The FLINT Program

The system program FLINT (FLexible diskette INTerchange) allows you to transfer information from IBM* diskettes (flexible diskettes) to standard disk files that RSTS/E can read (for example, DK:, DP:, SY:). Conversely, FLINT also allows the transfer of information from RSTS/E-readable disks to the diskettes that an IBM system can read. Either of these two transfers may involve multiple diskettes, referred to as volumes in the dialogue and in its description here. FLINT can transfer IBM diskette information to a RSTS/E disk only. For information on the transfer of files between RSTS/E disks, diskettes, and RT–11 format diskettes, see Section 6.4.

NOTE

FLINT can only process single density diskettes. FLINT cannot process double density diskettes on an RX02 drive.

FLINT also allows you to obtain a directory of an IBM diskette; this directory contains information related to the unique data format of an IBM diskette.

To perform these transfers and to print the directory of an IBM diskette, FLINT communicates with you by dialogue. The program asks you a series of questions, which you answer one at a time. There are two different dialogues for the transfer operations, IBM–to–RSTS/E and RSTS/E–to–IBM. There are also dialogues for the printing of an IBM directory and zeroing and erasing a diskette.

6.5.1 Running FLINT: The Initiation Commands

To run the FLINT program, you type:

```
RUN $FLINT
```

FLINT identifies itself and prints a number sign (#) to show its readiness to accept one of five initiation commands. Each of these initiates a different dialogue, as follows:

Command	initiates the dialogue for:
/DIRECTORY	listing an IBM diskette directory.
/TORSTS	transferring from an IBM diskette to a RSTS/E disk file.
/TOIBM	transferring from a RSTS/E disk file to an IBM diskette.
/ZERO	initializing the directory of a diskette.
/ERASE	initializing the directory and erasing all data on a diskette.

*IBM is a trademark of International Business Machines Inc.

Each command may be abbreviated, at the minimum, to its first three letters: /DIR, /TOR, /TOI, /ZER, and /ERA. You can also run FLINT by the CCL command FLI[NT] if the FLINT command was installed on your system at startup.

6.5.2 Listing the Directory of an IBM Diskette

Typing /DIR in response to FLINT's number sign (#) prompt will cause FLINT to print a dialogue of two questions. In the following description of the dialogue, FLINT's questions are followed by explanations.

```
OUTPUT TO?
```

This question asks you where you wish FLINT to print the directory of the diskette(s). If you wish it printed at your terminal, you give a null response (i.e., press RETURN). If, however, you wish to save the directory in a file, respond with an output file specification of the form:

dev:[proj,prog]name.typ/PROTECT:n

You must specify at least the device and file name. By default, [proj,prog] is the current account and the protection code is 60. Note that wildcard characters (? or *) cannot be used to designate project-programmer numbers.

```
DIRECTORY OF?
```

This question asks you to specify the diskette(s) for which the directory will be printed. A null response specifies the diskette DX0:. Other disk specifications must be typed in the form:

DXn: or n

where n = 0 to 7

Multiple diskette specifications must be separated by commas.

The following example is a listing of an IBM diskette directory obtained by FLINT:

```
DIRECTORY OF DX1:

DSN   BRL   BOE    EOE    EOD BI MVI VSN

DATA 080   01001 73026 01001

TOTAL OF 1 DATA SET ON DX1:
```

The abbreviated headings on the second line are listed and defined here, reading from left to right:

1. DSN — Data Set Name: the 1–8 character name you have given the data set (corresponds to file name in RSTS/E).
2. BRL — Block/Record Length: in each 128–position (byte) sector, the number of positions (bytes) containing data.
3. BOE — Beginning Of Extent: the address of the first sector in the data set; output is in the form tt0ss, where tt is track number and ss is sector number.
4. EOE — End Of Extent: the address of the last sector reserved for this data set (in the same format as BOE above).
5. EOD — End Of Data: the address of the next unused sector within the data set extent.
6. BI — Bypass Indicator: A blank in this field means the data set is intended for processing; a B means it is not.
7. MVI — Multi-volume Indicator: A blank in this field means a data set is wholly contained on this diskette; a C means that a data set is Continued on another diskette; an L means that this diskette is the Last on which a continued data set resides.
8. VSN — Volume Sequence Number: the sequence of volumes in a multi-volume data set. Blanks indicate that volume sequence checking is not to be performed.

6.5.3 Transferring IBM Diskette Data to RSTS/E

Type /TOR in response to FLINT's number sign (#) prompt to cause FLINT to print the dialogue for transferring diskette data to a standard RSTS/E disk file (DK0:, for example). In the following description of that dialogue, FLINT's questions are followed by explanations.

```
OUTPUT TO?
```

This question asks you to specify the RSTS/E file that is to receive data from the diskette(s). Respond with an output file specification of the form:

dev:[proj,prog]name.typ/PROT:n[/NH]

By default, dev: is the system disk, [proj,prog] is the current account, and the protection code is 60. Note that the device must be a disk (DP1:, for example) and that wildcard characters (? or *) cannot be used to designate project-programmer numbers. The optional switch /NH suppresses the printing of the 6–byte logical header (see Section 6.5.3.2).

```
TRANSLATE FROM EBCDIC TO ASCII <YES>?
```

This question asks you if you wish the diskette data to be translated from its current EBCDIC format, unreadable to RSTS/E, into ASCII format, readable to RSTS/E. As the default—enclosed in angle brackets—indicates, pressing RETURN will cause the data to be translated. If you type NO, FLINT will not translate the data.

```
INPUT FROM?
```

This question asks you to specify the diskette(s) from which data will be transferred to the output file specified in answer to the OUTPUT TO question. You can specify such an input diskette by typing a response of the form DXn: or simply n, where n is a device number from 0 to 7. Such a response will cause FLINT to transfer the first data set on the specified diskette. If, however, you give a null response (i.e., press RETURN), FLINT transfers the first data set from DX0:.

If you wish FLINT to transfer a specific data set, give a response of the form DXn:dsn where dsn is the data set name, composed of one to eight ASCII characters, including blanks (spaces and tabs). Since these blanks are recognized as part of the IBM data set name, care should be taken in their use. If FLINT cannot find the specified data set, it prints the message ?FILE NOT FOUND and repeats the INPUT FROM question.

If you do not specify a data set name, FLINT prints a message of the form:

```
dsn IS THE DATA SET BEING TRANSFERRED
```

where dsn is the name of the first data set on the first diskette specified.

FLINT examines the data set label—a header with statistical information—to determine if the data set resides on more than one diskette. If it does, and if you have specified only one diskette, FLINT displays the message/question:

```
dsn RESIDES ON MORE THAN ONE DISKETTE -
DO YOU WISH TO CONTINUE <NO>?
```

If you respond by typing YES, FLINT will proceed to its next question. As the default [NO] indicates, however, a null response causes no transfer, and instead causes FLINT to print its # prompt.

FLINT next computes the blocking factor to be used in the transfer of the current data set, and prints a message expressing that factor as the ratio of IBM to RSTS/E records. This message has the form:

```
N XXX CHARACTER IBM RECORDS = 1 RSTS RECORD
```

FLINT, after printing this message, proceeds with the transfer, extracting data from each specified input device.

FLINT computes this blocking factor by determining how many of the 128 positions (bytes) in each diskette sector contain data in the form of characters. This count is the IBM record size, and is used to divide the RSTS record size (512). The product of this division is printed as N in the above message. For example, if each IBM record contains 80 data characters, FLINT will find that 6 full IBM records can be placed in a 512–byte "RSTS record" (512/80 = 6.4). Thus, "6 80 CHARACTER IBM RECORDS = 1 RSTS RECORD".

NOTE

The first logical record on the RSTS/E output file is reserved for a statistical header of the form described in Section 6.5.3.2. This header is 6 bytes long. Thus, if the logical record length of the dataset to be transferred is less than 6 bytes, FLINT will use as many logical records as are necessary to contain the header. If, for example, the data set's logical records are only 4 bytes long, FLINT will have to allocate 2 of those records to hold the header. However, if you specify the /NH switch in response to the OUTPUT TO? dialogue question, the statistical header is not printed and the 6 bytes are freed for transferred data.

If the entire data set is not contained on the input device(s) specified, FLINT displays the question:

```
NEXT INPUT DEVICE?
```

Respond with the number of the drive on which the next portion of the data set resides. FLINT continues to perform transfers and to print NEXT INPUT DEVICE requests until it determines that the current diskette contains the end of the specified data set.

Note that on multi-volume input FLINT continually checks data set names against the one you specified, and, if possible, also checks volume numbers for correct sequence. If, for example, data set IDAT is being transferred and the third input volume specified contains no IDAT, FLINT displays the message:

```
FILE NOT FOUND - DXn:IDAT
```

and repeats the message NEXT INPUT DEVICE.

Moreover, if the volumes contain sequence numbers, and the number on a particular diskette is not 1 greater than that on the previous diskette, FLINT displays the message:

```
VOLUME #[m] CANNOT FOLLOW VOLUME #[n]
```

and repeats the message NEXT INPUT DEVICE.

If no fatal errors occur, FLINT, on completing the IBM–to–RSTS/E transfer, displays the message:

```
EXCHANGE COMPLETED
```

at your terminal, and lists both the number of IBM records read and the number of RSTS records written.

6.5.3.1 Specifying the Known Diskettes of a Data Set — Before you initiate the transfer of a multiple-volume data set, you may know on which diskettes the data set resides. In that case, respond to the INPUT FROM question by specifying the diskette on which the data set begins, then the data set's name, and then the other diskette(s) onto which the data set is continued. For example:

```
DX1:DATASETA,DX2:,DX3:,4
```

Note that this example, in accordance with good data procedure, causes the diskettes containing DATASETA to be mounted in order on consecutively numbered drives. You can, on the other hand, mount and specify diskettes out of sequence, but there is a chance for confusion. Note also that the unsequenced order of specifications must match exactly the unsequenced order of mounting. If it does not, FLINT will print the message VOLUME #[m] CANNOT FOLLOW VOLUME #[n], which is described in Section 6.5.3.

6.5.3.2 Format of the RSTS/E Disk File — FLINT writes, on the RSTS/E output file, a header record. This header contains the following information (all fields are 2–byte integer fields in standard CVT%$ format; see the *RSTS/E BASIC–PLUS Language Manual*):

Byte	Contents
1–2	Physical block number of the last logical block in the file.
3–4	Number of logical records in the last physical block.
5–6	Logical record length in bytes.
7–	Remainder of the logical record contains no valid data (the entire logical record is reserved for the header).

If the logical record length of the data set is less than 6 bytes, this header will occupy more than one RSTS/E logical record. Note that the /NH switch can be used to suppress this header (see Section 6.5.3).

6.5.4 Transferring RSTS/E Files to IBM Diskettes

Type /TOI in response to FLINT's number sign (#) prompt to cause FLINT to print the dialogue for transferring a RSTS/E file to IBM diskette(s). In the following description of that dialogue, FLINT's questions are followed by explanations.

```
OUTPUT TO?
```

This question asks you to specify the IBM data set to which FLINT will output the RSTS/E data. Respond with a data set name in the form:

DXn:dsn

DXn: specifies a diskette on which the data set is written; n is a device number from 0 to 7. dsn is the output data set name, composed of 1 to 8 characters, including blanks. If you omit the dsn, FLINT assumes the name RSTS.

Multiple volumes must be separated by commas; for example:

```
DX0:DATSETC,DX2:
```

```
TRANSLATE FROM ASCII TO EBCDIC <YES>?
```

This question asks you if you wish the RSTS input data to be translated from its current ASCII format into EBCDIC format. As the default <YES> indicates, pressing RETURN causes FLINT to translate the data. If you type NO, FLINT does no translation.

```
INPUT FROM?
```

This question asks you to specify the RSTS/E file from which data will be transferred to the data set specified in answer to the OUTPUT TO question. Specify an input file in RSTS/E format:

dev:[proj,prog]name.typ/PROT:n.

By default, dev: is the system disk, [proj,prog] is the current account, and the protection code is 60. Note that the device must be a RSTS/E disk and that wildcard characters (? or *) cannot be used to designate project-programmer numbers.

```
RECORD LENGTH <128>?
```

This question asks you to specify the number of characters to be included in each IBM output record. As the default indicates, pressing RETURN causes

the output records to contain 128 characters each. (IBM records contain 1 to 128 characters and are fixed length.) To specify a shorter record length, respond by typing any number smaller than 128 and greater than 0.

On receiving this response, FLINT computes the blocking factor to be used in the transfer of RSTS/E data, and displays a message expressing that factor as the ratio of IBM to RSTS/E records. This message has the form:

```
1 RSTS RECORD = N IBM RECORDS
```

For an explanation of how FLINT computes the blocking factor, see "INPUT FROM?" in the IBM–to–RSTS/E dialogue (Section 6.5.3).

After computing the blocking factor, FLINT proceeds with the transfer. If FLINT determines that the RSTS/E file cannot fit on the number of diskettes specified in answer to the OUTPUT TO question, it prints the question:

```
VOLUME #(n + 1)?
```

This question asks you to specify an additional diskette for output; respond with DXn: or with n, where n in either case is a device number from 0 to 7. Or, press RETURN, which is equivalent to DX0:. Until FLINT can complete the transfer, it will continue to ask the VOLUME # question and to accept one of these responses. All diskettes used in the transfer will be labeled with volume sequence numbers, and, where necessary, with continuation markers.

NOTE

No more than 99 diskettes can be used for one data set. If you attempt to use more, FLINT will print the message:

```
MAXIMUM NUMBER OF VOLUMES EXCEEDED
```

and will issue its # prompt.

If no fatal errors occur, FLINT, on completing the RSTS/E–to–IBM transfer, prints at your terminal a message of the form:

```
EXCHANGE COMPLETE
data set name RESIDES ON x DISKETTES
```

where x is the number of diskettes on which the named data set resides.

6.5.5 Initializing and Erasing a Diskette

Type /ZERO in response to the FLINT number sign prompt (#) to initiate the dialogue for initializing (zeroing) the directory of a single density diskette. In the following description of that dialogue, the FLINT questions are followed by explanations.

```
ZERO WHICH DISKETTE?
```

This question asks you to specify the diskette drive of the disk containing the directory you wish to zero. The drive specification takes the following form:

DXn: or n

where n is the unit number in the range of 0 to 7.

```
REALLY ZERO DXn: <NO>?
```

This question is asked to ensure that you have specified the correct diskette, where DXn: indicates the disk to be initialized. If you answer yes (Y) to this question, FLINT zeroes the directory of the diskette. If you accept the default (NO), FLINT prints a message and reprompts with a number sign (#).

Type /ERASE in response to the FLINT number sign prompt to initiate the dialogue for erasing a single density diskette. In the following description of that dialogue, the FLINT questions are followed by explanations.

```
ERASE WHICH DISKETTE?
```

This question asks you to specify the diskette drive of the disk you wish to erase. The drive specification takes the following form:

DXn: or n

where n is the unit number in the range of 0 to 7.

```
REALLY ERASE DXn: <NO>?
```

This question is asked to ensure that you have specified the correct diskette, where DXn: indicates the diskette to be erased. If you answer yes (Y) to this question, FLINT initializes the directory of the diskette and overwrites all data with zeros. Note that this operation can take some time. If you accept the default (NO), FLINT prints a message and reprompts with a number sign (#).

6.5.6 Dialogue Examples: /DIRECTORY, /TORSTS, /TOIBM, /ZERO, and /ERASE

In the first example, the directory of a diskette is requested and data from that diskette is transferred to a RSTS/E file.

```
FLINT   V7   RSTS V7   TIMESHARING
#/DIREC
OUTPUT TO? (RET)
DIRECTORY OF? DX0:

DIRECTORY OF DX0:

DSN     BRL   BOE    EOE    EOD  BI MVI VSN

XFILEXMA    128    01001  30011    30012
WUMPUSDB    080    30012  43019    43020
PHONE DB    128    43020  47003    47004

TOTAL OF 3 DATA SETS ON DX0:

#/TORS
OUTPUT TO? PHONE.DAT
TRANSLATE FROM EBCDIC TO ASCII <YES>? (RET)
INPUT FROM? DX0:PHONE DB

DX0:
 4  128 CHARACTER IBM RECORDS = 1 RSTS RECORD

EXCHANGE COMPLETED
 88 IBM RECORDS READ
 89 LOGICAL ( 23 PHYSICAL ) RSTS RECORDS WRITTEN
```

In the /DIRECTORY dialogue, the directory of DX0: is printed at the terminal.

In the /TORSTS dialogue, data set PHONE DB is transferred from diskette DX0: to the RSTS/E output file PHONE.DAT. By default, the data is translated into ASCII on being transferred. FLINT computes and prints the blocking factor (4 IBM records to 1 RSTS/E record).

In the next example, a RSTS/E file is transferred to an IBM diskette.

```
FLINT   V7   RSTS V7   TIMESHARING
#/TOIBM
OUTPUT TO? DX1:MYDATA
TRANSLATE FROM ASCII TO EBCDIC <YES>? (RET)
INPUT FROM? MYDATA.DAT
RECORD LENGTH <128>? (RET)

1 RSTS RECORD = 4 IBM RECORDS

EXCHANGE COMPLETED
MYDATA RESIDES ON 1 DISKETTE
```

In this /TOIBM dialogue, the RSTS/E file MYDATA.DAT is transferred to the IBM data set MYDATA on diskette DX1:. By default, the data is translated to EBCDIC on being transferred. FLINT computes and prints the blocking factor (1 RSTS/E record to 4 IBM records).

In the next example, a single density diskette is initialized and erased by means of the /ZERO and /ERASE operations.

```
RUN $FLINT
FLINT V7 RSTS V7 Timesharing
#/ZER
ZERO which diskette? 0
Really ZERO DX0: <No> ? Y
#/ERA
ERASE which diskette? 1
Really ERASE DX1: <No> ? Y
#/ERA
ERASE which diskette? 1
Really ERASE DX1: <No> ? N
ERASE aborted
```

6.5.7 FLINT Error Messages

FLINT, on encountering an error, displays an appropriate error message. If the error is non-fatal, FLINT repeats the prompting question; if the error is fatal, FLINT aborts the transfer. Table 6–7 lists and describes the error messages specific to FLINT. Error messages specific to RSTS/E are described in Appendix A.

Table 6–7: FLINT Error Messages

Message and Meaning
?DATA SET NAME TOO LONG The specified IBM data set name has more than eight characters, including spaces and tabs.
?DATA SET NOT FOUND FLINT cannot find the specified data set on the current diskette.
?ILLEGAL DATA SET NAME A data set name has been specified where none is permissible.
?ILLEGAL FILE NAME The specified file name contains unacceptable characters or violates the specification format.
?ILLEGAL PPN The specified account number contains wildcards, which are not allowed.
?LOGICAL DEVICE NAME NOT ASSIGNED The specified logical device name has not been previously assigned.

(continued on next page)

Table 6–7: FLINT Error Messages (Cont.)

Message and Meaning
?OUTPUT DEVICE MUST BE DISK In a /TORSTS transfer, an output device other than the only permissible type—a RSTS/E disk—has been specified.
?VOLUME #[m] CANNOT FOLLOW VOLUME #[n] In a multi-volume /TORSTS transfer, the specified input volume number [m] is not 1 greater than the previously specified volume number [n].
The following errors are fatal and cause the transfer to be aborted:
?BOE = EOD – TRANSFER ABORTED In the specified data set, the Beginning Of Extent address is the same as the End Of Data address; i.e., the data set has no length.
?DATA SET LABEL MUST BE HDR1 – TRANSFER ABORTED The label ID of a specified data set for transfer is not HDR1, which it must be according to IBM format.
?TRACK 00 DOES NOT CONTAIN ERMAP FIELD – TRANSFER ABORTED There is no ERMAP field in track 00 of the specified diskette; according to IBM format, this field must be present.

6.6 Sending a Message to the System Manager: The GRIPE Program

The GRIPE system program allows you to communicate comments to the system manager. Comments, which you type while running GRIPE, are written to a common file where they are retained for inspection by the system manager.

Run GRIPE by typing the following command:

```
RUN $GRIPE
```

GRIPE prints an identification header followed by a query line to indicate its readiness to accept comments:

```
GRIPE V7 RSTS/E V7 TIMESHARING
YES? (Press the ESCAPE key to end)
```

Following the prompt, type the text of your comment which is entered into the common file. Terminate the text of your comment by typing CTRL/Z or the ESCAPE or ALTMODE key. (Typing the ESCAPE or ALTMODE key is echoed at the terminal by a dollar sign ($) being printed.) No carriage return-line feed operation is performed. The program indicates its acceptance of the text and its termination by printing:

```
THANK YOU
```

6.7 Obtaining Help: The HELP Program

The system library program HELP consists of files that contain descriptions of other system programs and system resource commands. HELP is an interactive program that prompts you for topics and subtopics. To obtain information on system programs or commands, you type the desired topic and optional subtopic(s) in response to the HELP prompts. Your response causes HELP to display the contents of the appropriate file.

To invoke the HELP program, type:

```
RUN $HELP
```

When successfully invoked, HELP prints an identifying header line and a prompt for the desired topic. For example:

```
RUN $HELP

HELP V7 RSTS V7 Timesharing

Topic?
```

In response to the Topic? prompt, type one of the following:

1. The name of the system program or resource command you wish information on. If you abbreviate the name, information on all topics that match that abbreviation are displayed. In addition to the program or command, you can specify one or more of the subtopics (separated by spaces) associated with that program or command.
2. The RETURN key, which causes HELP to print descriptive text on its use and a list of possible topics.
3. An asterisk wildcard (*), which causes HELP to print information on all of the available topics.

You can also invoke the HELP program with a CCL command, as follows:

```
HELP

Help can be obtained on a particular topic by typing:
        HELP topic subtopic subsubtopic...
A topic can have the following format:
   1) an alphanumeric string (e.g., a command name, option, etc.)
   2) same preceded by a "/" (=> interpreted as a switch)
   3) the match-all symbol "*"
Examples:
    HELP DIRECTORY /S
    HELP SET STALL
Abbreviations result in all matches being displayed.
```

```
Additional information is available on:

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

Topic?
```

When HELP is invoked with the CCL command, it displays descriptive text on its use and a list of possible topics as well as the Topic? prompt (i.e., as if you had typed a RETURN key in response to the Topic? prompt).

By default, HELP displays the requested information on your terminal. To override the default, you can specify the /OUTPUT: switch that causes HELP to print the requested information to a user file. The switch has one of the following formats:

```
HELP/OUTPUT:filespec topic [subtopic [...]]
```

or

```
HELP/OUTPUT:filespec * [...]
```

where filespec can be a complete RSTS/E file specification as described in Chapter 11. A file name is required; HELP defaults the file type to .LST. Note that the /OUTPUT: switch can only be used with the HELP CCL command. Consider the following example:

```
HELP/OUTPUT:INFO SYSTAT WHO
```

This command causes the HELP program to create a file in your account on the public structure named INFO.LST. This file contains a copy of the HELP file on the SYSTAT subtopic WHO.

If you attempt to specify a HELP topic for which no file exists, HELP prints:

```
SORRY, NO INFORMATION AVAILABLE ON topic
```

Then it prints information on valid topics (as if you had typed HELP and a RETURN key or a RETURN key in response to the Topic? prompt).

Many of the topics described by HELP are further explained in subtopics. For example:

```
Topic? SYS

SYSTAT

  The SYSTAT program provides current system information. It may be
  run with the RUN command or the CCL command, "SYS[TAT]". The
  command string is of the form:
        [output file] [/options]
  Only logged-in users may specify an output file. If none is
  specified, the output will be to the keyboard. If no options are
  specified, the status of jobs, devices, disks, buffers, run-time
  systems, resident libraries and message receivers will be reported.
     Examples:
         SYSTAT
         SYS/A

  Additional information is available on:

  /A                /B                /C                /D
  /F                /J                /Kn               /L
  /M                /N                /o[:dev]          /P
  /proj,prog        /proj,*           /R                /S
  /U                /W[:dev]          /Q,Q              /-
  JOB               WHO               WHERE             WHAT
  SIZE              STATE             SWAPPING          RUN-TIME
  PRI/RB            RTS

  SYSTAT Subtopic?
```

In this example, the initial description of the SYSTAT program (abbreviated to SYS) makes note of additional information available on various options. To cause HELP to print the additional information, type the desired subtopic(s) in response to the HELP Subtopic? prompt. For example:

```
 SYSTAT Subtopic? WHO

SYSTAT

  WHO

  The "Who" column in the job status report gives the account under
  which the job is running. This column will contain one of the
  following:

          nn,mm      the job is running under account [nn,mm].
          [OPR]      the job is running under a system operator account.
          [SELF]     the job is running under your account.
          **,**      the job is not logged in.

SYSTAT Subtopic? ^Z

Topic? ^Z
```

The HELP program continues to prompt for subtopics until you type CTRL/Z, which causes HELP to return to the Topic? prompt; another CTRL/Z terminates the program.

A topic and optional subtopics can be specified with the HELP CCL command. In this case, HELP displays the requested information and exits. For example:

```
HELP SYS WHO

SYSTAT

  WHO

The "Who" column in the job status report gives the account under
which the job is running. This column will contain one of the
following:

        nn,mm       the job is running under account [nn,mm].
        [OPR]       the job is running under a system operator account.
        [SELF]      the job is running under your account.
        **,**       the job is not logged in.
```

However, if you wish to use the CCL command to display information but not exit upon completion, use the /PROMPT switch as follows:

```
HELP/PR[OMPT] topic
```

This switch (as with /OUTPUT) can only be used with the CCL command. Also, when /PROMPT is used with no topic, it causes HELP to inhibit the display of HELP use and topic text and print only the Topic? prompt. For example:

```
HELP/PR EXIT

EXIT

  Exit to the job keyboard monitor

Topic?
```

6.8 Entering the System: The LOGIN Program

The LOGIN system program runs from either a logged in or a logged out terminal. It activates a job at a terminal, attaches a detached job* to a terminal, or runs designated system programs from a logged out terminal.

6.8.1 Running LOGIN from a Logged Out Terminal

If you type HELLO (or any character) followed by RETURN, LOGIN prints the system identification line as in the following sample printout:

```
RSTS V7 Timesharing Job 12 KB16 21-Sep-81 08:37 AM
User: 120,80
```

The line contains the system name and version number, the local installation name, the job number activated, the keyboard number of the terminal, and the current system date and time. The User: prompt requests you to type your account number.

Type the project and programmer numbers separated by either a comma or a slash and terminated with the RETURN key. (Typing the slash as a separator inhibits printing of any system notice messages.)

If you like, you can also type HELLO, LOGIN, LOG or I followed by a space and your project-programmer number, as shown in the following sample dialogue:

```
HELLO 120,80
Password:
```

In this case, LOGIN does not print the User: prompt but immediately prompts you to enter the password. LOGIN disables echo printing at the terminal when the password is typed.

If either the account does not exist or the password does not match, LOGIN prints the INVALID ENTRY – TRY AGAIN message and the User: prompt. You can try the sequence to a maximum of five times. LOGIN allows 30 seconds in which to type an entry. After the fifth invalid entry, LOGIN prints the ACCESS DENIED message and frees the job for other use.

---

* A job becomes detached because the connection of a dial-up line is broken or a privileged job executed the SYS system function to detach the job from the terminal.

A valid entry causes LOGIN to check for any other jobs which may be running on the system under the same account number. If other jobs are running and none are detached, LOGIN reports how many such jobs by printing a message similar to the following sample and prints the system notice messages:

```
1 other user is logged in under this account
```

If any jobs are running detached under the current account, LOGIN instead reports the number of each such job and requests you to type the number of the job to be attached to the terminal. The following sample shows the procedure:

```
Job 16 is detached under this account
Job number to attach to? 16
Attaching to Job 16
```

When the job is attached, the current job number is freed for other users and the newly attached job runs at the terminal.

To continue running the current job, type the RETURN key in response to the query. LOGIN subsequently displays the message concerning other jobs running under the same account and displays the system notice messages, if any:

```
Job 16 is detached under this account
Job number to attach to? RET
2 other users are logged in under this account
```

System notices convey information which the system manager places in the file NOTICE.TXT in the system library. If the file does not exist, LOGIN proceeds. LOGIN exits to the default keyboard monitior, which clears the LOGIN program from memory, and prints a prompt, such as $ or Ready.

The complete sequence to log a job into the system when other jobs are running detached is shown below:

```
HELLO 1/210
Password:
Job 16 is detached under this account
Job number to attach to? RET
2 other users are logged in under this account

$
```

The complete sequence to attach another job to the terminal when logging into the system is shown in the following sample printout:

```
HELLO

RSTS V7 Timesharing Job 11 KB2 02-Nov-81 11:31 AM
User: 1/210
Password:
Job 16 is detached under this account
Job number to attach to? 16
Attaching to Job 16

$
```

To attach a job to a terminal when the job number is known, type the ATTACH or ATT command as follows:

```
ATTACH

RSTS V7 Timesharing Job 13 KB2 02-Nov-81 12:01 PM
Job number to attach to? 34
Attaching to Job 34

$
```

LOGIN runs and prints the system identification line and, on the next line, the JOB NUMBER TO ATTACH TO query. You must type the number of the detached job. If the job is not detached or does not exist, LOGIN prints an appropriate message followed by ACCESS DENIED. You must type another command to log into the system.

If the job is detached, LOGIN prompts you for the password of the account under which the detached job is running. After you enter the password, LOGIN prints the ATTACHING TO JOB x message and attempts to attach the specified job to the terminal. An incorrect password causes LOGIN to print the FAILURE TO ATTACH TO JOB x message and to terminate as shown in the following sample printout:

```
ATTACH

RSTS V7 Timesharing Job 13 KB2 02-Nov-81 12:03 PM
Job number to attach to? 21
Password:
Attaching to Job 21
Failure to attach to Job 21
```

You must try again. If the system successfully attaches the job to the terminal, the terminal becomes the console terminal of the job. Further terminal output is under programmed rather than system control.

To omit the printing of the identification and the query lines, type the job number on the same line as the ATTACH or ATT command as follows:

```
ATT 27
Password:
Attaching to Job 27

$
```

6.8.2 Running LOGIN at a Logged In Terminal

If you type the HELLO or the ATTACH command at a terminal already logged into the system, the LOGIN system program is invoked. Note that to use the ATTACH command at a logged-in terminal, ATTACH must be installed as a CCL command on the system. LOGIN prints the system identification line with one additional item inserted. Between the job number and the keyboard number printed on the line, LOGIN inserts the project-programmer numbers of the account under which the current job is running.

LOGIN determines if any other jobs are running under the same account and prints the message informing you of the number of those jobs. The following sample dialogue shows the procedure:

```
Ready

HELLO

RSTS V7 Timesharing Job 15 [1,210] KB3 02-Nov-81 02:53 PM
1 other user is logged in under this account

Ready
```

If any such jobs are running detached, LOGIN also prints the message informing you of the number of each such job and, on the following line, prints the ATTACH TO query as follows:

```
Ready

HELLO

RSTS V7 Timesharing Job 27 [1,210] KB2 02-Nov-81 01:02 PM
Job 15 is detached under this account
Job number to attach to? 40
No Job by that number - try again
Job number to attach to?
```

To attach one of the jobs to the terminal, type one of the job numbers reported in the message. LOGIN determines whether the job is nonexistent or whether it is already attached to another terminal. In either case, the program displays an appropriate error message saying try again and subsequently redisplays the ATTACH TO query.

To continue running the current job, type the RETURN key in response to the ATTACH TO query. As a result, LOGIN prints the information message telling how many other jobs are running under the same account and prints the command environment prompt.

```
Job number to attach to? RET
2 other users are logged in under this account

$
```

When you respond to the ATTACH TO query by typing one of the job numbers reported in the message, the program proceeds as shown in the following sample printout:

```
HELLO

RSTS V7 Timesharing Job 36 [1,210] KB3 02-Nov-81 02:55 PM
Job 15 is detached under this account
Job number to attach to? 15
Attaching to Job 15

$
```

To attach to a job known to be running detached under the same account, type the job number on the same line as the ATTACH command (ATTACH must be installed as a CCL command). The LOGIN program determines if the specified job exists and is detached. If not, it displays an appropriate error message and the ATTACH TO query. You can type another job number or the RETURN key. If the job exists and is detached, LOGIN compares the account numbers under which both the current job and the detached job are running. If the accounts are different, the program prompts you for the password of the account under which the detached job is running. The following sample printout shows the procedure:

```
ATTACH 51
No Job by that number - try again
Job number to attach to? 15
Password:
Attaching to Job 15
Failure to attach to Job 15
```

6.9 Leaving the System: The LOGOUT Program

LOGOUT is called when you have completed all processing and are ready to leave the terminal. The LOGOUT program is started when the BYE command is typed at a user terminal logged into the RSTS/E system. LOGOUT checks your current disk quota to ensure that you do not log out of the system with more than the acceptable amount of disk storage being used for your files. If your disk files are within the acceptable disk quota size, LOGOUT logically disconnects the terminal from the system, removes the current job number from the list of active jobs and displays some information regarding your system usage.

In response to the BYE command, LOGOUT prints:

```
CONFIRM:
```

You can type any of the responses shown in Table 6–8.

When you type I, indicating individual deletion mode, LOGOUT prints the name, size, protection code, and creation date of each file stored under your current account number on the system disk. This information is followed by a ? after which the system awaits a response from you which can be:

File Deletion Mode Response	Meaning
RETURN key	Save the file just listed.
K	Delete (kill) the file just listed.

Table 6–8: LOGOUT CONFIRM: Responses

CONFIRM: Response	Meaning
Y	The system performs the checks described above. If successful, the LOGOUT messages are printed. If not successful, an error message is printed and you must delete some files.
N CTRL/C	These responses indicate that you do not want to log out of the system. The LOGOUT procedure is terminated without logging you off the system and the system prints the READY message.
?	Causes LOGOUT to print an explanation of the acceptable responses to CONFIRM:.
RETURN key	Causes LOGOUT to print a message instructing you to type ? to obtain a description of logout procedures.
I	Causes LOGOUT to enter individual file deletion mode.
F	Causes a fast logout procedure if your disk storage space is within acceptable limits.
Other	Same as RETURN key.

An example of a LOGOUT sequence is shown below:

```
BYE
Confirm: Y
Disk quota of 400 exceeded by 52 blocks
Some file(s) must be deleted before logging out
Type '?' for help
Confirm: I
DATA          .001          200       60        03-Nov-81 ?
DATA          .002          150       60        03-Nov-81 ?
DATA          .003          100       60        03-Nov-81 ? K
Confirm: Y
Saved all disk files; 352 blocks in use, 48 free
Job 24 User 100,101 logged off KB3 at 03-Nov-81 03:09 PM
System RSTS V7 Timesharing
Run time was 1.4 seconds
Elapsed time was 3 minutes, 59 seconds
Good afternoon
```

In DCL environment, there is no CONFIRM prompt. In other environments, you can omit the CONFIRM: message by typing the BYE command and the response to the CONFIRM: message. For example, to perform a fast logout, type:

```
BYE F
```

The LOGOUT program runs and performs the fast logout procedure by printing a series of LINE FEED characters instead of printing the final accounting information. If your job exceeds the acceptable limit for disk storage, LOGOUT displays the QUOTA EXCEEDED message and the CONFIRM: message to allow you to delete some files before logging out.

6.10 Obtaining Account Data: The MONEY Program

MONEY is the RSTS/E system accounting program which allows you to obtain data on your account status. Type:

```
RUN $MONEY
```

In the following example, you are logged into the system under account [100,100], and run the MONEY program:

```
RUN $MONEY
MONEY   V7 RSTS V7 Timesharing
System Accounting Program

Acct     Password  CPU-Time KCT'S Connect Device  Disk   Quota UFD
100,100              26.8  2436    23       2     316    5000  16
```

The headings and information contained in the MONEY report are described in Table 6–9.

Table 6–9: The MONEY Report

Heading	Information
ACCT	Your account number.
PASSWORD	Not printed for non-privileged users.
CPU–TIME	The total number of seconds of central processor time used by the account.
KCT's	The total number of "kilo-core ticks" used by the account. This is a measure of total central processor usage, along with central processor time and memory usage. Whenever a program uses one-tenth of a second of CPU time, this value is incremented by the size of the program in K words.
CONNECT	The total number of minutes the account is logged into the system.
DEVICE	The total number of minutes devices are assigned by this account.
DISK	The number of blocks used on the public structure. If the account is using more than 65535 blocks, the entry is shown as > = 65535.
QUOTA	The disk quota in blocks.
UFD	The cluster size of your file directory.

These values do not reflect the current time-sharing session, because the accounting information is updated when the job is logged off.

6.11 Device Transfer: The PIP Program

The PIP (Peripheral Interchange Program) system program is a file transfer and maintenance utility that can copy files from one RSTS/E device to another, concatenate, delete, and rename files, change protection codes and file attribute data, zero accounts, and list directories.

To invoke PIP, you must be logged into the system and type the following command:

```
RUN $PIP
*
```

In response, PIP prints an asterisk (*) prompt to indicate its readiness to accept input.

In addition to the RUN command, the RSTS/E CCL command PIP invokes the program. For example:

```
PIP
*
```

Note that if you use the PIP CCL command, you can include PIP commands on the same line.

To terminate PIP, type CTRL/Z in response to the PIP asterisk prompt:

```
PIP
*^Z
```

A CTRL/Z causes PIP to complete the current operation and exit in an orderly fashion. If you type CTRL/C in response to an asterisk prompt, PIP exits but does not complete the current operation. If you type CTRL/C during a PIP operation, PIP halts the operation (some data may be lost) and reissues its prompt.

6.11.1 PIP Command Line Specifications

The PIP command line specifies the actions you wish performed and the files involved. PIP accepts a command line of up to 80 characters in the following general format:

```
*[output[/sw]=]input[/sw][,input[/sw],...]
```

where output and input are file specifications and /sw are one or more switches that specify the action to be taken on the files. You terminate a PIP command line with the RETURN key.

An output file specification has the form:

```
dev:[proj,prog]name.type/PROT:n
```

where:

dev — is a device specification. If none is specified, the default is DK:. Note that DK: defaults to the public disk structure (SY:) unless you have previously assigned DK: to another device.

[proj,prog] — is a project-programmer number. If none is specified, the default is your current account.

name — is a file name specification. PIP allows you to specify only one output file specification. If none is specified, the default is an asterisk (*) wildcard character (see Section 16.11.2). Note that for output-only (non-file-structured) devices such as KB:, LP:, and PP:, the system ignores the file name and type specifications.

.type — is a file type. If none is specified, the default is an asterisk (*) wildcard (see Section 6.11.2). If you specify only the dot, the type is null.

/PROT:n — The number n specifies a protection code (see Section 4.1.4.5). If you do not specify a protection code, the following rules apply.

For the executable (64) bit, PIP uses the executable bit of the input file if the input file has a protection code. (Only disk files have a protection code.) If the input file comes from some other source, for example, magnetic tape, then PIP adds the executable bit to the output file protection code if the file type for the output file indicates that it is executable (.BAC, .TSK, .SAV)*. Otherwise, PIP clears the executable bit.

For bits other than the executable bit, PIP uses the user-assigned protection code (which you might establish with the command ASSIGN /PROT:40) if there is one. If there is no user-assigned protection code, PIP uses the protection code of the input file, if it has one. If it does not (a magnetic tape file, for example), PIP uses the default protection code (normally 60).

Note that PIP does not allow you to set the privileged bit (128) for an output file unless you are a privileged user. No error is returned if you try; PIP does not set the bit.

---

* PIP recognizes a file type as executable by comparing it against the file types currently in the run-time system list at the time. You can see the file types in the run-time system portion of a SYSTAT report.

If you do not type any output specification, PIP assumes your terminal (KB:).

An input file specification has the form:

```
dev:[proj,prog]name.type/PROT:n
```

where:

dev:	is a device specification. If none is specified, the default is DK: as described for output. Note that if a preceding input file contains a device specification, that device becomes the default for the current command line until overridden by another specification.
[proj,prog]	is a project-programmer number as described for output.
name	is a file name specification. PIP allows you to specify up to 6 input file specifications separated by commas. If none is specified, the default is an asterisk (*) wildcard (see Section 6.11.2). Note that an input file specification is required in a delete operation (see Section 6.11.6.18). For input-only (non-file-structured) devices such as KB:, PR:, and CR:, the system ignores a file name and type specification.
.type	is a file type as described for output.
/PROT:n	is a protection code. PIP ignores input file protection codes unless the file is the target of a rename operation (see Section 6.11.6.19).

If you do not type any input file specification, PIP returns the error:

```
?NO INPUT FILE
```

A PIP switch specification (see Section 6.11.4) always begins with a slash (/). When you specify more than one switch in the command line, you must separate them with slashes. When no switch is specified, PIP performs a block mode transfer as described in Section 6.11.6.4.

Note that both the input and output file specifications can contain the RSTS/E file switches /PROTECT, /FILESIZE, /CLUSTERSIZE, /MODE, /RONLY, and /POSITION. However, file switches must immediately follow the file specification; an error is returned if file and PIP switches are interspersed. Also, /CLUSTERSIZE is ignored on non-disk output file specifications. File specification switches and their format are described in Section 4.1.4.

If you type neither an input nor an output file specification (that is, if you press RETURN in response to an asterisk prompt), PIP displays its current version and revision level numbers.

6.11.2 Wildcard Specifications

PIP allows you to use the asterisk (*) and question mark (?) wildcard characters in input and output file specifications. The use of wildcards can save you typing time and permits you to designate multiple files in a single file specification and thereby circumvent the PIP restriction of six maximum input file specifications.

Wildcards can be specified in the following manner:

Specification	Result
FILE.*	All files with the name FILE and any type.
*.TYP	All files with the type .TYP.
.	All files.
FILE.TY?	All files with the name FILE and .TY as the first two characters of the file type. The third character is any character including blank.
FILE??.TYP	All files with 4– to 6–character names, the first four of which are FILE, and a file type of TYP.
FILE??.T??	All files with 4– to 6–character names, the first four of which are FILE, and any type whose first character is T.
AB?.TYP	All files with 2– to 3–character names, the first two of which are AB, and a file type of TYP.
FILE??.*	All files with 4– to 6–character names, the first four of which are FILE, and any type.
*.TY?	All files of any name whose type begins with .TY.

In the absence of an element or elements of the full input or output file specification (see Section 6.11.1), PIP applies the following defaults to the missing elements (note that delete operations do not permit default input specifications):

Specification	Default	Meaning
null	*.*	All files in the current account.
*	*.*	All files in the current account.
*.	*.	All files with null types in the current account.
.	*.	All files with null types in the current account.
.TYP	*.TYP	All files with a file type of .TYP in the current account.
FILE	FILE.*	All files named FILE with any type in the current account.

You can use the question mark wildcard character in output file specifications, but if such use causes PIP to try to create a file name with embedded spaces, it will print the error ?ILLEGAL FILENAME, ignore the request, and proceed to the next input file. See Section 6.11.6 (File Transfer Operations) for an example.

PIP allows you to specify an asterisk wildcard character in a project-programmer number specification. However, if wildcard accounts are specified for input files, the file must be resident on disk or magnetic tape.

NOTE

You *cannot* limit a wildcard designation by including a protection code. The protection code is simply ignored. The file specification *.*/PROT:40 will refer to all files in your account, *not* just those with a protection code of 40.

6.11.3 Indirect Command Files in PIP

PIP allows you to specify an indirect command file in response to its prompt. An indirect command file is a user-created (with PIP or an editor) ASCII file that contains all of the information PIP needs to perform an operation. If a line of the file begins with a semicolon (;), PIP treats the line as a comment and skips to the next line in the file. If the file causes the execution of a /ZE or /IN switch operation, confirmation questions are printed at your terminal.

You specify an indirect command file to PIP in the following manner:

```
*@dev:[proj,prog]file.type
```

where the at sign (@) is the first character following the prompt. The device defaults to the public structure if none is specified. The project-programmer number defaults to the current account. The file name must be specified. However, if the type is not included, it defaults to .CMD.

An indirect command file, in addition to containing PIP commands and file specifications, can contain references to other indirect command files. This process is called nesting. The number of files you are allowed to nest is dependent on the amount of memory available to you. If memory is exceeded, PIP prints the error ?NO BUFFER SPACE FOR INDIRECT COMMAND FILE, halts the operation, and returns to the prompt level. Note that a file, which contains nested indirect commands, must be resident on a file structured disk.

Most errors during command file processing cause PIP to return to prompt level. On the following errors, however, PIP prints the error message and continues processing:

1. ?NO FILES MATCHING <spec>
   where <spec> is a file specification and the operation was /DE or /LI.

2. ?FILE OR ACCOUNT ALREADY EXISTS
   where the operation was /RE.

The following command shows the procedure you would use to execute an indirect command file:

```
*@ABC
```

PIP reads the file named ABC.CMD in the current account on the public structure. If you had specified KBn: following the at sign (@KB3:, for example), PIP would read the specified keyboard for commands. You terminate the keyboard command set with a CTRL/Z.

Because a logical account assignment (see Section 4.3.6.5) and the PIP indirect command are both designated with an at sign, the placement of the at sign in the PIP command line is important. When the command line contains an at sign, PIP scans the line for validity as an indirect file command. If PIP encounters more than one file specification, wildcard file specifications, or a PIP switch, PIP assumes that the at sign is a logical account and processes the command line accordingly.

6.11.4 PIP Switches — an Overview

You specify the operations that PIP is to perform with one or more switch specifications in the command line (see Section 6.11.1). A switch is always preceded by a slash (/) and, where more than one is specified, separated by slashes. You terminate the command line with the RETURN key.

Table 6–10 lists the PIP switches, their format, and the operations that they perform. The upper-case characters in each switch name indicate the minimally permissible abbreviation for that switch.

Table 6–10: PIP Switches

Switch	Format	Meaning
Informational switches		
HElp	/HE	Causes PIP to print a text message at your terminal that describes file specifications, switches, and options.
File transfer switches		
none		See the description of /BL.
ACcess	/AC	For disk input only, causes PIP to change the file's last access date to the current date. If you do not specify this switch, PIP preserves the last access date.

(continued on next page)

Table 6–10: PIP Switches (Cont.)

Switch	Format	Meaning
APpend EXtend	/AP /EX	For disk output only, causes PIP to append the input file to the output file (extend the output file). If no output file exists, PIP displays a warning message and creates the file.
AScii	/AS	Formatted ASCII transfer; causes PIP to discard nulls, parity bits, and rubout characters.
BLock	/BL	Causes PIP to perform a block-by-block transfer with no data translation.
Block SIZE	/BSIZE:n	For magnetic tape output only. PIP uses the physical blocksize specified by n (in bytes) for data written to the tape.
CLustersize	/CL:n	For disk only, sets the output file cluster size to n. To have effect, this switch must be adjacent to the output file specification. If this switch is omitted from a disk-to-disk transfer, the current cluster size is preserved.
IGnore GO	/IG /GO	Causes PIP to ignore ?DATA ERROR ON DEVICE errors.
MOde	/MO:n	Sets the mode in which the file is to be opened. To have effect, this switch must be adjacent to the file specification. For an explanation of modes, see the *RSTS/E Programming Manual.*
NEw file	/NE	For disk output only, creates a new file with the current date of creation and access.
RETain	/RET	For disk output only, retains creation and access date of input file.
NOAttributes	/NOA	Causes PIP to transfer the file without writing attributes to the output file.
NOSupersede	/NOS /NOS:NOWARN /NOS:Q /NOS:IN	Causes PIP to display an error message and ignore the operation if an output file already exists. The :NOWARN option suppresses the error message. The :Q and :IN options are synonymous. They cause PIP to display the question "OK to replace file filespec?" if the output file already exists. Type Y, YE, or YES to replace the file.
PRotect	/PR /PR:NOWARN /PR:Q /PR:IN	Same as NOS. NOS is recommended to avoid conflict with protection code option.

(continued on next page)

Table 6–10: PIP Switches (Cont.)

Switch	Format	Meaning
Run-Time System	/RTS:name	Sets the output file's run-time system name to that specified in name. If no output file is specified, PIP renames the input file's run-time system. PIP's default action is described in Section 6.11.6.12.
UPdate	/UP	Causes PIP to open a pre-existing disk file and overwrite the existing data. If no file exists, PIP creates a new one.
Files with attributes, translation switches and options		
RMS	/RMS	For input disk files, translates RMS format to formatted ASCII or formatted binary. For output disk files, translates formatted ASCII or formatted binary to RMS variable length records. For non-disk transfers, translation depends on current format (see Section 6.11.6.15). If the /RMS switch is not specified for disk files, no translation is performed. The switch cannot be applied to both input and output files in the same command line.
	/RMS:FA	For input or output files, specifies formatted ASCII in the translation.
	/RMS:FB	For input or output files, specifies formatted binary in the translation.
	/RMS:FTN	For input files, translates FORTRAN carriage control to formatted ASCII. For non-disk output files, translation is automatic.
	/RMS:IM	For input files only, performs the same as /RMS but no data translation (i.e., suppresses transfer of attributes and removes RMS variable length header information).
	/RMS:PRN	For input disk files, translates RMS print files to formatted ASCII.
Date switches		
AFter	/AF:dd-mmm-yy	Causes PIP to include only files that were created after, but not on, the given date.

(continued on next page)

Table 6–10: PIP Switches (Cont.)

Switch	Format	Meaning
BEfore	/BE:dd-mmm-yy	Causes PIP to include only files that were created before, but not on, the given date.
ON	/ON:dd-mmm-yy	Causes PIP to include only files that were created on the given date.
SINce	/SIN:dd-mmm-yy	Causes PIP to include only files that were created on or after the given date.
TOday	/TO	Causes PIP to include only files that were created on the current date.
UNtil	/UN:dd-mmm-yy	Causes PIP to include only files that were created on or before the given date.
Date of Last Access	/DLA	For disk only, causes PIP to use the date of last access in file operations.
CREation	/CRE	Causes PIP to use the date of creation in file operations.
File operation switches - general		
HAlt	/HA	Causes PIP to halt a magnetic tape wildcard search as soon as it detects a file that does not match.
INspect QUery	/IN /QU	Causes PIP to display (one-at-a-time) the file specifications that match a wildcard specification. If used in transfer or /DE operations, as each file specification is displayed, type Y (for yes) to transfer or delete; type any other character to omit. Type CTRL/Z to end the display and operation.
	/IN:S	Prints file name, type, filesize, protection code, creation date, and last access date.
LOg WAtch	/LO /WA	Causes PIP to display a report on all actions taken during execution. The report is printed at your terminal.
NO LOg	/NOLO	Causes PIP to suppress the display on actions taken during execution.
NOrewind	/NO /RW:NO	Causes PIP to suppress the default rewind on magnetic tape prior to an input file search.
VErsion IDentify	/VE /ID	Causes PIP to display the program's current version number.

(continued on next page)

Table 6–10: PIP Switches (Cont.)

Switch	Format	Meaning
File operation switches - deletion		
DElete	/DE	For disk and DECtape only, causes PIP to delete the specified file.
	/DE:NO	Causes PIP to suppress the display of an error message if the file to be deleted does not exist.
ERase WIPE Out	/ER /WIPE or /WO	For disk, causes PIP to overwrite the file with zeroes prior to deletion. These switches are synonymous and must be used in conjunction with the /DE switch.
File operation switch - rename		
REname	/RE:option	For disk and DECtape only, causes PIP to rename the input file to that of the output file.
File operation switches - listing		
BRief Fast	/BR /F	Causes PIP to display a brief directory listing.
DIrectory LIst Slow	/DI /LI /S	Causes PIP to display a full directory listing. The listing switches accept option specifications that modify the directory listing (see Section 6.11.6).
File operation switches - zeroing		
ZEro	/ZE	Causes PIP to zero (initialize) the directory of an account or device or initialize the labels on a magnetic tape.
DENsity	/ZE/DEN:n	For magnetic tape only, sets the tape density prior to the zero operation.
PARity	/ZE/PAR:ODD /ZE/PAR:EVEN	For magnetic tape only, sets the parity of the tape prior to the zero operation.
Privileged-only switches		
LOCK	/LOCK	Causes the system to lock PIP in memory for the duration of the current operation. Note that /LOCK is not abbreviated.
PRIORity	/PRIOR	Causes the system to run PIP at special priority for the duration of the current operation.

6.11.5 PIP Information Switch (/HE)

The /HE switch (/HELP) causes PIP to display a file that contains information on PIP to your terminal. This file describes the input and output file specifications required by PIP as well as the legal switches and options.

6.11.6 File Transfer Operations

File transfer operations have the following command format:

```
*output/sw=input/sw
```

where output is a single file specification, and input can be one to six file specifications as described in Section 6.11.1. If you specify only one input file, that file is copied to the output file (the input file is unchanged). If you specify more than one input file, copies of the input files are concatenated and written to the output file (again, the input files are unchanged).

You can also specify one or more switches (described in the following sections) to modify the file transfer. If you do not specify a switch on disk-to-disk transfer, a block mode transfer (/BL, see Section 6.11.6.4) is performed by default. A block mode transfer causes PIP to copy the input file data, block-by-block, and any attributes on the input file into the output file. The transfer continues until the last block of the input file is copied.

If you do not specify a switch on disk to non-disk file transfers, a file transfer with format translation (see Section 6.11.6.15) is performed by default. That is, the transfer of RMS files with attributes to non-disk structured devices can cause PIP to translate the file to ASCII stream or formatted binary, depending on the attributes of the file. However, if the file contains fixed length records of 512 bytes (for example, a task image or library file), PIP does not translate the file but, rather, performs a block mode transfer. Do not use PIP to transfer a bootable magnetic tape between tapes of different densities. The bootstrap block on the tape works at only one density.

Because PIP can write magnetic tapes that conform to a subset of the ANSI (American National Standard Institute) standard X3.27–1978 – magnetic tape and file structure for information interchange, file transfer operations with tape can preserve file attribute and data format information.

To ensure the proper transfer of information, the following procedures are recommended.

RSTS/E Disk to RSTS/E Tape transfer:

1. Mount the output magnetic tape and use the ASSIGN command to specify ANSI label format. For example:

   ```
   ASSIGN MT0:.ANSI
   ```

2. Use PIP to initialize the tape, write a volume ID, and set density and parity, if desired (see Section 6.11.6.21). For example:

```
PIP MT0:TAPIT/ZERO/DEN:800/PARITY:ODD
```

In the above command line, MT0: defines the device, TAPIT is the volume ID, and the DEN and PARITY switches request a density of 800 bits per inch and odd parity. If you omit the volume ID when zeroing an ANSI magnetic tape, you will get the error message ?ILLEGAL FILE NAME — FILE MMx:.

3. Use the /BL switch (see Section 16.2.2.4) to copy the files to the tape. For example:

```
PIP MT0:FORMS.DAT=FORMS.DAT/BL
```

The output file is created in U (undefined) format and all attribute information is retained in the file header label. Note that this type of transfer is only used between RSTS/E systems. Only PIP recognizes U format; it is not defined by ANSI standard X3.27–1978.

RSTS/E Disk to non-RSTS/E Tape transfer:

1. Mount, ASSIGN, and initialize the output tape as described in the previous steps 1 and 2.

2. To transfer a file without attributes that contains stream ASCII data to ANSI D record format, use the PIP /RMS:FA switch and option (see Section 6.11.6.15). For example:

```
PIP MM0:REPORT.DAT/RMS:FA=REPORT.DAT
```

To transfer FCS (RSX File Control Services) or RMS files with attributes, no switch is needed. However, only RMS sequential organization files can be copied. Thus, to transfer RMS relative and indexed organization files to ANSI labeled tape, you must first use the RMS conversion utility (RMSCNV) to convert them to sequential organization. Sequential files with attributes will be translated to their appropriate ANSI record format as described in Section 6.11.6.15.

Tape to disk transfer:

1. Mount the input tape and use the ASSIGN command to specify ANSI label format, or use the UMOUNT program to mount and assign the tape (see Section 6.18). For example:

```
MOUNT MT0:TAPIT/ANSI/NOVERIFY/DEN:800/PARITY:ODD
```

TAPIT is the ANSI volume ID for the tape.

2. Use PIP to copy the tape file to disk. For example:

```
PIP FORM.DAT=MT0:FORM.DAT
```

PIP reconstructs the attributes of the file from tape and restores the file to its former format.

If the file that is being transferred to disk was originally transferred to tape with the PIP /RMS:FA switch and option (to convert from stream ASCII to ANSI D format), the output disk file will be an RMS variable length record file with implied carriage return.

Note that you can use the /CL switch (see Section 6.11.6.6) to ensure that the files are copied with the correct cluster size. Also, if the tape contains RMS relative or indexed organization files that were converted to sequential organization, you must use RMSCNV to convert them back to their original format. Note that if the file is contained within one volume (tape reel), you can use either RMSCNV to convert to relative organization format or RMSIFL to load an indexed file directly from tape. For information on these RMS utility programs, refer to the *RMS–11 User's Guide.*

3. Dismount the tape to release the assignment made in step 1. In addition, you may include the /UNLOAD switch.

```
DISMOUNT MT0:/UNLOAD
```

NOTE

Because PIP has the ability to transfer files that are larger than 65535 blocks, PIP is the only method whereby you can selectively backup large files. That is, the RSTS/E BACKUP system program (see the *RSTS/E System Manager's Guide*) can not preserve files that are larger than 65535 blocks; the RSTS/E SAVE/RESTORE system program (see the *RSTS/E System Manager's Guide*) can preserve large files but only as an entire disk-to-disk or disk-to-tape copy. Thus, to preserve selected files from one disk to another disk or tape when those files are larger than 65535 blocks, use the PIP file transfer.

Input and output file specifications can also contain wildcard characters as described in Section 6.11.2. For example:

```
*DL1:*.*/CL:4=[2,2]*.MAC
```

This command line causes PIP to copy all files with a .MAC type in account [2,2] on the public structure to your account on RL01 unit 1. Protection

codes are preserved and all output files are created with a cluster size of 4 (see Section 6.11.6.6). Because PIP substitutes wildcards for missing elements of the command line, the following command is equivalent to the previous one:

```
*DL1:/CL:4=[2,2].MAC
```

Note that asterisk wildcards are legal for output file accounts and for input file accounts if the input file is on disk or magnetic tape.

Note that while wildcards can be very useful in transferring a set of files from one device to another, note that there is no advantage to simply copying files from one disk to another within the same account in the public structure. The RSTS/E system is designed to store files on the public structure so that they can be accessed efficiently. A copy such as that mentioned does not improve this efficiency and can get you into trouble if you use wildcards. An exceptional condition such as an I/O error or CTRL/C may cause both old and new copies of the file being transferred to be deleted.

If you find that it is necessary to transfer files between public disks in the same account, there are two safe ways to transfer the files:

1. Mount either the source or the destination disk as a private disk. This will prevent RSTS/E from deleting the source file when the destination file is opened. After you have confirmed that the operation has succeeded, delete the source files, dismount the private disk and remount it as a public disk.

2. Assign a temporary name to each file to transfer, and then delete the old files after the transfer operation has succeeded. For instance, to move all *.ODL files from DB1: to DB0:, you might use the following commands:

```
PIP DB1:*.XDL=DB1:*.ODL/RE    [Rename all the .ODL files on DB1:]
PIP DB0:*.ODL=DB1:*.XDL       [Transfer the files to DB0:]
PIP DB1:*.XDL/DE              [Delete the files from DB1:]
```

3. Use an intermediate (two transfers): go first to tape, or to a private disk, or to another account.

You can also use the question mark wildcard character in input or output file specifications. For output file specifications, PIP will supply for a question mark the corresponding character from the input file specification(s), if such exists. If the output specification is such that PIP tries to create a filename with imbedded spaces, PIP will print the error ?ILLEGAL FILENAME, ignore the request, and proceed to the next input file, if any. For example, suppose the files MYFIL1.TXT and MYFIL2.TXT exist in your account on the public disk structure.

```
*HS????.TXT=MYFIL1.TXT, MYFIL2.TXT
```

The above command will create two files HSFIL1.TXT and HSFIL2.TXT consisting of the files MYFIL1.TXT and MYFIL2.TXT, respectively.

Suppose the files M1.TXT and M2.TXT exist in your account on the public disk structure.

```
HS????.TXT=M1.TXT,M2.TXT
```

The above command will result in two copy operations. The file M1.TXT is copied to HS.TXT, and then the file M2.TXT is copied to HS.TXT. The final result, then, is one file named HS.TXT consisting of the contents of M2.TXT.

```
*????M?.TXT=M1.TXT,M2.TXT
```

The above command will result in two error messages; first, PIP tries to create a file named M1 M .TXT. Since this file name contains imbedded spaces, PIP displays the ?ILLEGAL FILENAME error message. It then proceeds to the next input file, resulting in an attempt at a file named M2 M .TXT. Again, the ?ILLEGAL FILENAME message is displayed.

If you specify a non-file structured device in the input file specification, PIP ignores the file name and type even if wildcards are present.

If you transfer data to the line printer and the printer goes off-line, PIP prints the message:

```
LINE PRINTER HUNG - PUT ONLINE TO CONTINUE OR TYPE CTRL/C
```

Following the message, PIP causes the terminal bell to signal every ten seconds until the printer is on-line or you type a CTRL/C. If you type a CTRL/C, PIP returns to prompt level.

6.11.6.1 Access Switch (/AC) — The /AC switch is applied only to disk resident input files. When you specify this switch, PIP changes the input file's date of last access to the current date prior to copying the file to the output file. If you do not specify this switch, PIP leaves the file's date of last access unchanged, and preserves it during the transfer. Note that even if the input disk is initialized for an access update on date of last write, the /AC switch causes the access date to change.

6.11.6.2 Append and Extend Switches (/AP and /EX) — The /AP and /EX are synonymous. When you specify either of these switches at the end of an input file list, PIP transfers the data from the input files (which can be on tape or disk) and appends it to the end of the output file (which must be on disk). This has the effect of extending the output file. For example:

```
*A.DAT=B.DATC.DAT/AP
```

causes A.DAT to contain its original data followed by the data that was transferred from B.DAT and C.DAT. Note that the use of these switches is not recommended if the output file has attributes and was created under control of the RSX File Control Services or RMS, because the end-of-file attribute is not updated.

If the specified output file does not exist, PIP returns the error ?CAN'T FIND FILE OR ACCOUNT and creates the file. Also, you cannot append a file to itself. If you try, PIP returns the error ?PROTECTION VIOLATION.

6.11.6.3 ASCII Switch (/AS) — The /AS switch is applied to output files that are to contain formatted ASCII data. When you specify this switch, PIP transfers the data from the input file, discards null and RUBOUT characters, and copies the data to the output file. This switch can be useful following an append transfer (see Section 6.11.6.2). That is, the /AP switch causes PIP to append input data to the last free block of the output file. If the last block contains null characters, these characters remain in the file following the transfer. You can use the /AS switch in a subsequent transfer to delete the null characters.

You should not use the /AS switch to transfer files that contain 8–bit binary data. The bytes in a virtual array or block I/O file can take any pattern of 8 bits, one of which can be CTRL/Z. Because PIP terminates the transfer when it encounters CTRL/Z, some data can be lost. To transfer such a file, use the /BL (Block) switch described in Section 6.11.6.4.

6.11.6.4 Block Switch (/BL) — The /BL switch causes PIP to perform a block-by-block image copy of the input file to the output file. No translation is performed on the data and all of the file's attributes are preserved. However, if an RMS sequential file is being transferred to a nondisk device without an /RMS or /BL switch in the command line, PIP automatically converts the RMS format to formatted ASCII or formatted binary. To override this conversion and preserve the RMS format, specify the /BL switch. The /RMS switch is described in Section 6.11.6.15. If you specify the /BL switch on a transfer to an ANSI labeled magnetic tape, PIP writes U format records and performs the block mode transfer. Note that you should use the /BL switch to correctly transfer RMS relative and indexed files between RSTS/E systems.

If you specify a single contiguous input disk file, the /BL switch, and a single output disk file, PIP attempts to create a contiguous disk file. If PIP fails to create the output file contiguously, the file is created non-contiguous and the % FILE CREATED NON–CONTIGUOUS message is displayed.

When PIP performs a block mode transfer (either by default or by means of a /BL specification) from a non-disk device to a disk, it examines the output file type. If the extension is executable (.BAC, .TSK, .SAV, etc.), PIP

creates an executable file (sets the executable bit) and names the file as executable under its appropriate run-time system.

NOTE

PIP determines if a file is compiled by comparing it to a list of default executable file types. PIP creates this list from the list of installed run-time systems that are current when PIP is invoked. If PIP detects conflicting types (for example, RSX and BASIC–PLUS–2 both use .TSK), PIP assigns the name of the first run-time system on its list whose type matches the output file type. See Section 6.11.6.12.

Note that a block mode transfer is actually an image mode transfer. Thus, a /BL specification allows transfers between DECtape (510 byte block) and disk or magnetic tape (512 byte block). When such a transfer takes place, /BL causes PIP to transfer each DECtape block to a disk block and fill the excess bytes in the last block with nulls. For a disk to DECtape transfer, the two extra bytes are placed in a new block; PIP does not pad excess bytes with nulls until the last output block is reached.

6.11.6.5 Block Size Switch (/BSIZE:n) — The /BSIZE switch allows you to specify a blocksize for output to magnetic tape. If this switch is not used, PIP transfers data to an output magnetic tape with a default blocksize of 512 bytes. The /BSIZE:n switch, where n is an even integer in the range of 18 to 4096 bytes, can be used to create a magnetic tape file whose blocksize is larger or smaller than the default of 512 bytes. If an illegal value for n is specified, PIP prints a BAD BLOCK SIZE error message.

The use of the /BSIZE switch is subject to the following restrictions:

1. The switch only applies to output on magnetic tape; PIP automatically handles input magnetic tape with blocksizes other than 512 bytes.
2. The switch can only be used for magnetic tape; its use on disk is ignored.
3. If the output tape is in ANSI format and is intended for interchange on another operating system, the blocksize specification must be an even integer between 18 and 2048 bytes.
4. If the output tape is intended for use on the RT–11 operating system, the blocksize must be 512 bytes.
5. The use of the /BSIZE switch can increase the amount of buffer space used by PIP. Thus, when reading or writing blocksizes larger than 512, invoke PIP with the CCL /SIZE:n switch or, if under the RT–11 Run-Time System, with the SIZE n immediate mode command.

6.11.6.6 Clustersize Switch (/CL:n) — The /CL:n switch specifies the cluster size in an output disk file specification. The cluster size you specify in n becomes the cluster size of the output file or files that PIP creates. The size you specify in n must be:

1. A power of 2,
2. Greater than or equal to the pack cluster size of the disk involved in the transfer,
3. Less than or equal to 256, or,
4. The negative of a power of 2 to specify either a cluster size of –n or the pack cluster size of the output device, whichever is greater.

When this switch is present on the PIP command line, it must immediately follow the output filename (or the /MO switch if it is also present, see Section 6.11.6.8). If you do not include the /CL switch in the command line, PIP preserves the cluster size of the input file during the transfer. For non-disk devices, the switch is ignored. For a complete discussion of cluster size, refer to the *RSTS/E Programming Manual.*

6.11.6.7 Go or Ignore Switches (/GO or /IG) — You can use the /GO or /IG switches to cause PIP to ignore ?DATA ERROR ON DEVICE or magnetic tape record length errors. If this error occurs in a record other than a label record, and one of these switches is present on the command line, PIP ignores the error and continues with the operation. With these switches, you can use PIP to partially salvage files from damaged or compromised media. Files whose label records are unreadable cannot be retrieved with PIP.

6.11.6.8 Mode Switch (/MO:n) — The /MO:n switch allows you to specify the mode in which files are to be opened. That is, RSTS/E allows you to specify modes in OPEN statements that set special characteristics for devices. These mode values and their uses are described in the *RSTS/E Programming Manual.*

6.11.6.9 New File and Retain Switches (/NE, /RET) — The /NE switch is specified for disk output files and causes PIP to create a new file with the current date as both date of creation and date of last access. The /RET switch is also specified for disk output files, but unlike /NE, it causes PIP to retain the creation date and last access date of the input file. The /RET switch is the default.*

---

* The system manager can install an optional feature patch that causes the /NE switch to be the default.

6.11.6.10 No Attributes Switch (/NOA) — The /NOA switch causes PIP to suppress the transfer of disk file attributes. That is, if the input file contains attribute information that you do not want to copy to the output file, specify /NOA in the command line. The presence of the switch instructs PIP to refrain from writing the input file's attributes to the output file. Without this switch, attributes are automatically transferred.

6.11.6.11 No Supersede Switch (/NOS) — For an output file, the /NOS switch prevents PIP from making a file transfer if the specified output file already exists. This switch allows you to protect the output file from any possible deletion or overwrite. Note that the action of this switch is independent of the file's protection code, which can also prohibit writing to the file.

When you specify /NOS and the output file does not exist, PIP creates the file and transfers the data. No error is returned. When you specify /NOS and the output file does exist, PIP displays the ?NAME OR ACCOUNT ALREADY EXISTS error and cancels the data transfer. If you specify the NOWARN option (/NOS:NOWARN) and the output file exists, PIP cancels the data transfer but does not display an error. If you use the QUERY or INSPECT option (/NOS:Q or /NOS:IN), PIP asks "OK to replace file filespec?" if the output file already exists. Type Y, YE, or YES to replace the file. Type N, NO, or <RETURN> to cancel the data transfer.

6.11.6.12 Run-Time System Name Switch (/RTS:name) — For a file structured disk transfer, the /RTS switch allows you to change the run-time system name of the output file. If there is no output file specification, PIP changes the name of the input file's run-time system to the name you specify in the switch.

If you do not include the /RTS switch in the command line, PIP transfers a run-time system name to the output file based on the following conditions:

1. If the input file is on a file structured disk, the input file's run-time system name is transferred to the output file, or
2. PIP searches for a run-time system whose runnable file type matches the output file type. The first such run-time system encountered is used to assign a name to the output file. Note that if this condition is met, PIP also assigns an executable protection code to the output file.
3. If neither condition 1 nor condition 2 is met, PIP assigns the run-time system name under which it is running to the output file.

6.11.6.13 Update Switch (/UP) — The /UP switch is used for output files to disk and causes PIP to attempt an update in existing files of the same name before it creates a new file. That is, when you include /UP in the command line, PIP searches the output device for files that match the output file

specification. If it finds a matching file, PIP opens the file and replaces the file's data with that of the corresponding input file. If PIP is unable to locate a matching output file, it creates a new file.

If a matching output file is found that is larger than the corresponding input file, PIP performs the following operations:

1. PIP writes the input file's data to the output file without disturbing the output file's excess data. For example, a 34–block input file changes only the first 34 blocks of a 48–block output file.
2. If the /LO or /WA switches are also specified (see Section 6.11.6.17), PIP reports that the input file was COPIED INTO PREFIX of the output file.

If a matching output file is found that is shorter than the corresponding input file, PIP extends the output file to contain the excess data. For example, if the input file is 48 blocks and the output file is 34 blocks, the output file is extended to 48 blocks and its data is replaced by that of the input file.

6.11.6.14 Multi-Volume ANSI Magnetic Tape File Transfer — PIP allows you to transfer files from any input device to one or more ANSI labeled magnetic tapes, thus creating a volume set. You can also begin a file on one volume of the set and end it on another, assign identification labels to the volumes of the set, and transfer the content of the volume set to any output device. Note that DOS format does not allow files to be segmented across tape volumes, and thus does not allow multi-volume transfers.

To perform a multi-volume transfer, the output magnetic tapes must be initialized (with the /ZE switch, see Section 6.11.6.20). The first tape of the output set may contain some data, but each succeeding tape must be free of data. PIP allows you to specify volume identifications for each output tape, which allows you to create an identifiable volume set containing the transferred data.*

When you initiate a multi-volume transfer, PIP prints a series of messages that notify you of the end of the output tape. PIP then asks you to specify the device name and unit number of the second output tape (the second volume). This process is repeated for each end-of-tape until all of the specified input is exhausted. Also, after PIP reaches the end of an output tape volume, it rewinds the tape and takes it off-line.

To ensure that files are retrieved in the correct order, PIP assigns section and sequence numbers to the files it transfers. These numbers are checked when files are transferred from the tape volumes. These numbers are internal and not user-specified.

---

*The system manager can install an optional feature patch that causes PIP to require a volume ID specification.

PIP allows you to use file transfer switches (see Section 6.11.6) on multi-volume transfers. The directory switches (see Section 6.2.8), however, return directory listings on only a single volume. Thus, if a file is segmented across volumes, a directory of all volumes is required to show the true state of the file.

Consider the following example, which transfers the contents of a user account to three magnetic tape volumes:

```
RUN $PIP
*MM0:*.*=[20,25]*.*/BL

%END OF ANSI MAGTAPE OUTPUT VOLUME HAS
%BEEN REACHED

%PLEASE TYPE THE DEVICE NAME AND UNIT
%NUMBER OF THE DRIVE WHERE THE NEXT
%VOLUME MAY BE FOUND

                                        ?MM1:TAPE2

%END OF ANSI MAGTAPE OUTPUT VOLUME HAS
%BEEN REACHED

%PLEASE TYPE THE DEVICE NAME AND UNIT
%NUMBER OF THE DRIVE WHERE THE NEXT
%VOLUME MAY BE FOUND

                                        ?MM0:TAPE3

*
```

In this example, you specify the block mode transfer of all files in account [20,25] on the system disk to the tape mounted on MM0:. When PIP reaches the end of the output tape, it displays a message that requests the next volume. In response to the prompt, type the device name and unit number (MM1:) and the optional volume ID (TAPE2). PIP checks the output volume ID, if specified, and that the tape contains ANSI labels and is initialized. If the ID is valid, the tape is properly initialized, and free of data, PIP continues with the transfer. This process is repeated each time the end of a tape is reached until all of the specified files on the input device are transferred.

When the transfer is complete, PIP displays its asterisk prompt.

Consider the following example, which transfers the contents of three tape volumes to a user account:

```
RUN $PIP
*[20,25]=MM0:*.*/BL

%END OF ANSI MAGTAPE INPUT VOLUME HAS
%BEEN REACHED
```

```
%PLEASE TYPE THE DEVICE NAME AND UNIT
%NUMBER OF THE DRIVE WHERE THE NEXT
%VOLUME MAY BE FOUND

                                        ?MM1:TAPE2

%END OF ANSI MAGTAPE INPUT VOLUME HAS
%BEEN REACHED

%PLEASE TYPE THE DEVICE NAME AND UNIT
%NUMBER OF THE DRIVE WHERE THE NEXT
%VOLUME MAY BE FOUND

                                        ?MM0:TAPE3

*
```

In this example, you specify the block mode transfer of all files on the tape mounted on MM0: to account [20,25] on the system disk. When PIP reaches the end of the input tape, it displays a message that requests the next volume. In response to the prompt, type the device name and unit number (MM1:) and the optional volume ID (TAPE2). PIP checks the input volume ID, if specified, and ensures that the file is in the correct sequence (i.e., PIP checks the section and sequence numbers it assigned to the files). PIP also checks that the input tape contains ANSI labels. If the volume ID is valid, the files correct, and the tape has ANSI labels, PIP continues with the transfer. This process is repeated each time the end of a tape is reached until all of the files on the input tape volumes are transferred.

When the transfer is complete, PIP displays its asterisk prompt.

Note that if you respond to the volume prompt (?) with a CTRL/Z, PIP interrupts the transfer and terminates.

6.11.6.15 File Operations Involving Attributes – (/RMS:options) — A RSTS/E disk file may or may not have attributes, depending on the type of file. For example, all RMS files have attributes, while files created by BASIC–PLUS have no attributes. In addition, RMS files and non-RMS files are formatted differently. That is, RMS files can have fixed length or variable length record format and one of several carriage control formats: implied, embedded, FORTRAN, or print format. Non–RMS files can have stream (formatted ASCII) records, binary record format, or BASIC–PLUS block I/O, or virtual array format files. Similarly, DOS labeled magnetic tape differs from ANSI labeled tape in file attributes and format. DOS labeled tapes have no attributes and can contain formatted ASCII or formatted binary files. ANSI labeled tapes have limited file attribute information that is similar to RMS sequential files.

As PIP transfers input disk file data to output disk files, it allows you to translate from one format to another. PIP provides this capability by means of the /RMS switch and its options as described in Section 6.11.6.15.

If you do not specify the /RMS switch on a transfer of disk files to DOS labeled magnetic tape, the following default transfers occur:

1. If the input RMS file has implied or RMS FORTRAN carriage control, or if it is a print format file, PIP creates an output file in stream (formatted ASCII) format.
2. In all other cases, PIP creates a file in formatted binary.

If you do not specify the /RMS switch on a transfer of disk files to ANSI labeled magnetic tape, the following default transfers occur:

1. If the input RMS file contains fixed length records, PIP creates an output magnetic tape file in ANSI F record format (fixed length format).
2. If the input RMS file contains variable length records, PIP creates an output magnetic tape in ANSI D record format (variable length format).
3. If the input file contains no attributes, PIP creates an output magnetic tape file in U (undefined) record format. Tapes with U format may not be transferable to non–RSTS/E systems.

If you do not specify the /RMS switch on an ANSI magnetic tape to disk transfer, the following default transfers occur:

1. If the input magnetic tape file is in ANSI F record format, PIP creates an output disk file with RMS fixed length records.
2. If the input magnetic tape file is in ANSI D record format, PIP creates an output disk file with RMS variable length records.
3. If the input magnetic tape contains ANSI labels and is not in F or D record format or the tape format does not contain a header label 2 (HDR2), PIP performs a block mode transfer (see Section 6.11.6.4).

The use of the PIP /RMS switch options to transfer files with attributes is subject to the following restrictions:

1. PIP cannot concatenate two or more RMS data files. An attempt to do so results in a file with invalid attributes. That is, all the data in the file is not accessible.
2. If a file contains variable length records or fixed length records with RMS FORTRAN carriage control, PIP can translate the file to stream (formatted ASCII) records.
3. If a file is an object module, PIP can translate the file to formatted binary.

4. To concatenate RMS files, use the RMS utility RMSCNV as described in the *RMS–11 User's Guide.*

The /RMS switch is used only on input or output files that are resident on disk devices or ANSI labeled magnetic tape. The switch allows you to force the translation of the file's data format when PIP transfers the file.

If you include the switch in the command line for input disk files, PIP translates RMS format to formatted ASCII or formatted binary. If you include the switch in the command line for output disk files, PIP translates formatted ASCII or formatted binary to RMS format.

If you do not specify this switch, PIP transfers the file as described at the beginning of this section. Note that you cannot apply this switch to both input and output files on the same command line.

When you specify /RMS, PIP determines whether to translate between RMS format and formatted ASCII or formatted binary. However, PIP allows you to include one of five options in the switch to specify a particular translation. The options available to you are:

:FA For input or output disk files, translate to or from formatted ASCII.

:FB For input or output disk files, translate to or from formatted binary.

:FTN For input disk files, translate RMS FORTRAN carriage control to formatted ASCII.

:IM For input disk files, translate from RMS format to formatted ASCII or formatted binary (as with the switch with no option), but also remove record lengths from variable length records and padding bytes.

:PRN For input disk files, translate RMS print file format to formatted ASCII.

Note that to translate a Task Builder .MAP file to formatted ASCII, you must use the /RMS:FA switch.

6.11.6.16 Date Related Switches (/DLA, /CRE, /AF, /BE, /ON, /SIN, /TO, and /UN) — The date switches allow you to specify the inclusion or exclusion of certain files in a PIP transfer based on the creation date or date of last access of those files. These switches are especially useful for modifying wildcard file specifications.

By default, file transfers are based on the creation date.* However, you can specify a /DLA or /CRE switch in the command line to override the installed default. The /DLA switch causes PIP to use the date of last access; the /CRE switch causes PIP to use the date of creation.

The Today switch, /TO, does not contain an argument and causes PIP to operate only on those files that were created or accessed on the current date.

The On switch, /ON:date, contains a single argument that specifies a date in the form:

```
/ON:dd-mmm-yy
```

where dd is the day of the month, mmm are the first three letters of the month, and yy are the last two digits of the year. When you include the /ON switch in the command line, PIP operates only on those files with a date that matches the switch specification.

The After, Before, Since, and Until switches, when used in combination, allow you to specify a range of dates. That is, only one switch is necessary in the PIP command line but two switches can be used to specify an upper boundary and a lower boundary. When you specify a range in the command line, PIP processes only those files that fall within the range and ignores those whose dates are outside the specification. Note that because of a conflict between the PIP switches /SIZE and /SINCE, the minimum abbreviation for the /SINCE switch is /SIN.

You specify a date in these switches in the following format:

```
:dd-mmm-yy
```

as described for the /ON switch. Note that you can omit the year from a date specification and accept the default of the current year.

Consider the following examples:

Switches	Lower Boundary	Upper Boundary
/SIN:10–APR–81 /UN:23–SEP–81	10–APR–81	23–SEP–81
/AF:10–APR–81 /UN:23–SEP–81	11–APR–81	23–SEP–81
/SIN:19–APR–81 /BE:06–JUL–81	19–APR–81	05–JUL–81
/AF:08–MAY–81 /BE:22–SEP–81	09–MAY–81	21–SEP–81

* The system manager can install an optional features patch that causes PIP to use the date of last access instead of the date of creation.

6.11.6.17 File Operation Switches (/HA, /IN, /QU, /LO, /NOL, /WA, /NO, /RW:NO, /VE, /ID) — The switches described in the following paragraphs can be applied to the four major operations performed by PIP; copy, delete, rename, and directory listing operations. You can use these switches to speed PIP operations and as auxiliaries to those operations.

Halt Switch (/HA) — You can specify the /HA switch when the PIP operation is to conduct wildcard searches on magnetic tape devices. When /HA is included in the command line, it causes PIP to stop processing when it encounters a file on magnetic tape that does not match the wildcard specification. Without this switch, PIP continues its search until it reaches the end of the tape.

Inspect or Query Switch (/IN:option or /QU:option) — You can specify the /IN switch when the PIP operation involves wildcard specifications. The switch can appear at any point in the command line and, when specified, causes PIP to display at your terminal the file name of each file that matches the wildcard specifications. The file names display one at a time and PIP allows you to respond with one of the following:

Y (for yes)	Causes PIP to execute the operation on this file and search for the next matching file.
CTRL/Z	Causes PIP to terminate processing of the current wildcard specification without processing this file.
any other response	Causes PIP to skip this file and search for the next matching file.

Note that when the /IN switch appears in the command line, it applies to every input file wildcard specification.

The /IN switch accepts an option (/IN:S) that causes PIP to print the file name, file type, filesize, protection code, creation date, and date of last access of each file that matches the wildcard specifications.

Log, No Log, or Watch Switches (/LO, /NOL, or /WA) — The /LO or /WA switches are synonymous and can be specified at any point in the command line.* When you specify /LO or /WA, PIP displays on your terminal a report on all actions taken during execution.

The /NOL switch is used to suppress the display of the execution report and is used when logging is enabled as the system default.

---

* The system manager can install an optional feature patch which causes logging to occur by default.

No Rewind Switches (/NO or /RW:NO) — When PIP begins its search for an input file on magnetic tape, it first rewinds the tape. You can specify the /NO or /RW:NO switches at any point in the command line and cause PIP to suppress the automatic rewind on tape. When these switches are specified, PIP begins its file search at the current tape position.

Version or Identification Switch (/VE or /ID) — You can specify the /VE or /ID switch at any point in the command line and cause PIP to print information about the version and current configuration on the terminal. /ID includes which optional feature patches your system manager has installed.

6.11.6.18 File Deletion Switches (/DE, /ER, /WO, and /WIPE) — When you apply the /DE switch to a command line input file specification, it causes PIP to delete all files that match that specification. The switch can only be applied to a disk or DECtape input file specification. If you do not specify a file name or a type, PIP prints the following error:

```
?FILENAME AND TYPE MUST BE SPECIFIED
```

Note that PIP allows you to use wildcard characters in the file specification. If this is the case, you may also include the /IN switch (see Section 6.11.6.17) in the command line.

The /DE switch applies to all input file specifications on the command line in which it appears. Therefore, you should not attempt to specify any other PIP operation in a command line that contains a /DE switch.

PIP allows you to modify the /DE switch with a NO option, /DE:NO. When NO is appended, it instructs PIP to suppress the display of an error message if the file to be deleted does not exist.

If the file to be deleted has the privileged (128.) bit set in its protection code, PIP writes zeroes on the file, renames it to a nonprivileged code, and deletes it. When a file with the privileged bit is deleted from the system by means of the UNSAVE command or KILL statement, the File Processor performs the same actions as the PIP delete. Therefore, if you desire such a delete operation, you should use PIP (which is faster) rather than the File Processor, which is subject to greater system demand.

In conjunction with the /DE switch, PIP allows you to specify an /ER, /WO, or /WIPE switch. When you include one of these switches with /DE in the command line, PIP writes zeroes in the file prior to its deletion. As with the /DE switch, these switches apply to every input file specification on the command line.

6.11.6.19 File Rename Switch (/RE:option) — When you apply the /RE switch to a disk or DECtape input file specification, PIP changes the input filename to that of the output file specification.

The /RE switch can be used with wildcard specification to cause multiple file renames. Also, PIP requires that you specify only those elements of the output file specification that will change. For example:

```
PIP *.C??=STAT?.B?S/RE
```

causes PIP to change the file type of every file whose file name begins with STAT and whose type begins with B and ends with S. The type is changed, in each case, such that the first character is C. Note that if no input file specification appears on the command line, PIP renames all files in the current account on the public structure.

When you specify the /RE switch, PIP does not transfer any data.

Consider the following:

```
* newname.type/PROT:n =oldname.type/RE
```

Upon execution, the file oldname.type is renamed to newname.type and its protection code is changed to that specified in the command line. However, the data in newname is unchanged.

To change only the protection code, include the new protection code on the input file specification and omit the output file specification. For example:

```
*oldname.type/PROT:n/RE
```

If you attempt to rename a file that does not exist, PIP returns the error ?CAN'T FIND FILE OR ACCOUNT, skips that operation, and continues processing. If you attempt to rename a file to a name that already exists, PIP returns the error ?NAME OR ACOUNT NOW EXISTS, skips that operation, and continues processing. Note that you can use the NOWARN option (/RE:NOWARN) to suppress the display of the ?NAME OR ACCOUNT NOW EXISTS message.

6.11.6.20 Directory Listing Switches (/BR, /F, /DI, /LI, and /S) — The PIP program provides five switches and several options that allow you to display a directory listing. This listing is similar to that which you can obtain using the DIRECT system program (see Section 6.2).

The /DI and /LI switches cause PIP to display a full directory listing and also accept options that allow you to modify the listing. Options are specified in the following format:

```
/DI:option
```

or

```
/LI:option
```

The /BR and /F switches do not accept an option specification, and when included in the command line, cause PIP to display an abbreviated form of the directory listing. That is, only the file names and types are printed. The /S switch causes PIP to display a full directory that is similar to the slow listing obtained from the DIRECT system program (see Section 6.2).

The /DI, /LI, /S, /F, and /BR switches can display the directory of any file-structured device. When one of these switches appears in a command line file specification, which can contain wildcards, only the files that match that specification are displayed. PIP displays the listing on your terminal by default but you can output the listing to another device or file with an output specification.

Table 6–11 lists the options that you can specify to the /DI or /LI switches.

Table 6–11: Directory Listing Options

Option	Result
:AL	Displays file name, type, and the number of blocks allocated to the file.
:AT	Same as :SA.
:CL	Displays file name, type, and file cluster size.
:DA	Displays file name, type, and file creation date.
:DI	Displays file name, type, size, protection code, creation date, and column headers.
:EX	Displays file name and type same as :TY (provided for compatibility with previous versions of RSTS/E).
:FU	Full listing, print all information except file attributes.
:HD	Includes column headers in the listing.
:LA	Displays file name, type, and date of last access or date of last write (depending on disk initialization).
:NA	Displays file name.
:OA	Displays file name, type, and octal representation of file attributes.
:PR	Displays file name, type, and protection code.
:RT	Displays file name, type, and associated run-time system.
:S	Slow listing, prints all information.
:SA	Displays file name, type, symbolic file attributes (see Section 11.6), and caching status. Caching is indicated by one of the following entries: CACHE:ON:RAN; file will be automatically cached randomly when open. CACHE:ON:SEQ; file will be automatically cached sequentially when open. CACHE:OFF:SEQ; file will not be automatically cached, but if cached, it will be cached sequentially.
:SI	Displays file name, type, and file size.

(continued on next page)

Table 6–11: Directory Listing Options (Cont.)

Option	Result
:SU	Displays only a summary report.
:SZ	See :SI.
:TI	Displays file name, type, and date and time of creation.
:TY	Displays file name and type.
:W	Displays file name and type across the length of the page or screen.
:WI	Displays file names only across the length of the page or screen.

Note that the :OA, :SA, :AT, and :S options cause PIP to display file attributes in octal or symbolic representation. Section 6.2.2 describes the format and meaning of these attribute representations.

6.11.6.21 Zeroing Directories Switch (/ZE) — To cause PIP to zero (i.e., initialize) a device or account directory, include the /ZE switch in the PIP command line. When you specify this switch, it causes PIP to delete all of the files stored in a specified account on disk or, if you specify a magnetic tape or DECtape, to initialize the volume mounted on the specified drive.

The use of the /ZE switch is subject to the following restrictions:

1. No output file specification can be present on the command line and only one input file specification is allowed. The format is as follows:

   ```
   *dev:[proj,prog]/ZE
   ```

   Note that only a privileged user can specify a zeroing of another's account.

2. To initialize an ANSI labeled magnetic tape, specify the device and a filename, which represents the volume label.

3. If a device is not specified, PIP assumes the public disk structure and the current account.

When PIP detects the /ZE switch in its command line, it prints the following prompt:

```
REALLY ZERO dev:[proj,prog] ?
```

Two possible replies are available to you:

Y confirms that you wish PIP to zero the device or account.

any other response — aborts the operation and returns you to the PIP command level.

The /ZE switch also permits you to specify a change in the density or parity of the magnetic tape prior to the zero operation. System-wide defaults for magnetic tape density, parity, and label are set by the system manager during system initialization. If you wish the system to accept a tape with non-default characteristics, you should ASSIGN the tape prior to setting those characteristics (with MOUNT or PIP). Note that when the tape is DEASSIGNed, the system reverts to the default characteristics set during system initialization. The specifications for density and parity are as follows:

```
/ZE/DEN:n
```

and

```
/ZE/PAR:parity
```

The /ZE/DEN:n switch combination causes PIP to set the density of the tape prior to initializing the tape. This operation results in a tape initialized at the desired density. The densities that you can specify in n are as follows:

800 — BPI for 9–track tape.

1600 — BPI for 9–track tape.

The /ZE/PAR:parity switch combination causes PIP to set the desired parity of the tape prior to initializing the tape. This operation results in a tape initialized at the desired parity. The parities that you can specify are as follows:

/ZE/PAR:ODD — for odd parity.
/ZE/PAR:EVEN — for even parity.

Note that 1600 BPI tape can only support odd parity. Also, unless the tape is to be processed on a system that requires even parity, odd parity is recommended for all output tapes.

A zero operation, which contains a volume label specification, is recommended for all new tapes.

6.11.6.22 Privileged Only Operations (/LOCK and /PRIOR) — The /LOCK and /PRIOR switches can be specified in a PIP command line only if the user is privileged.

When a privileged user specifies /LOCK in the command line, it causes the system to lock PIP into memory for the duration of the current operation.

When a privileged user specifies /PRIOR in the command line, it causes the system to run PIP at a special priority for the duration of the current operation. If this switch is not specified, PIP runs at the current user's default priority. When /PRIOR appears in the command line, the system boosts the priority of PIP by 4.

6.11.7 PIP Error Messages

Two types of errors are possible during a PIP operation: RSTS/E system errors (such as ?CAN'T FIND FILE OR ACCOUNT) and PIP program errors. Appendix C describes the RSTS/E errors and the recovery actions that you can take. Errors that are distinctive to PIP are as follows:

```
?WRITE FAILURE
?CLOSE FAILURE
?READ FAILURE
?LOOKUP FAILURE
?ENTER FAILURE
?RENAME FAILURE
?ZERO FAILURE
```

These errors indicate problems within PIP itself and the only recovery is to retry the operation (the error may be due to a transient condition). If the error continues, notify your system manager.

6.12 Formatting a Post-Mortem Dump: PMDUMP

PMDUMP is a system library program that formats the contents of a post-mortem dump into human-readable form and allows you to copy it to a file on any RSTS/E device. The post-mortem dump is written in binary by the RSX run-time system or an RSX-based run-time system such as BASIC–PLUS–2. The information contained in the dump allows a system programmer or software specialist to view the actual state of the program at the time it aborted. When you invoke the PMDUMP program, it performs the following operations and writes the information to a user-specified output file:

1. Formats the binary dump and produces a human-readable dump of the contents of low-core memory.
2. Produces an octal dump of the user's task.

When the program completes its operation, it returns control to RSTS/E command level.

6.12.1 When to Use PMDUMP

Post-Mortem dumps are generated under control of the RSX Run-Time System and are used to determine the state of a task after that task has aborted. Post-Mortem dumps do not return an on-line snapshot of the state of a job. However, the following conditions must be true for PMDUMP to be successful in formatting the dump:

1. The run-time system that controls your job must be RSX or RSX-based.
2. The task to be dumped must have already aborted, i.e., terminated abnormally.
3. A request for a post-mortem dump must be specified when the program is input to the Task Builder, i.e., the /PM switch must be used as described in the *RSTS/E Task Builder Reference Manual.*

6.12.2 How to Invoke PMDUMP

To invoke PMDUMP, type:

```
RUN $PMDUMP
```

or a CCL command, if one is installed.

If you invoke PMDUMP with the RUN command, the program prints a header line followed by a prompt (#) to indicate its readiness to accept command input.

In response to the prompt, type one of the following:

RETURN or LINE FEED	accepts a complete default.
CTRL/Z	terminates PMDUMP.
outfil.ext=infil.ext	specifies the input and output files to be used by the program.

If you do not specify an input file, the program defaults the filename to the following:

```
PMDnnn.PMD
```

Where nnn is the current job number. Note that this filename construct is identical to that of the dump file created by the RSX Run-Time System.

If you do not specify an output file, the program defaults the filename to that specified or defaulted for input. However, the program substitutes the extension .LST for the input extension of .PMD.

The program defaults for all elements of the file specification are as follows:

device	SY:
account	the current account
filename	the input filename
extension	.LST for output, .PMD for input

If you type the RETURN key in response to the prompt, the following defaults are applied:

1. The program places the files in the current account on the public disk structure (SY:).
2. The input filename is PMDnnn.PMD.
3. The output filename is PMDnnn.LST.

6.12.3 Contents of the Post-Mortem Dump

The following description of the contents of the PMDUMP listing is keyed to the example in Figure 6–1.

Item	Description
1	The name and account of the task being dumped and the date and time that the dump was generated. This information prints as a header line on each page of the dump listing.
2	The program name, or the name you assigned in the Task Builder TASK option, for the task being dumped. Partition name is GEN or is the name you specified in the Task Builder PAR option.

(continued on next page)

Item	Description
3	The size of the task in octal and decimal words. The size is given for the task when it was initially loaded and also for when the dump was made.
4	The synchronous and asynchronous system traps in effect when the task was dumped.
5	The task status flags that were set at the time of the dump.
6	The entry point of the program, either by line number in Parameter Entry or by CCL command. CCL Entry also indicates whether a /DETACH or /SIZE switch is specified. Directive Status and Impure Area Pointers are used by the RSX emulation code.
7	The logical unit assignments in effect at the time of the dump. This information includes the device name, the status flags set for that device, and the allocated buffer space in octal and decimal words.
8	A breakdown of the task's keyword value. The last two octal characters of the value represent those set by the user program. The other flags are set by the run-time system and by the RSTS/E Monitor.
9	The contents, in octal, of the file request queue block (FIRQB) and the transfer request block (XRB). This information represents the Monitor requests for file and I/O operations at the time of the dump.
10	The contents of the job's core common area when the dump was made, in ASCII characters and octal byte values.
11	The assigned project-programmer number, default protection code, and logical name table entries assigned by the user job at the time the dump was made.
12	The program counter and processor status word at the time the task was loaded. This information represents initial conditions.
13	The general registers, stack pointer, and processor status at the time of the dump (contrast with item 12).
14	The task stack at the time of the dump.
15	An octal dump of the task. The first 1000 octal bytes reflect the status printed in the previous portion of the listing.

Figure 6–1: Post-Mortem Dump

```
1 Post-Mortem Dump of SY:[1,201]PMD030.PMD on 20-Jul-78 at 09:15 AM
  Formatted Dump of Low Memory


2 Task Name    : FOO
  Partition Name : GEN

  Task Size (without Extension) : 002040 (1056.) words
3 Load Size of Task             : 004000 (2048.) words
  Current Task Size             : 004000 (2048.) words
```

(continued on next page)

Figure 6–1: Post-Mortem Dump (Cont.)

```
  ODT SST Vector Address    : 000000
  ODT SST Vector Length     :  000 (0.)
4 Task SST Vector Address   : 000000
  Task SST Vector Length    :  000 (0.)
  FPP AST Service Address   : 100124
  CTRL/C AST Service Address : 000000

5 Task Flags : 010000 (4096.)    Task Flags Set :
                                  Post-Mortem Dump Requested


  User Parameter on Entry : 000000
  CCL Entry Flags         : 000000
  Directive Status Word   : 000001
6
  FCS Impure Area Pointer     : 000000
  OTS Impure Area Pointer     : 002372
  Auto-Load Impure Area Pointer : 004044
  Extended Impure Area Pointer : 000000

7  LUN (Logical Unit Number) Table :

  LUN #0 :
          Device Name : TI:
          Device Flags : 000007  Flags are :
                                  Record Oriented Device
                                  Carriage Control Device
                                  Terminal Device
          Buffer Size : 000200 (128.)

  LUN #1 :
          Device Name : SY:
          Device Flags : 000010  Flags are :
                                  Directory Device
          Buffer Size : 001000 (512.)

  LUN #2 :
          Device Name : SY:
          Device Flags : 000010  Flags are:
                                  Directory Device
          Buffer Size : 001000 (512.)

  LUN #3 :
          Device Name : SY:
          Device Flags : 000010  Flags are :
                                  Directory Device
          Buffer Size : 001000 (512.)

  LUN #4 :
          Device Name : SY:
          Device Flags : 000010  Flags are :
                                  Directory Device
          Buffer Size : 001000 (512.)

  LUN #5 :
          Device Name : SY:
          Device Flags : 000010  Flags are :
                                  Directory Device
          Buffer Size : 001000 (512.)
```

(continued on next page)

Figure 6–1: Post-Mortem Dump (Cont.)

```
LUN #6 :
        Device Name : SY:
        Device Flags : 000010  Flags are :
                                Directory Device
        Buffer Size : 001000 (512.)

LUN #7 :
        Device Name : SY:
        Device Flags : 000010  Flags are :
                                Directory Device
        Buffer Size : 001000 (512.)

LUN #8 :
        Device Name : SY:
        Device Flags : 000010  Flags are :
                                Directory Device
        Buffer Size : 001000 (512.)

LUN #9 :
        Device Name : SY:
        Device Flags : 000010  Flags are :
                                Directory Device
        Buffer Size : 001000 (512.)

LUN #10 :
        Device Name : SY:
        Device Flags : 000010  Flags are :
                                Directory Device
        Buffer Size : 001000 (512.)

LUN #11 :
        Device Name : SY:
        Device Flags : 000010  Flags are :
                                Directory Device
        Buffer Size : 001000 (512.)

LUN #12 :
        Device Name : SY:
        Device Flags : 000010  Flags are :
                                Directory Device
        Buffer Size : 001000 (512.)


8 Job Keyword :
        Value : 103300

        User Controlled : 00

        Run-Time System Controlled :
                Post-Mortem Dump Enabled
                Traps Enabled
                No Temporary Privileges

        System Controlled :
                JFFPP
                JFPRIV
```

(continued on next page)

Figure 6–1: Post-Mortem Dump (Cont.)

```
9 Job's FIRQB :
          IOSTS: 000 (0.)

          FIRQB:
           000000
           010074
           000034
           000401
           006273
           034330
           021042
           000021
           001000
           000000
           100000
           024010
           041104
           176001
           013560
           000000

   Job's XRB :
          000002
          000020
          000020
          000020
          000711
          177770
          000001
          000000
          000000

10 Job's Core Common :
       Length : 8
     T   K   B     @   F   O   O
     124 113 102 040 100 106 117 117

11 User Assignable PPN : <None>
   User Default Protection : <None>
   User Logical Name Table :
       <Table Empty>

12 Initial PC : 002000
   Initial PS : 174017

13 Registers at time of Dump :
          R0       000040
          R1       004020
          R2       121030
          R3       002372
          R4       002112
          R5       001774
          SP       001762
          PC       120150
          PS       000000
```

(continued on next page)

Figure 6–1: Post-Mortem Dump (Cont.)

```
14 Job's Stack :
          Initial SP = 002000   Final SP = 001762

    SP
    !
    V
   001762 / 000000 121036 174000 101744 002112 000001 003770 004437


15  Post-Mortem Dump of SY:[1,201]PMD030.PMD on 20-Jul-78 at 09:15 AM
                              Memory Dump, in Octal

         00   02   04   06   10   12   14   16
  000000 / 023747 000000 026226 000000 000041 000000 000000 000000
  000020 / 000000 100124 000000 010000 000000 000000 000100 000100
  000040 / 000014 000000 000000 000001 000000 002372 004044 000000
  000060 / 176347 002620 000450 000700 000000 010000 000170 166214
  000100 / 174000 001740 003400 000000 000000 017112 112561 064143
  000120 / 146264 023032 115635 061043 066063 013032 110664 061063
  000140 / 106063 053072 115704 061063 006063 017112 112561 064243
  000160 / 146264 023032 115635 061043 066063 000000 000000 000000
  000200 / 044524 177777 000007 000200 054523 177777 000010 001000
  000220 / 054523 177777 000010 001000 054523 177777 000010 001000
  000240 / 054523 177777 000010 001000 054523 177777 000010 001000
  000260 / 054523 177777 000010 001000 054523 177777 000010 001000
  000300 / 054523 177777 000010 001000 054523 177777 000010 001000
  000320 / 054523 177777 000010 001000 054523 177777 000010 001000
  000340 / 054523 177777 000010 001000 000000 000000 000000 000000
  000360 / 000000 000000 000000 000000 054523 177777 000010 001000


        00   02   04   06   10   12   14   16
  000400 / 103300 000000 010074 000034 000401 006273 034330 021042
  000420 / 000021 001000 000000 100000 024010 041104 176001 013560
  000440 / 000000 010000 010000 000000 000036 000000 000000 000000
  000460 / 052010 041113 040040 047506 000117 042057 046525 027520
  000500 / 047516 040515 000120 000000 000000 000000 000000 000000
  000020 / 000000 000000 000000 000000 000000 000000 000000 000000
  000540 / 000000 000000 000000 000000 000000 000000 000000 000000
  000560 / 000000 000000 000000 000000 000000 000000 000000 000000
  000600 / 000000 000000 000000 000000 000000 000000 000000 000000
  000620 / 000000 000000 000000 000000 000000 000000 000000 000000
  000640 / 000000 000000 000000 000000 000000 000000 000000 000000
  000660 / 001742 000002 000020 000020 000020 000711 177770 000001
  000700 / 000000 000000 000000 000000 000000 000000 000000 000000
  000720 / 000000 140300 000000 001774 002000 174017 000000 000000
  000740 / 000000 000000 000000 000000 000000 000000 000000 000000
  000760 / 000000 000000 000000 000000 000000 000000 000000 000000
```

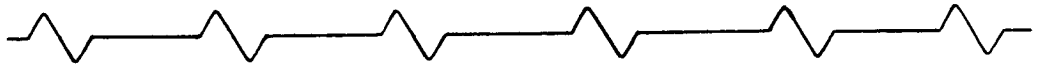
(continued on next page)

Figure 6–1: Post-Mortem Dump (Cont.)

```
      00    02    04    06    10    12    14    16
001000 / 000000 000000 000000 000000 000000 000000 000000 000000
001020 / 000000 000000 000000 000000 000000 000000 000000 000000
001040 / 000000 000000 000000 000000 000000 000000 000000 000000
001060 / 000000 000000 000000 000000 000000 000000 000000 000000
001100 / 000000 000000 000000 000000 000000 000000 000000 000000
001120 / 000000 000000 000000 000000 000000 000000 000000 000000
001140 / 000000 000000 000000 000000 000000 000000 000000 000000
001160 / 000000 000000 000000 000000 000000 000000 000000 000000
001200 / 000000 000000 000000 000000 000000 000000 000000 000000
001220 / 000000 000000 000000 000000 000000 000000 000000 000000
001240 / 000000 000000 000000 000000 000000 000000 000000 000000
001260 / 000000 000000 000000 000000 000000 000000 000000 000000
001300 / 000000 000000 000000 000000 000000 000000 000000 000000
001320 / 000000 000000 000000 000000 000000 000000 000000 000000
001340 / 000000 000000 000000 000000 000000 000000 000000 000000
001360 / 000000 000000 000000 000000 000000 000000 000000 000000

      00    02    04    06    10    12    14    16
001400 / 000000 000000 000000 000000 000000 000000 000000 000000
001420 / 000000 000000 000000 000000 000000 000000 000000 000000
001440 / 000000 000000 000000 000000 000000 000000 000000 000000
001460 / 000000 000000 000000 000000 000000 000000 000000 000000
001500 / 000000 000000 000000 000000 000000 000000 000000 000000
001520 / 000000 000000 000000 000000 000000 000000 000000 000000
001540 / 000000 000000 000000 000000 000000 000000 000000 000000
001560 / 000000 000000 000000 000000 000000 000000 000000 000000
001600 / 000000 000000 000000 000000 000000 000000 000000 000000
001620 / 000000 000000 000000 000000 000000 000000 000000 000000
001640 / 000000 000000 000000 000000 000000 000000 000000 000000
001660 / 000000 000000 000000 000000 000000 000000 000000 000000
001700 / 002322 000000 001726 004070 002016 000000 001722 100330
001720 / 174000 100330 174000 163046 002372 000071 000032 000170
001740 / 120230 000040 004020 121030 002372 002112 001774 001762
001760 / 120150 000000 121036 174000 101744 002112 000001 003770

      00    02    04    06    10    12    14    16
002000 / 004437 100212 002034 004012 001000 004070 002322 002120
002020 / 000002 000000 002122 000000 003762 000000 002066 003772
002040 / 003772 003762 000004 004014 000002 004020 002000 000000
002060 / 047506 020117 020040 117172 000012 144306 000040 003770
002100 / 144302 003770 101732 000001 121030 117172 077777 125630
002120 / 000000 000000 000000 000000 000000 141622 101550 101550
002140 / 101550 142064 101550 101550 101550 142252 101550 101550
002160 / 101550 142742 101550 101550 101550 141724 101550 101550
002200 / 101550 141724 101550 101550 101550 141622 101550 101550
002220 / 101550 141622 101550 101550 101550 101550 101550 101550
```



(continued on next page)

Figure 6–1: Post-Mortem Dump (Cont.)

```
007120 / 000000 000000 000000 000000 000000 000000 000000 000000
007140 / 000000 000000 000000 000000 000000 000000 000000 000000
007160 / 000000 000000 000000 000000 000000 000000 000000 000000
007200 / 000000 000000 000000 000000 000000 000000 000000 000000
007220 / 000000 000000 000000 000000 000000 000000 000000 000000
007240 / 000000 000000 000000 000000 000000 000000 000000 000000
007260 / 000000 000000 000000 000000 000000 000000 000000 000000
007300 / 000000 000000 000000 000000 000000 000000 000000 000000
007320 / 000000 000000 000000 000000 000000 000000 000000 000000
007340 / 000000 000000 000000 000000 000000 000000 000000 000000
007360 / 000000 000000 000000 000000 000000 000000 000000 000000

      00   02   04   06   10   12   14   16
007400 / 000000 000000 000000 000000 000000 000000 000000 000000
007420 / 000000 000000 000000 000000 000000 000000 000000 000000
007440 / 000000 000000 000000 000000 000000 000000 000000 000000
007460 / 000000 000000 000000 000000 000000 000000 000000 000000
007500 / 000000 000000 000000 000000 000000 000000 000000 000000
007520 / 000000 000000 000000 000000 000000 000000 000000 000000
007540 / 000000 000000 000000 000000 000000 000000 000000 000000
007560 / 000000 000000 000000 000000 000000 000000 000000 000000
007600 / 000000 000000 000000 000000 000000 000000 000000 000000
007620 / 000000 000000 000000 000000 000000 000000 000000 000000
007640 / 000000 000000 000000 000000 000000 002000 000000 000000
007660 / 000000 000000 000000 000000 000000 000000 000000 000000
007700 / 000000 000000 000000 000000 000000 000000 000000 000000
007720 / 000000 000000 000000 000000 000000 000000 000000 000000
007740 / 000000 000000 000000 000000 000000 000000 000000 000000
007760 / 000000 000000 000000 000000 000000 000000 002376 000000
```

6.13 Using System Spooling Services: The QUE Program

The QUE system program allows you to submit requests, or jobs, to be executed by spooling programs. QUE also executes auxiliary operations such as listing pending requests, killing pending requests, and modifying pending requests. The QUE program checks the syntax and validity of each request and passes it to another program for further processing if the request is valid. If any part of a request is invalid, the program rejects the entire request, prints an appropriate error message, and allows you to try again.

For the system to execute a request submitted to QUE, the system must contain three other programs: a queue manager program (QUEMAN), an operator services program (OPSER), and a spooling program. QUEMAN processes a request created by QUE and places it in the system queue file. OPSER communicates between the operator and the spooling programs. The particular spooling program executes requests on its related device when the QUEMAN program passes it a pending request. Valid spooling devices are as follows:

LP: Line Printer

BA: Batch

RJ: RJ2780

The spooling programs supplied on standard RSTS/E systems are SPOOL (LP:), BATCH (BA:), and (optionally) RJ2780 (RJ:).

6.13.1 Running QUE at a Terminal

A privileged or nonprivileged user runs QUE at a terminal logged into the RSTS/E system by typing the RUN $QUE command as follows:

```
RUN $QUE
QUE      V7      RSTS V7      TIMESHARING
```

When QUE runs, it prints a header containing the program name, its version and revision numbers, and the system name and version and revision numbers. The program ensures that the system queue file exists and is accessible on the system library account. If no errors occur, QUE prints a number sign (#) prompt to signal its readiness for a command. QUE also prints this number sign after processing each command.

If QUE runs and encounters an error when accessing the queue file, it prints the following message and reprompts:

```
?QUEUE NOT INITIALIZED
#
```

To run QUE without an error, have the system manager run the QUEMAN program and initialize the queue file.

You can terminate QUE by typing E or CTRL/Z in response to the # sign character, as follows:

```
#E
Ready
```

Control is returned to keyboard monitor command level, as indicated by the READY prompt (for the BASIC–PLUS environment). Commands to queue, list, kill, and modify jobs and the related error messages are explained in the following sections. A summary of the commands appears in Table 6–12.

Table 6–12: QUE Program Commands

Command	Format	Description
Que	Q device:jobname=file spec(s)	Allows you to give file specification(s) of a file to be processed. A comma is used to separate multiple file specifications. See Table 6–13 for options you can give with a file specification.
List	L device:=jobname(s)	Displays at the terminal a list of the currently pending requests for the spooling program. If device: is not given, LP0: is assumed. If device: is not given and jobname(s) is given, the = character is required. Valid devices are LP:, LP0:–LP7:, BA:, BA0:–BA7:, RJ:. Specifying LP: or BA: indicates all units of that device. If no jobname appears, QUE displays all jobs for the specified device.
Short	S device:=jobname(s)	Displays at the terminal a short form list of the currently pending requests for the spooling program. If device: is not specified, LP0: is assumed. If device: is not specified and jobname(s) is specified, the = character is required. Valid devices are LP:, LP0: – LP7:, BA:, BA0: – BA7:, RJ:. Specifying LP: or BA: indicates all units of that device. If no jobnames appear, QUE displays all jobs for the specified device.
Kill	K device:=jobname(s)	Removes from the queue the job(s) indicated by device specification and jobname. If device: is not given, QUE assumes LP0:. If device: is not given and jobname(s) is given, the = character is required.

(continued on next page)

Table 6–12: QUE Program Commands (Cont.)

Command	Format	Description
Modify	M device:jobname/options	Modifies the parameters of a job already in the queue. The following parameters can be modified by the M command: priority, forms name, after date or time, type of forms control, number of job copies, job status, and mode.
Flush	F dev: F BA:	Causes QUE to abort all current jobs in the queue. BA: specifies the BATCH Processor, dev: specifies a line printer unit (LP0: is the default). If you specify LP: with no unit number, all jobs in any device queue are aborted. The F command (FLUSH) is privileged and is available only to privileged users or to those included in the OPSER operator table.
Help	H	Causes an information message to be displayed at the terminal.
Exit	E	Causes QUE to terminate and to return control to system command level.
CTRL/Z	You type the CTRL/Z combination	Causes QUE to terminate (same as E).

6.13.2 Using the Q Command

The Q command typed in response to the # sign creates a request for a spooling program. The Q command has the following general format:

Q device:jobname=file spec, file spec,..., file spec

where:

device: designates the device and unit number (LP:, LP0:–LP7:, BA:, BA0:–BA7:, or RJ:) that the spooling program services. If a device: is not given, LP0: is assumed. LP: and BA: cause the request to be queued for any available spooling program for that device type. If a device is specified, the = sign is required.

If a unit number is specified, the request is queued for that unit and is executed only if a spooling program is running on that unit. If a unit number is not specified (LP: or BA:), the request is for the first available device. The general batch processor BA: processes only those jobs queued for BA: and not jobs queued for BA0: through BA7:. BATCH

processors BA0: through BA7: however, process requests queued for their respective units and requests queued for BA:.

jobname — is a maximum of six alphanumeric characters to identify your request in the queue. If device: and jobname are not specified, the = sign is optional and the file name of the first file specified in the request is the job name. Several options described in Table 6–13 may accompany the jobname.

NOTE

For privileged users, and for non-privileged users included in the OPSER program's operator table (if so allowed by the system manager), an account specified in the jobname parameter can be any valid account on the system. If not specified, the default is your current account. If specified, you assume the level of privilege of the account specified. Also, a file with the privileged protection code (128.) can only be queued by a privileged user; for such a file, the /DE switch (see Table 6–14) is not allowed.

file spec — is the file specification of the file to be processed from the queue and any combination of Q command options. Up to 11 files can be included in a request. File specifications are separated by a comma. (See Table 6–14 for the options.)

The jobname (on the left hand side of the = character) can have several output options which apply to the entire request. Each option must be preceded by a slash character (/). Options may be minimally abbreviated to their first two letters; for example, /MO, /AF, /PR, etc. Table 6–13 lists and describes the output options.

The /AFTER option, which can be specified with the jobname (and is briefly described in Table 6–13), causes the queue manager to initiate a requested job after a particular date and time. The /AFTER option arguments that specify the date and time take one or more of the following forms:

calendar date — in the form :dd-mmm-yy, where dd is the date, mmm is the first 3 letters of the month, and yy is the year. For

example, to initiate a job after midnight of September 22, 1981 (i.e., September 23):

/AFTER:22–SEP–81

If the year is not specified, the default is the current year.

numeric date — in the form :yy.mm.dd, where yy is the year, mm is the number of the month, and dd is the day. For example, to initiate a job after midnight of November 14, 1981:

/AF:81.11.14

relative date — in the form :+n D[AYS], where n is the number of days from the current date. For example, to initiate a job after midnight of September 5 (when the current date is September 3):

/AFTER:+2 D

military time — 24–hour time in the form :hh:mm, where hh is the hour and mm is the minute. For example, to initiate a job on December 12 of the current year after 1:30 pm:

/AFTER:12–DEC:13:30

AM/PM time — in the form :hh:mm am/pm, where hh is the hour, mm is the minute, and am or pm is used to designate before or after noon. For example, to initiate a job after 2:30 p.m. on October 15, 1981:

/AF:81.10.15:02:30 PM

relative time — in the form :+n H[OURS], where n is an integer in the range of 0 to 24. For example, to initiate a job 5 hours after the current time:

/AFTER:+5 H

The date and time specifications used with the /AFTER option are subject to the following rules:

1. The date specifications (calendar, numeric, and relative) when used without a time specification, cause the job to begin after midnight of the specified date.

2. Only one specification of each type is allowed. For instance, a relative date cannot be used with a calendar date nor can a relative time be used with military time.

3. When a relative time of 0 is used with a relative date specification, it causes the job to begin after the current time on the date specified. For example, if the following option is used at 3 p.m. on September 12:

   /AFTER:+ 2 DAYS:+0 HOURS

   the job is initiated after 3 p.m. on September 14.

Table 6–13: QUE Job Output Options

Option Format	Description
/MODE:n	The value n specifies the MODE which the spooling program will use in its OPEN statement for the line printer (see the *RSTS/E Programming Manual*). The following specific MODE options may be used instead of /MODE:n).
/LENGTH:nnn	The value nnn (1 to 127) sets form length in lines per page.
/CONVERT	Change character 0 (zero) to character O (oh).
/TRUNCATE	Truncate lines longer than unit's configured length.
/LPFORM	Enable software formatting.
/UPPERCASE	Translate lower- to uppercase characters.
/SKIP	Skip 6 lines at bottom of each form.
/AFTER:spec	The value in spec represents the date and time after which the queue manager will initiate the current request. The values in spec can be one or more of the following as described in Section 6.2: /AFTER: dd-mmm-yy, yy.mm.dd, +n D[AYS]; hh:mm, hh:mm:am/pm, +n H[OURS]
/PRIORITY:n	QUE sets the priority to the value n which can be between 0 and 255. If /PRIORITY:n is not specified, QUE assigns the value 128. A privileged user can specify a priority between 0 and 255. A nonprivileged user can specify a priority between 0 and 127. The priority setting determines the job's place in the queue; the higher the number, the higher the priority. If two or more jobs have the same priority, they go to the spooler in order of job requests.
/TYPE:xxx	Use the format specified by xxx for printing the file. Value may be: FTN FORTRAN forms control. EMB Embedded forms control. IMP Implied forms control (LF and CR printed before each record).

(continued on next page)

Table 6–13: QUE Job Output Options (Cont.)

Option Format	Description
/FORMS:<form name>	The string <form name> specifies the name of the form on which the job is printed; <form name> is up to 6 characters, of which the first character should not be a digit. The default form name is NORMAL.
/JCOPIES:n	The value n is the number of job copies to be printed. If /JCOPIES:n is not specified, one copy of the job is printed.

NOTE

The file name options /CO:nn, /NH, /DE, and /BI may also be specified as job options. (These are described in Table 6–14.) Any file name option so specified is applied to each file in the job.

The external file specification of a file to be included in the job can have the following form:

device:filename.type[proj,prog]/option/option...

A valid device is either a valid disk designator or null (which indicates the system device in the public structure). QUE interprets logical names if you make the assignment before QUE runs. Otherwise, an unassigned logical device name generates the ?ILLEGAL INPUT FILE error. See Section 4.3.6 for a description of logical names.

Note that when you are queueing a file to a line printer, the file cannot be larger than 32767 blocks.

The file name and type may consist of ? and * wildcard characters described in Section 6.3.1. A file specified with no file type is given the default type .LST if it is in a line printer queue, or the default file type .CTL if it is in a BATCH queue. The [proj,prog] designation is the project-programmer number (account) under which the file to be printed is stored. The designation can be omitted if the file is stored under your current account. Note that wildcard characters (? or *) cannot be used to designate project-programmer numbers. The /option designation is a slash character (/) followed by one of the Q command options. The options are described in Table 6–14.

Table 6–14: Q Command Options

Option	Format	Description
Copies	/CO:nn	Causes SPOOL to print the number of copies indicated by the decimal integer n. If /CO:n is not given, one copy is printed.
Noheader	/NH	Causes SPOOL to suppress printing of the file header.
Delete	/DE	Causes SPOOL to delete the file after it is printed if the protection code of the file permits. If /DE is not specified, the file is left intact. The QUEMAN program checks the protection code of the file. Note that /DE will not work for file specifications that include wildcards.
More	/MORE	Used only at the end of a command line to indicate to QUE that you have more text to type on the next physical line.
Binary	/BI	Used to indicate a binary file to the RJ2780 program, which must be in TRANSPARENT mode.
Restart	/RE	Causes SPOOL to restart the file or job if a malfunction occurs in printing.

The following command typed to QUE:

```
#Q ABC.DAT
#
```

causes the file ABC.DAT stored under your current account on the public structure to be sent to the queue manager for printing on the spooled line printer, unit 0. SPOOL generates one copy of the file under job header ABC.

To queue a file from a device other than the public disk, specify the device designator in the command. For example:

```
#Q DK1:A.BAS/CO:2
#
```

The indicated file on DK1: is queued for printing on line printer unit 0. SPOOL generates two copies of the file under the job header A.

To specify a job name for the request, type the name and the = character in the command. For example:

```
#Q SORT=DK1:FILEA,FILEB/CO:2
#
```

As a result, SPOOL prints SORT in the job header and generates one copy of DK1:FILEA.LST and two copies of DK1:FILEB.LST.

To specify a spooling program other than the one for line printer unit 0, type the appropriate designator and the = character in the command. For example:

```
#Q LP1:=DK1:FILOUT.001
#
```

As a result, QUE creates a request for FILOUT.001 on line printer unit 1, with job name FILOUT.

To specify a request longer than one physical line, type the /MORE option as the last item on the line. For example:

```
#Q LP1:PRINT1=FILE1.001,ABCDEF.001,/MORE
MORE > HELP.TXT,ACCT.SYS
#
```

As a result, QUE prints, as a prompting indicator, the text MORE>, after which you can continue typing the request. QUE allows up to eleven files in one request. Type /MORE as many times as necessary. Note that /MORE is invisible to the parser; the completed command is interpreted as if it were typed as one line. Therefore, all punctuation (commas, colons, etc.) must appear in the appropriate places.

When a device designator is specified in the command, QUE uses that device (and not the system device) as a default for subsequent input file names on the physical line. For example, when the following command is executed:

```
#Q LP1:SORT=DK2:FILEA,FILEB
```

QUE sets the device to DK2: for both FILEA.LST and FILEB.LST. Because no device designator is specified for FILEB, DK2: becomes the default device.

To set a priority to a job when queuing a file, specify the /PR:n option with a value between 0 and 255. Without the /PR option, QUE assigns a priority of 128 to the job. The queue management program processes jobs with a priority of 128 or greater before processing jobs less than 128. Only privileged users can specify a priority greater than 128. With such a priority scheme, the queue manager can process important jobs before routine jobs and can delay processing less important jobs until routine jobs are completed. Note that scheduling is a strict function of priority; if you submit a high-priority batch job that will run for a long time, it will delay other jobs submitted to the same spooler.

6.13.3 Using the L and S Commands

The L command lists, at your terminal, information about pending requests for a specified device. The following example illustrates the command and its results:

```
#L LP0:

LP0 QUEUE LISTING          19-Feb-81       04:12 PM
UNIT     JOB               S / P     FILES
 0       NOTICE[2,201]/SE:1038/FO:NORMAL
                           SK/128/TY:EMB
                                           SY :[1,2]NOTICE.TXT

 0       BNFTRN[2,201]/SE:1039/FO:NORMAL
                           0 /128/TY:EMB
                                           SY :[2,201]BNFTRN.SIF
                                           SY :[2,201]BNFBLD.SIF
                                           SY :[2,201]PERCNT.BAS

 0       VISITS[2,201]/SE:1040/FO:NORMAL
                           0 /128/TY:EMB
                                           SY :[2,201]VISITS.LST
```

NOTE

Because both QUE and QUEMAN can access a queue list simultaneously, QUE tests the linkage of the queue list to verify that the queue list has not been changed by QUEMAN during QUE's search. If QUE detects that the linkage has changed, QUE will display the following message and will restart the listing from the beginning of the queue list.

```
*****QUEUE CHANGING--WILL RESTART LISTING*****
```

The L command causes QUE to display two header lines for the specified spooled device. The first header line contains the device and unit designator of the specified device (LP0: by default), the current system date, and the time of day. The second header line contains headings that denote the unit queued to, the jobname and account number of the requester, and the status, priority, and full file specifications associated with each job. If no jobs are queued, QUE displays:

```
dev: QUEUE IS EMPTY
```

Jobs are displayed in the order in which QUEMAN accesses them. In the first line of each job description, the jobname is followed by its account number and the sequence number (/SE:n) assigned the job by QUEMAN. Following the sequence number is the /FO[RMS] option and its argument, the forms name: the default NORMAL or the name you specified.

In the second line of each job description, the status appears under the heading S, and is indicated by one of the following:

0 Job is in queue waiting to be processed.

S Job has been sent to spooler.

A Job is waiting for an /AFTER date/time to expire.

H Job is in hold status: it has been modified (by the M command) to be put into hold, or it was put into hold if, upon restarting, QUEMAN found that the job was previously sent to the spooler.

K Job is in kill status.

Some status indicators may appear together; for example, SK means that the job was sent to the spooler, and that a K command was later issued for it.

Next on the second line appears the priority (0 to 255), under the heading P; 128 is the default. Following the priority are any job options you specified.

Appearing under the lines describing a job are the descriptions of individual files in that job—the full file specifications followed by any file options you specified.

To list jobs on all line printers, type the L LP: command. QUE displays the related unit number in the UNIT column for each queued job. Display jobs queued for the BATCH processor by typing the L BA: command; display jobs queued for the RJ2780 program by typing the L RJ: command.

To list only information about one request or several requests, include the = character followed by the jobname(s) in the L command. The following example illustrates:

```
#L LP0:=REW[200,57]
LP0 QUEUE LISTING          09-Nov-81       03:22 PM
UNIT     JOB               S / P       FILES
 0       REW    [200,57]/SE:2775/FO:NORMAL
                           S /128/TY:EMB
                                       SY :[200,57]EXPENS.BAS
                                       SY :[200,57]VISITS.LST/C:3
```

QUE prints the header lines and the data for the jobname specified. If you specify more than one jobname, separate them by commas, as in the following example:

```
#L LP0:=REW[200,57],REF[200,57]
LP0 QUEUE LISTING        09-Nov-81       03:22 PM
UNIT     JOB             S / P       FILES
 0       REW    [200,57]/SE:2775/FO:NORMAL
                         A /128/TY:EMB
                                     SY :[200,57]EXPENS.BAS
                                     SY :[200,57]VISITS.LST/C:3

 0       REF    [200,57]/SE:2776/FO:NORMAL
                         S /128/TY:EMB
                                     SY :[200,57]CHOICE.BAS/C:3/N
                                     SY :[200,57]CURZ  .LST/C:7/D
```

The S command lists, in a shortened form at your terminal, information on pending requests for a specified device. The following example illustrates the command and its result:

```
$S LP0:                                          AM
LP0 SHORT QUEUE LISTING 02-Oct-81                11:25 AM
UNIT     JOB                                     S / P
 0       NEWV05[1,234]/SE:1374/FO:NORMAL         S /128
 0       RMSTST[2,211]/SE:1375/FO:NORMAL         0 /128
```

Unlike the L command, the S command causes QUE to print a "short queue listing" header line followed by a single line description of each job in queue. The single line description contains the following information:

- jobname
- job's account number
- sequence number (/SE:)
- /FORMS option argument (/FO:)
- job status
- job priority
- /AFTER time specified

6.13.4 Using the K Command

The K command removes a job or jobs from the queue for a specified device. To remove a job, you can specify the device designator and the = character followed by the jobname or jobnames separated by commas. For example

```
#K LP1:=PRINT1,BATCH1
#
```

As a result, each job for your account in the LP1: part of the queue with the name PRINT1 or BATCH1 is deleted if it is not currently being processed by the LP1: spooling program.

In the case of matching jobnames, specify the job to be removed from the queue by including the /SE:nnnnn option in the K command. The argument nnnnn represents the job's sequence number.

You can find out the sequence number for a job by using the L command, Section 6.12.3. For example:

```
#K LP1:=PRINT1/SE:1475
#
```

6.13.5 Using the M Command

The M command allows you to modify the parameters of a job that is already in the queue. By issuing M with appropriate options, you can modify the job parameters originally set in the Q command: priority, mode, date/time, number job copies, form, and type. If, however, the job has already been sent to a spooling program, QUEMAN will not allow any modifications.

To specify modifications to the job, type the device designator followed by the jobname and the desired modification option(s) in the following format:

```
M device:jobname/option(s)
```

Each option must be preceded by a slash (/). The following list contains job modification options that may be specified with the M command. Except where otherwise indicated (for example, with /HOLD and /UNHOLD), these options correspond in name, function, and format with those of the Q command. Their full descriptions are in Table 6–13.

/HOLD	Halt the job's advancement in the queue, and do not send it to a spooling program until the /UNHOLD option is specified.
/UNHOLD	Continue the job's advancement in the queue.
/SE:nnnnn	Modify only the job with sequence number nnnnn (assigned by QUEMAN; see Section 6.13.3). Used in case of matching jobnames.
/PRIORITY:nnn	(See Table 6–13 for explanations of this and the remaining options listed here.)

/MODE:nnnn or
/LENGTH:nnn /UPPERCASE
/CONVERT /SKIP
/TRUNCATE
/LPFORM

/AFTER:	dd-mmm-yy	hh:mm
	yy.mm.dd	hh:mm am/pm
	+n D[AYS]	+n H[OURS]

/TYPE:xxx

/FORMS:form name

/JCOPIES:n

6.13.6 Error Messages and Codes

QUE reports an error condition by displaying a message or by returning an error code. When running at a terminal, QUE displays a unique message which describes the error condition. No part of the typed command is executed. Retype the entire command in the correct format. Any RSTS/E system errors encountered are reported in the ?ILLEGAL INPUT FILE message. When running by means of a CHAIN operation from a BASIC–PLUS program, QUE reports an error by an error code when it passes control. Table 6–15 gives both the messages and codes and describes the related error conditions.

Table 6–15: QUE Error Messages and Codes

Message and Meaning	Error Code
?INVALID COMMAND-<text> The <text> indicated is an undefined command.	1
?SW ON WRONG SIDE OF COMMAND The switch specified in the command line to QUE is reserved for the input side and was found on the output side, or vice-versa.	2
?INVALID SWITCH FORMAT The switch contains an invalid request.	3
?UNDEFINED SWITCH The command line contains an undefined switch.	4
?TOO MANY FILES No more than eleven files can be given in the Q or K command.	5
?NULL FILE SPEC At least one file specification must be given in a Q or K command and none was found; or two commas occurred with no file specification between them.	6

(continued on next page)

Table 6–15: QUE Error Messages and Codes (Cont.)

Message and Meaning	Error Code
Not used. 7	
?ILLEGAL INPUT FILE- <text—-filespec> The request is invalid because the file indicated by the file specification caused the RSTS error described in the text. Retype the request.	8
?WILDCARDS NOT ALLOWED IN PPN An asterisk is not allowed in the project-programmer field.	9
?QUEUE FULL The queue file is temporarily full. Try again when the spooling program has processed some pending requests.	10
?BAD SWITCH VALUE-<text> There is an invalid or improper number in the option indicated by <text>. For example, you specified /MODE:A or, being nonprivileged, you specified /PR:129 or greater.	11
?'MORE' REQUESTED ON A CHAIN The /MORE option is not allowed when a program runs QUE by a CHAIN operation.	12
?NOT A QUEUABLE DEVICE You attempted to queue a request to a device which cannot handle queued requests. For example, Q LG:=JOB.	13
Not used.14	
?QUEUE NOT INITIALIZED The queue file does not exist in the system library account. A privileged user must run QUEMAN to initialize the queue.	15
?QUEMAN NOT RUNNING - CAN'T QUE OR KILL You have attempted to queue or kill a job and, because the queue manager is not running on the system, QUE generates an error. You can, however, list jobs when QUEMAN is not running.	16
?QUEUEING DISABLED QUEMAN is in the shutdown process.	17
?ILLEGAL SWITCHES FOR CMD Specified options are illegal for commands used; e.g., /HOLD and /UNHOLD are illegal for Q.	18

6.13.7 Running QUE by CCL Commands

You can run QUE by means of the Concise Command Language (CCL) command QUEUE, which may be minimally abbreviated to QU. The correct format is QUEUE/<command line> where <command line> is any valid QUE program command. If the QUE program does not find a / character immediately following the characters QUEUE, it defaults the command to a Q command. For example, the command:

```
QUE/Q JOB1 = ABC.DAT
```

has the same effect as the command:

```
QUE JOB1 = ABC.DAT
```

These commands are equivalent since both enter JOB1 in the queue for line printer 0. In either case, the QUE program runs, executes the command, and exits. An error in the command causes QUE to print the related error message on a subsequent line, and exit.

To include an output option when queuing a job, begin the QUE command with /Q or else give an explicit jobname in the QUE command. For example:

```
QUE/Q/MODE:2048=DK1:DISK.BAS
```

The / character preceding the Q character differentiates the Q command from the jobname Q.

To list or kill jobs, type QUE, followed by a slash (/) character and the appropriate command. For example:

```
QUE/L DISPLY[1,253]
LPO QUEUE LISTING     09-Nov-81      03:22 PM
UNIT     JOB          S / P       FILES
 0          DISPLY[1,253]/SE:558/FO:NORMAL
                      0 /128/TY:EMB
                                  SY :[1,253]DISPLY.V6B

Ready
```

QUE runs and executes the command indicated following the / character, after which control is returned to system command level.

6.14 Obtaining a Disk Quota Report: The QUOLST Program

The QUOLST system program allows you to determine what portion of your disk quota is currently occupied and the number of free blocks remaining on the system disk. QUOLST is run as follows:

```
RUN $QUOLST
```

Output from QUOLST includes your account number and information printed under the following headings:

Table 6–16: QUOLST Column Headings

Column Heading	Meaning
STR	STRucture, device being reported.
USED	number of used 256–word blocks under your account. If the system supports large files, the number of blocks can appear as > = 65535.
FREE	number of free blocks remaining in your account disk quota.
SYSTEM	number of free blocks remaining to the system on the structure indicated.

Output from QUOLST appears as follows:

```
RUN $QUOLST
QUOLST  V7.0 RSTS V7.0 Timesharing

User:   [200,57]
Str     Used            Free            System
SY:     60              4940            5452
```

In this example, you are logged into the system under account [200,57] and have used 60 blocks on the public disk structure with a quota of 5000 blocks (5000–60 = 4940 free blocks). There are 5452 free blocks on the public disk structure.

6.15 Changing the Default Keyboard Monitor: The SWITCH Program

Each job on the system can have a job keyboard monitor, different from the default.

The command:

```
RUN $SWITCH
```

initiates a dialogue which allows you to change your keyboard monitor. The dialogue to switch to RT11, for example, is as follows:

```
RUN $SWITCH
KEYBOARD MONITOR TO SWITCH TO? RT11
```

The system transfers control to the RT11 keyboard monitor. A carriage return typed in response to the above query causes the job to return to the default keyboard monitor.

The system manager can install the CCL command:

```
SW[ITCH]
```

to invoke the SWITCH program. In this case, type:

```
SW RT11
```

to change the current keyboard monitor to RT11. Typing the CCL command alone causes the system to revert to the default keyboard monitor.

SWITCH Program Error Messages

The following error messages are generated by the SWITCH program.

Message and Meaning

?ILLEGAL KEYBOARD MONITOR NAME
The keyboard monitor name is misspelled or incorrect. Repeat the request.

?NO KEYBOARD MONITOR
Either the keyboard monitor requested is not installed on this system, or you typed a run-time system name that does not have a keyboard monitor.

6.16 Printing a System Status Report: The SYSTAT Program

The SYSTAT program provides current system information in the areas of job, device, disk, buffer status, run-time systems, resident libraries, and message receivers.

To start SYSTAT, type:

```
RUN $SYSTAT
```

The program responds by displaying:

```
OUTPUT STATUS TO?
```

At this point, you can indicate any RSTS/E device or a file specification for the status report output. Possible replies are described below:

SYSTAT Output Response	Meaning
LP:	send status report to the line printer if only one line printer is on the system or to line printer unit 0 if multiple line printers are on the system.
LPn:	send status report to line printer unit n if that printer is not currently in use.
KB:	send status report to your terminal. (The RETURN key is equivalent to responding KB:.)
KBn:	send status report to terminal n in the system if that terminal is on-line and not currently in use.
dev:filename.type	send the status report to the file specified. The default device is the system device. No type is appended unless specified.
?	send status report to a file on the public structure, and print the file name. The file is named according to the current date and time of day; its type is .RPT. (The file name is explained in the following paragraph.)

An example of an output file created by the ? response to the OUTPUT TO query is F04N59.RPT. The file name has 4 parts. The first is a letter from A to L, and denotes the month according to its alphabetical position; here, F, the sixth letter of the alphabet, denotes June. The second part is two digits from 01 to 31 denoting the day of the month; here, the fourth. The third part is a letter from A to X denoting the hour from 00 to 23; here, N denotes the 14th hour (2:00 p.m.). The fourth part is two numbers denoting the minute of the hour.

Following the device or file name specification, you can specify one of the options in Table 6–17 to obtain a partial system status report.

Table 6–17: SYSTAT Options

SYSTAT Option Specification	Meaning
/A	Report only status of attached jobs.
/B	Report only busy device status.
/C	Report memory allocation on system (privileged).
/D	Report only disk status.
/F	Report only free buffer status.
/Kn	Report only job status of terminal n in the system.
/L	Report only resident library status.
/M	Report only message receiver status.
/N	Report only status of nonprivileged accounts.
/n	Report status of job n only.
/n,m	Report status of account [n,m] only.
/n,*	Report status of jobs with project number n only.
null	Report complete system status to include job, run-time system and resident library status, busy device, disk structure, free buffer status, and message receiver statistics.
/P	Report only status of privileged accounts.
/R	Report only run-time system status.
/S	Report only job status.
/U	Report only status of unattached (that is, detached) jobs.
/0,0	Report only status of jobs not logged into the system.
/O	Report all open non-system files (privileged).
/O:dev	Report all non-system files open on a specific device (privileged).
/W	Report all open non-system files, the jobs accessing them, and block number being accessed (privileged).
/W:dev	Report all open files, the jobs accessing them, and block number being accessed for a specific device (privileged users only).
A minus sign (–) may be included with any option, in any position following the slash, to cause printing of an account number instead of [OPR] and [SELF].	

The options S,A,B,D,F,N,M,L,O,C,W,R,P and U can be specified as separate options or in any combination. If multiple options are specified, only one slash is required. The options /O, /C, and /W can only be specified by a privileged user.

The following examples illustrate some typical SYSTAT options:

RUN $SYSTAT OUTPUT STATUS TO? STAT	creates complete system status report in the file STAT under the current account in the public structure.
RUN $SYSTAT OUTPUT STATUS TO? LP: /3	causes output of a status report for job 3 to the line printer.
RUN $SYSTAT OUTPUT STATUS TO? /D	causes output of disk status report to the user terminal.
RUN $SYSTAT OUTPUT STATUS TO? /SF	causes output of job and free buffer status to the user terminal.

6.16.1 Contents of the Status Report

A complete system status report is shown below:

```
RUN $SYSTAT
SYSTAT V7 RSTS V7

Output Status to? (RET)

RSTS V7 status at 28-Sep-81, 10:49 AM Up: 10:11:04

Job   Who    Where   What    Size  State   Run-Time   RTS
 1   [OPR]   Det     ERRCPY  5K    SR D56    12:17.4 BASIC
 2   [OPR]   Det     OPSRUN  16K   SL D62     5:43.1 BASIC
 3   [OPR]   Det     QUMRUN  16K   SL D58    32:26.8 BASIC
 4   [OPR]   Det     SPLIDL  16K   RN            4.1 BASIC
 5   [OPR]   Det     SPLIDL  16K   SL D61        0.4 BASIC
 6   [OPR]   Det     BATIDL  13K   SL D49        0.1 BASIC
 7   [OPR]   Det     BATIDL  13K   SL D48        0.0 BASIC
 8   [OPR]   Det     BATIDL  13K   SL D60        0.0 BASIC
10   [OPR]   Det     NPKDVR  8K    SR D63       14.4 BASIC
11   [OPR]   Det     DYN.2R  2K    SL D57    48:34.9 RSX
12   [OPR]   Det     V52DPY  16K   SL D52  5:01:48.9 BASIC
13   2,203   KB39    SPEED   8K    KB           3:58.6 BASIC2
14   5,224   KB44    QUE     14K   KB A02       6:15.3 BASIC
15   6,253   KB38    ERRDET  9K    RN A06       8:16.9 BASIC
16   2,251   KB32    ALARM   3K    SL D51       4:38.8 BASIC
17   2,225   KB30    VTEDIT  15K   KB A14       9:58.6 TECO
18   2,243   KB29    VTEDIT  15K   KB            14.7 TECO
19   [SELF]  KB68    SYSTAT  9K    RN Lck         2.4 BASIC
20   2,250   KB33    VTEDIT  15K   KB A15       4:02.4 TECO
21   2,241   KB45    QUE     14K   ^C A01        18.6 BASIC
22   2,200   KB22    NONAME  2K    ^C A11        18.5 BASIC
23   2,201   KB43    ...TKB  14K   RN           5:38.7 RSX
24   3,202   KB19    VTEDIT  16K   KB A03       5:16.1 TECO
25   4,246   KB18    NONAME  2K    ^C A12        14.3 BASIC
26   5,218   KB21    VTEDIT  19K   KB           1:08.1 TECO
```

(continued on next page)

```
27    2,243   KB40    LOGOUT   5K     RN A08      2:32.5 BASIC
28    2,244   KB41    VTEDIT   15K    RN A03      1:29.4 TECO
29    2,222   KB26    VTEDIT   15K    KB A13      3:11.3 TECO
30    2,242   KB31    SPEED    8K     KB A05        14.9 BASIC2
31    2,234   KB72    SPEED    8K     RN A03        53.9 BASIC2

Busy Devices:
Device  Job      Why
KB70    12       INIT
DMP-1   NSP      AS+OPEN

Disk Structure:
Dsk Open     Size       Free    Clu  Err  Name     Comments
DM0    0    27104    1472   5%    8    0  MYND     Pri, DLW
DR0    0   131648   15548  11%    4    0  R        Pri, DLW
DR1   51   131648    8000   6%    4    0  ARK      Pub, DLW
DR2    0   242572   12496   5%    4    0  H        Pri, R-O, DLW
DR3   11   500352   12432   2%    8    1  W        Pri, DLW
DR4    1   242572    5296   2%    4    0  M        Pri, DLW

General  FIP                        Hung
Buffers  Buffers  Jobs/Jobmax  TTYs  Errors
 302       108       28/50        0     121

Run-Time Systems:
Name  Typ  Size      Users  Comments
BASIC  BAC  16(16)K  19     Perm, Addr:41, DFKBM, CSZ
TECO   TEK  7(24)K   7      Temp, Addr:107
RT11   SAV  4(28)K   0      Non-Res, KBM, CSZ, EMT:255
RSX    TSK  3(28)K   2      Temp, Addr:253, KBM
RMS11  TSK  4(28)K   0      Non-Res
BP2COM TSK  4(28)K   0      Non-Res, KBM
BASIC2 TSK  16(16)K  3      Temp, Addr:214

Resident Libraries:
Name     Prot    Acct    Size   Users       Comments
RMSRES   <42>   [0,1]   23K      0      Non-Res;Addr:66

Message Receivers:
Rcvrid   Job     Rib    Obj    Msgs/Max    Links/Max   Access
ERRLOG     1       0      0       0/40         0/0      Prv
OPSER      2       0      0       0/30         0/0      Lcl
QUEMAN     3       0      0       0/60         0/0      Lcl
LPOSPL     4       0      0        0/5         0/0      Prv
BA0SPL     5       0      0        0/5         0/0      Prv
BA1SPL     6       0      0        0/5         0/0      Prv
EVTLSN     8       0     26       0/16         0/8      Net, Prv
EVTLOG     8       1      0       0/32         0/0      Evt, Prv
EVTTRN     8       2      0        0/5         0/0      None
NWPK09     9       0     23       0/16         1/8      Net
```

The job status information includes a list of all active jobs by job number, the account number under which each job runs, the keyboard involved (the keyboard number is followed by an asterisk if the job logged in by means of a dial-up line), the program name and size, the job state, the total amount of central processor run time used, and the run-time system. If a job is running with temporary privilege, a plus sign (+) is appended to the job

number; if a job has temporarily dropped its temporary privilege, a minus sign (–) is appended to the job number. In the WHO column, SYSTAT substitutes OPR for the project-programmer number to denote an operator account. An operator job has a project number of 1 and a programmer number between 1 and 200. In the WHERE column, DET appears in place of the keyboard number for jobs which run detached from a keyboard. Also, 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. The SIZE column shows the current size of the job. The STATE column contains an abbreviation (see Table 6–18) telling the current condition of the job. The RUN-TIME column gives hours, minutes, seconds, and tenths of seconds of central processor time the job has consumed. The RTS column gives the name of the run-time system under which the job is running.

The busy device status information reports 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 (in the WHY column). Assigned disk units are reported in the disk status information.

The disk status information describes each disk in use on the system. Items reported are: disk name (device specification), number of open files, total number of blocks on the disk, number of free 512–byte blocks, percentage of free blocks, pack cluster size, disk hardware error count, the pack identification or system logical name (if any) assigned for the device, and comments on its status. Disks assigned to jobs are reported as non-file structured and private and can include the job number and the designation "Dirty" (if appropriate). See Table 6–18 for abbreviations in the COMMENTS column.

The open files report lists the device, account, file name and file type, and protection code of each file opened on the system or specified device. The report also lists the number of jobs that have the file opened (or opened in read regardless mode), the cluster size of the file, and comments on the status of the file. Refer to Table 6–18 for an explanation of comments. Note that if the /W switch is used (see Table 6–19), the report also lists the jobs that are accessing the file as well as the block being accessed.

The buffer status provides the following information: 1) the number of small (16–word) buffers that are generally available and available from the "FIP pool" (an installation option), 2) the number of jobs currently running and the maximum number allowed to run (two numbers separated by a slash), 3) a count of the number of times a hung terminal was found. (A hung terminal is one that fails to respond to character transmission within a given time period) and 4) the total number of errors logged on the system.

The run-time system information gives the name of each run-time system, the default file type for (executable) files created by that run-time system,

the size of the run-time system in K words, the number of user jobs currently executing under its control, and comments regarding its status. See Table 6–18 for abbreviations in the COMMENTS column.

The resident library information gives the name of the resident library, its protection code and account number, its size, the number of users accessing that library, and comments concerning its status. See Table 6–18 for abbreviations in the COMMENTS column.

The message receiver report gives the receiving job's name and number, the number of Receiver ID blocks (Ribs), the object type code, the number of messages queued for the job and the declared maximum number of messages the job can have queued, and the number of logical links in use and the maximum number of links allowed. It also tells whether local and network senders are allowed, whether the job should handle only one link, and whether local senders must be privileged.

The abbreviations used in the SYSTAT report are defined in Table 6–18.

Table 6–18: SYSTAT Abbreviations

Abbreviation	Meaning
job status (states)	
DET	job is detached from all terminals.
,	job is not logged into the system.
[OPR]	job runs under a system operator account.
[SELF]	job runs under your account.
RN	job is running or waiting to run.
RS	job is waiting for residency.
BF	job is waiting for buffers (no space is available for I/O buffers).
SL	job is sleeping (SLEEP statement).
SR	job is sleeping and is a message receiver.
FP	job is waiting for file processing action by the system (opening or closing a file, file search).
TT	job is waiting to perform output to a terminal.
HB	job is detached and waiting to perform I/O to or from a terminal.
KB	job is waiting for input from a terminal.
^C	job is in CTRL/C state, awaiting input for the keyboard monitor.
CR	job is waiting for card reader input.
MT, MM, or MS	job is waiting for magnetic tape I/O.

(continued on next page)

Table 6–18: SYSTAT Abbreviations (Cont.)

Abbreviation	Meaning
LP	job is waiting to perform line printer output.
DT, or DD	job is waiting for DECtape I/O.
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.
DK,DM,DB,DS, DP,DL,DF,DR	job is waiting to perform disk I/O.
DX	job is waiting for flexible diskette I/O.
RJ	job is waiting for RJ2780 I/O.
??	job's state cannot be determined.
The following status descriptions may appear after one or more of the other job state abbreviations:	
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 by A,B,C, or D to represent files 0 through 3 of the swapping structure.
busy device status	
AS	device is explicitly assigned to a job.
OPEN	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.
disk status	
PUB	cartridge or pack is public.
PRI	cartridge or pack is private.
NFS	disk is open as non-file-structured device.

(continued on next page)

Table 6–18: SYSTAT Abbreviations (Cont.)

Abbreviation	Meaning
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 put at beginning of directory.
Job n	private disk is assigned to job n.
Dirty	disk needs cleaning.
run-time system and resident library status	
Non-Res	run-time system or library is nonresident.
Loading	run-time system or library is being loaded into memory.
Temp	run-time system or library will be removed from memory when not being used.
Perm	run-time system or library will stay in memory when not being used.
Addr:xxx	denotes the run-time system's or library's starting address.
DF KBM	run-time system is the default keyboard monitor.
KBM	run-time system can serve as keyboard monitor.
1US	run-time system or library can serve only 1 user.
R/W	run-time system or library allows read/write access.
NER	errors occurring within run-time system or library will not be sent to system error log.
Rem	run-time system or library will be 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.
message receiver status	
Local	local senders are allowed for this receiver ID.
Priv	local senders must be privileged to send to this receiver ID.
Oneshot	user has indicated that job should handle only one network link.
Network	network senders are allowed for this receiver ID.

Note that when a privileged user runs SYSTAT, the listing contains additional information as follows:

1. The SIZE column includes a number preceded by a slash that indicates the size to which the job can expand.
2. A column headed Pri/RB prints a set of numbers that indicate the job's priority and run burst.
3. If the /O or /W option is specified SYSTAT lists all open files on the system and the jobs accessing them. The list has the following form:

```
DS0: -- System files only
DS1: -- None

DR1: -- File                    Op/RR      Size     Clu    Status
DR1:[  1,248]SYSTAT.LOG<60>     1/O          1       4
       Job 23  Block 1          Rd, Wr

DR1:[  1,253]INIT  .LOG<60>     1/O          2       4
       Job 14  Block 2          Rd, Wr

DR1:[  1,243]SEND28.TMP<60>     1/O          0       4     Tent
       Job 28  Block 0          Rd, Wr, Tent
          .
          .
          .
```

Table 6–19 lists the abbreviations used in the report and their meanings.

Table 6–19: Abbreviations for /O and /W Report

Abbreviation	Meaning
open files	
Pla	file is placed.
Upd	file is open in UPDATE mode.
Tent	Tentative file.
MDL	file is marked for deletion.
Ctg	file is contiguous.
NoK	file can not be renamed or deleted.
UFD	file is a UFD–type entry.
None	no files are open on the disk.
System files only	only swap files or run-time systems are open on the disk.

(continued on next page)

Table 6–19: Abbreviations for /O and /W Report (Cont.)

Abbreviation	Meaning
jobs accessing open files	
Rd	user has read access to the file.
Wr	user has write access to the file.
Ca	file is open for user data caching.
Sq	file is open for sequential user data caching.
RR	file is open read regardless.
Tent	tentative file.
Up	file is open for UPDATE.
SpUp	file is open in special UPDATE mode.

4. If /C is specified, SYSTAT prints a table showing the allocation of memory on the current system. Because the table shows starting and ending addresses for currently installed run-time systems and resident libraries, the table is useful to privileged users who need to determine available addresses for the installation of new run-time systems or resident libraries.

   The table lists the following items:

   a. The system memory allocation as set by INIT.SYS (see the *RSTS/E System Generation Manual*). These entries include:

      Monitor (the resident portion)
      XBUF (extended buffer pool, if present)
      Locked out memory (if any)
      Non-existent memory (if any exists before the end of physical memory)

   b. Resident libraries.

   c. Run-time systems that were added at specific addresses or were loaded with the STAY attribute.

The table has the following form:

```
Memory allocation table:
 Start    End   Length   Permanent     Temporary
   0K -    46K (  47K) MONITOR
  47K -    62K (  16K) BAS2F  RTS
  63K -    68K (   6K) ** XBUF **
  69K -   100K (  32K)   (User)
 101K -   123K (  23K)   (User)      RMSRES LIB
 124K -   *** END ***
```

where:

Start and End	starting and ending addresses of that portion of memory.
Length	the length of that memory segment.
Permanent	those items allocated by INIT.SYS, and run-time systems and resident libraries that are permanently loaded. Memory that is not permanently allocated in one of these ways is labeled USER.
Temporary	run-time systems and resident libraries that were added at specific addresses but not permanently loaded.

Note that the table lists only those items assigned to specific addresses; it does not reflect the dynamic state of memory.

6.16.2 SYSTAT as a CCL Command

SYSTAT as a CCL command can contain a file specification. The following commands show the proper procedure.

```
SYS B
```

SYSTAT creates file B and writes to it a complete system status report. The file resides under your current account in the public structure.

```
SYS /B
```

SYSTAT prints a busy device status report at the terminal. To create a status report in a file, type the file specification with an option as follows.

```
SYS B/B
```

SYSTAT creates a busy device status report in file B.

6.17 Setting Terminal Characteristics: The TTYSET Program

The TTYSET system program establishes terminal characteristics for your terminal.

Run TTYSET as follows:

```
RUN $TTYSET
TTYSET V7 RSTS V7 TIMESHARING
TERMINAL CHARACTERISTICS PROGRAM
? XXXX
```

where xxxx is one of the TTYSET commands shown in Table 6–20. Enter the command with the RETURN or ESCAPE key. TTYSET again prints ? to accept additional commands. To return control to the keyboard monitor, type EXIT, CTRL/C, or CTRL/Z.

Table 6–20: RSTS/E TTYSET Commands

Command	Function
LIST	Lists the current characteristics of the terminal.
CTRL/R	Enables the CTRL/R retype facility (see Section 4.2.3).
NO CTRL/R	Disables the CTRL/R retype facility (see Section 4.2.3).
WIDTH n	Sets the width of the print line for this terminal to n, which can be between 1 and 254. As a result, the system automatically generates a carriage return/line feed sequence if n printing characters have been printed or echoed without a carriage return/line feed sequence and another printing character is to be transmitted.
TAB	Enables hardware tab control. System transmits <HT> characters without translation.
NO TAB	Disables hardware tab control. To move to the next tab stop, the system transmits the correct number of space characters instead of transmitting an <HT> character.
FORM	Enables hardware form feed and vertical tab control. System transmits form feed and vertical tab characters without translation.
NO FORM	Disables hardware form feed and vertical tab control. System transmits 4 line feed characters in place of a form feed or vertical tab character.
LC OUTPUT	System transmits the lowercase characters a through z and the } and ~ characters to the terminal without modification.
NO LC OUTPUT	System translates any lowercase character to its uppercase equivalent before transmitting it to the terminal.

(continued on next page)

Table 6–20: RSTS/E TTYSET Commands (Cont.)

Command	Function
XON	Special terminal hardware allows the computer to interrupt transmission of characters from the terminal by sending the terminal an <XOFF> character (value 19_{10}). Similarly, the computer instructs the terminal to resume transmission of characters by sending the terminal an <XON> character (value 17_{10}). The terminal hardware must respond to <XOFF> and <XON> characters by ceasing and resuming transmission, respectively. **The VT100 requires XON.**
NO XON	Terminal does not have hardware required for XON feature.
RESUME ANY CTRL/C	Defines the XON/XOFF processing. RESUME ANY enables typeout and echo when any character is typed after XOFF. RESUME CTRL/C enables typeout and echo only when <XON> or CTRL/C are typed after <XOFF>. **The VT100 requires RESUME CTRL/C.**
LOCAL ECHO	Terminal, or its acoustic coupler, echoes characters as they are generated locally. System does not echo characters received from such a terminal.
FULL DUPLEX	Characters generated are sent only to the computer. System therefore echoes each character received so that it will be displayed locally and translates certain characters to perform the proper action. For example, a <CR> character received is echoed as a carriage return and line feed character sequence.
SCOPE	Terminal uses a CRT display and has the following characteristics: a. Conforms to synchronization as described under the STALL command. b. System echoes a <DEL> character (RUBOUT) as backspace, space, and backspace sequence. c. System generates <NUL> characters as fill for timing the following operations: home, erase to end of screen, erase to end of line, direct cursor addressing, and line feed.
NO SCOPE	Terminal is not a CRT display device. The system echoes a <DEL> character (RUBOUT) by printing a backslash (\) character and the last character typed and removes the last character typed from the terminal input buffer. Subsequent <DEL> characters cause the next to last characters to be sequentially printed and removed from the terminal input buffer until a character other than <DEL> is received. As a result, the system echoes another \ character to delimit the erased characters and echoes the correct character.

(continued on next page)

Table 6–20: RSTS/E TTYSET Commands (Cont.)

Command	Function
LC INPUT	Terminal transmits the full ASCII character set and system does the following: a. Recognizes only the ESC character (value 27_{10}) as an escape character, b. Echoes and uses the } character (value 125_{10}) and ~ character (value 126_{10}) without translation, and c. Echoes and uses lowercase alphabetic characters without translation.
NO LC INPUT	System treats the ESC, }, and ~ characters (values 27_{10}, 125_{10}, and 126_{10} respectively) as escape characters and translates lowercase characters received to their uppercase equivalents.
NO FILL	System does not generate fill characters for any characters transmitted.
FILL n	Sets the fill factor to n for this terminal where n is between 0 and 6. As a result, the system generates a multiple of fill characters for each hardware control character it transmits. See Section 6.17.3 for a discussion of generalized fill characters.
SPEED n	Sets to n the rate at which the terminal's interface can accept or pass characters. The value for n can be any legal speed for the interface, unless restricted by the system manager. This command can be used only on lines whose interfaces can be programmed to handle more than one speed (e.g., DH11, DZ11).
SPLIT SPEED i/o	Sets to i the rate at which the terminal's interface passes input to the computer and set to o the rate at which the terminal's interface accepts output from the computer. The values i and o can be any legal speed for the interface, unless restricted by the system manager. This command can be used only on lines whose interfaces can be programmed to handle more than one speed (e.g., DH11). **Split speeds are not recommended for VT100.**
NO PARITY	System ignores the parity bit on characters it receives and treats the parity bit on characters it transmits to the terminal as if it were a data bit.
EVEN PARITY	System sends characters to the terminal with the parity bit properly set for even parity but ignores the parity bit on characters received.
ODD PARITY	System sends characters to the terminal with the parity bit properly set for odd parity but ignores the parity bit on characters received.

(continued on next page)

Table 6–20: RSTS/E TTYSET Commands (Cont.)

Command	Function
STALL	Terminal obeys the following synchronization standard: if the terminal sends an <XOFF> character (equivalent to the CTRL/S combination), the computer interrupts transmission until the terminal sends either an <XON> character (equivalent to the CTRL/Q combination) or a CTRL/C combination. If the system receives an <XON> character, it does not keep that <XON> character as data. If the system receives another character, it resumes transmission and keeps that character as data. No characters are lost. **The VT100 requires STALL.**
NO STALL	<XON> and <XOFF> characters sent by the terminal have no special meaning.
UP ARROW	System echoes a control and graphic character combination as the ^ or ↑ character (value 94_{10}) followed by the proper graphic. For example, CTRL/E prints a ^E or ↑E.
NO UP ARROW	System echoes control and graphic character combination as is.
PRINT filename	Sends the specified file to the terminal, in binary mode. This command's special use is in initializing a terminal for which special software settings must be made: setting 8 spaces to a TAB, for example. In such a case, 8 escape sequences would be placed in the file to be specified with PRINT.
ESC SEQ	System treats an ESC character (value 27_{10}) as an indication of the start of an incoming escape sequence. The character is not echoed, nor are the characters in the sequence. The processing of input escape sequences is described in the *RSTS/E Programming Manual.*
NO ESC SEQ	System treats an ESC character (value 27_{10}) as a line terminating character and echoes it as a $ character.
DELIMITER x	Makes the printable character x a private delimiter for use with GET statements. Character will not be echoed.
DELIMITER "x"	Makes the nonprintable character x (TAB, for example) a private delimiter for use with GET statements. This format also works for printable characters. Character will not be echoed.
DELIMITER	Disables the private delimiter. Private delimiter is also disabled by any macro (multiple characteristic) command.
ESC	System treats only ASCII 027_{10} code as ESCAPE (ESCAPE or ALTMODE key).
NO ESC	System treats ASCII 027_{10}, 125_{10}, 126_{10} as ESCAPE (ESCAPE or ALTMODE key).
EXIT	Terminates TTYSET and returns control to the keyboard monitor.
/RING	(Privileged only) Specifies characteristics of modem control terminals.

(continued on next page)

Table 6–20: RSTS/E TTYSET Commands (Cont.)

Command	Function
HELP	Prints a listing of single and macro commands.
BREAK	System treats a BREAK keystroke as a CTRL/C: ASCII 003_{10}.
NO BREAK	System treats a BREAK keystroke as a NUL: ASCII 000_{10}.
GAG	Disables system broadcast capability. That is, messages from UTSEND utility or SEND syscall will not be displayed at your terminal.
NOGAG	Enables system broadcast capability.

In addition to the commands described in Table 6–20, TTYSET allows you to set all the individual characteristics for a certain type device by executing one command called a macro command. Each macro command assigns a set of predefined characteristics, any of which can be altered in turn by proper use of an individual characteristic command. Table 6–21 describes the default values for each macro command.

Table 6–21: Default Single Characteristic Settings

Individual Characteristics	Macro Command								
	VT05	VT05B	VT50	VT50H	VT52	VT55	VT100	ASR33	ASR35
Width n	72	72	80	80	80	80	80	72	72
Tab/Notab	Tab	Tab	Tab	Tab	Tab	Tab	Tab	Notab	Tab
Form/Noform	Noform	Noform	Noform	Noform	Noform	Noform	Noform	Noform	Form
LC output/ No LC Output	LC Output	LC Output	LC Output	LC Output	LC Output	LC Output	LC Output	LC Output	LC Output
XON/NOXON	NOXON	NOXON	NOXON	NOXON	NOXON	NOXON	XON	XON	XON
Local echo Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex
Scope/ No Scope	Scope	Scope	Scope	Scope	Scope	Scope	Scope	No Scope	No Scope
LC Input/ No LC Input	No LC Input	No LC Input	No LC Input	No LC Input	LC Input	LC Input	LC Input	No LC Input	No LC Input
Fill n	0	3	0	0	0	0	0	0	1
Speed n Split Speed i/o 2741	-	-	-	-	-	-	-	110	110
No Parity Even Parity Odd Parity	No Parity	No Parity	No Parity	No Parity	No Parity	No Parity	No Parity	No Parity	No Parity
Stall/ No Stall	Stall	Stall	Stall	Stall	Stall	Stall	Stall	Stall	Stall
Up Arrow/ No Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow
ESC Seq/ No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq
ESC/No ESC	ESC	ESC	ESC	ESC	ESC	ESC	ESC	No ESC	No ESC
Delimiter 0	DEL	DEL	DEL	DEL	DEL	DEL	DEL	DEL	DEL
Resume CTRL/C Resume Any	-	-	-	-	-	-	CTRL/C	-	-
Break/ No Break	Break	Break	Break	Break	Break	Break	Break	Break	Break
Gag No Gag	No Gag	No Gag	No Gag	No Gag	No Gag	No Gag	No Gag	No Gag	No Gag

Table 6–21: Default Single Characteristics Settings (Cont.)

Individual Characteristics	Macro Command								
	LA30	LA36	LA120	LA180	KSR33	KSR35	2741	RT02	LA34/LA38
Width n	80	132	132	132	72	72	130	32	132
Tab/Notab	Notab	Notab	Tab	Notab	Notab	Tab	Tab	Notab	Notab
Form/Noform	Noform	Noform	Form	Form	Noform	Form	Noform	Noform	Noform
LC Output/ No LC Output	LC Output	LC Output	LC Output	LC Output	LC Output	LC Output	LC Output	LC Output	LC Output
XON/NOXON	NOXON	NOXON	NOXON	NOXON	NOXON	NOXON	NOXON	NOXON	NOXON
Local Echo Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex	Full Duplex
Scope/ No Scope	No Scope	No Scope	No Scope	No Scope	No Scope	No Scope	No Scope	No Scope	No Scope
LC Input/ No LC Input	No LC Input	LC Input	LC Input	LC Input	No LC Input	No LC Input	LC Input	No LC Input	LC Input
Fill n	0	0	0	0	0	1	2	1	0
Speed n Split Speed i/o 2741	-	-	-	-	110	110	2741	-	-
No Parity Even Parity Odd Parity	No Parity	No Parity	No Parity	No Parity	No Parity	No Parity	Odd Parity	Even Parity	No Parity
Stall No Stall	Stall	Stall	Stall	Stall	Stall	Stall	No Stall	No Stall	Stall
Up Arrow No Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	No Up Arrow	No Up Arrow	Up Arrow
ESC Seq/ No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq	No ESC Seq
ESC/No ESC	ESC	ESC	ESC	ESC	No ESC	No ESC	No ESC	ESC	ESC
Delimiter 0	DEL	DEL	DEL	DEL	DEL	DEL	DEL	DEL	DEL
Resume CTRL/C Resume Any	-	-	-	-	-	-	-	-	-
Break/ No Break	Break	Break	Break	Break	Break	Break	Break	Break	Break
Gag No Gag	No Gag	No Gag	No Gag	No Gag	No Gag	No Gag	No Gag	No Gag	No Gag

NOTE

1. None of the macro commands in Table 6–21 performs an automatic PRINT.

2. If no speed is given, the terminal speed is not changed from the current setting.

The TTYSET program checks commands before and during execution and reports errors by printing the messages described in Table 6–22. If any errors involve the terminal speed file, immediately notify the system manager or responsible system programmer.

If you wish to define nonstandard macros or override any of the terminal characteristics set by existing macros, your system manager can create a file and include the desired TTYSET macro. Note that this file is examined only if the specified macro has no match in TTYSET's predefined macros (that is, the macro names that you define must be unique).

Table 6–22: TTYSET Error Messages

Message and Meaning
?ILLEGAL PRIVATE DELIMITER – <string> Delimiter was not specified or has been disabled.
?ILLEGAL FILE TO PRINT No filename specified in PRINT command.
?TERMINAL OPEN ERROR – <text> The RSTS/E error denoted by <text> was encountered while executing the command.
?<string> IS ILLEGAL The keyboard number denoted by <string> is not between 0 and 63 or is not a valid number.
?ILLEGAL FILL FACTOR Fill factor specified is not between 0 and 6.
?ILLEGAL WIDTH Width specified is not between 1 and 254.
?COMMAND <string> ILLEGAL Command indicated by <string> is undefined.
?ILLEGAL SPEED Speed given is not one defined for the device in the $TTYSET.SPD file.
%WARNING – $TTYSET.SPD ERROR Program warns you of possible corruption in the terminal speed file and denotes the exact error by printing the COMMAND ERROR message. Consult your system manager.

Table 6–22: TTYSET Error Messages (Cont.)

Message and Meaning
?COMMAND ERROR – <text> The RSTS/E error denoted by <text> was encountered when executing the command.
?ERROR – <text> The RSTS/E error denoted by <text> was encountered when executing the system function to change terminal status.
%WARNING – CANNOT OPEN $TTYSET.SPD FILE Program could not access the terminal speed file and denotes the exact error by printing the COMMAND ERROR message. Consult your system manager.

6.17.1 ESCAPE, ALTMODE, and PREFIX Characters

The RSTS/E system translates certain ASCII characters in a special manner. Some terminals have the outmoded ASCII character keys ALTMODE (125_{10}) and PREFIX (126_{10}). More recently designed terminals incorporate the 1968 ASCII character set and include the following control characters:

```
ESCAPE =  27₁₀
     } =  125₁₀
     ~ =  126₁₀
```

The RSTS/E system interprets ASCII 027_{10} as a line terminating character and automatically translates any ASCII 125_{10} and ASCII 126_{10} characters input from a terminal into 27_{10}. The system thus treats 125_{10} and 126_{10} as line terminators.

If you have a terminal with the 1968 ASCII character set, you can make the system treat 125_{10} and 126_{10} characters as they are printed rather than as control characters. The TTYSET commands LC INPUT and NO LC INPUT alter internal parameters for a given terminal so that the system treats 125_{10} and 126_{10} as they are printed or translates them automatically to their uppercase equivalents.

6.17.2 Lowercase and Uppercase Characters

Many terminals can send and display both lowercase and uppercase characters. Such terminals can therefore display the echo returned by the system to give you an accurate visual representation of the character transmitted.

Terminals such as the VT05 alphanumeric display can send either lowercase or uppercase characters but can display only uppercase characters. Consequently, the echo response of such a terminal to a lowercase character is the uppercase counterpart. The terminal gives you no visual indication that the character transmitted was a lowercase character.

Terminals such as the ASR–33 and KSR–33 neither send nor receive lowercase characters. If such a terminal receives a lowercase character, it displays the corresponding uppercase character.

Normally, RSTS/E software translates all lowercase characters to their uppercase counterparts before processing them. To take advantage of different features of terminal hardware, you can include or omit lowercase translation depending upon whether the terminal produces lowercase output, lowercase input, or both. The LC OUTPUT and LC INPUT commands, respectively, omit translation of lowercase characters generated as output on and input from a terminal. NO LC OUTPUT causes lower to uppercase translation on output to the terminal; NO LC INPUT causes lower to uppercase translation on input from the terminal.

6.17.3 Generalized Fill Characters

The RSTS/E system automatically generates a variable number of <NUL> characters (decimal value 0) as fill characters after outputting certain control characters. The generation of these meaningless <NUL> characters allows the terminal sufficient time to complete the physical action initiated by the control character, thus permitting the terminal to synchronize itself properly for printing the next meaningful character. The values in Table 6–23 show the numbers of <NUL> characters generated for control characters at certain TTYSET characteristic settings.

NOTE

If the fill factor is 0, no fill characters are ever generated. The expressions in Table 6–23 do not apply for FILL 0.

The TTYSET command FILL n sets the fill factor to be used in Table 6–23; the command NO FILL or FILL 0 disables the automatic generation of fill characters.

In Table 6–23, the fill factor is used as an exponent of 2. When the fill factor is increased by 1, therefore, the number of fill characters generated is doubled. Most terminals require no fill characters at low baud rate settings. But at some baud rates (for example, 600 baud for VT05s), a terminal starts to require fill characters. As the baud rate increases, the number of fill characters required also increases. Because the common baud rates increase by doubling, increase the fill factor by 1 each time the baud rate is doubled. (For example, the appropriate fill factor for a VT05 at baud 600 is 1; at 1200, 2; at 2400, 3; etc.)

Table 6–23: Generalized Fill Characters

Control Character	Decimal Value	Scope	No Scope	Form	No Form	Tab	No Tab
CR	13	0	2741: POS/10+1 NOT 2741: $1 \times 2^{Fill-1}$	N/A	N/A	N/A	N/A
LF	10	$1 \times 2^{Fill-1}$	0	N/A	N/A	N/A	N/A
HT(tab)	9	N/A	N/A	N/A	N/A	$1 \times 2^{Fill-1}$	Spaces are sent.
VT(Vert. Tab)	11	$1 \times 2^{Fill-1}$	See FORM/NO FORM	$5 \times 2^{Fill-1}$	Do 4 LFs	N/A	N/A
FF	12	N/A	N/A	$9 \times 2^{Fill-1}$	Do 4 LFs	N/A	N/A
CTRL/N (direct cursor addressing) VT05)	14	$1 \times 2^{Fill-1}$	0	N/A	N/A	N/A	N/A

6.17.4 XON/XOFF Remote Reader Control

To operate a low-speed paper tape reader connected to the RSTS/E system by either a data set (dial up) or a communications line, two requirements must be fulfilled as follows.

a. The terminal must have the requisite hardware option for XON/XOFF remote reader control.

b. The XON/XOFF feature must be enabled for the given terminal.

You can selectively enable and disable remote reader control for a remote terminal by the TTYSET commands XON and NO XON.

For low speed readers connected to the system by a local line interface, none of these requirements is necessary.

Terminals with the ability to transmit characters to the RSTS/E system at a high rate of input require the XON characteristic, which allows temporary suspension of input when the system's buffers are full.

6.17.5 Output Parity Bit

The RSTS/E software always ignores the parity bit on characters received from a terminal and omits the parity bit on characters it transmits. Because some terminals not supplied by DIGITAL can receive even or odd parity characters, the software must be conditioned to send parity characters. The TTYSET commands EVEN PARITY, ODD PARITY, or NO PARITY condition the software to send the correct parity. The software ignores parity bits on characters input to the system.

6.17.6 Private Delimiters

TTYSET enables you to establish, on your keyboard, a special delimiter. Creating such a private delimiter is especially useful on a data entry terminal with a specialized keyboard: you can designate a large or conveniently located key as the delimiter key. Note, however, that a character designated as a private delimiter will not echo at the terminal. This rule applies to nonprintable as well as printable characters; <TAB>, for instance, will have no visible effect.

To make a private delimiter of a printable character, type the SET DELIMITER command followed by the character, either with or without enclosing quotation marks. The following examples illustrate the format:

```
SET DELIMITER A
```

or

```
SET DELIMITER "A"
```

Either of these commands makes the character A into a private delimiter.

To make a private delimiter of a nonprintable character, type the SET DELIMITER command followed by the nonprintable character enclosed in quotation marks. The following example makes the <TAB> character into a private delimiter:

```
SET DELIMITER "     "
```

To disable the private delimiter, type the SET DELIMITER command with no argument:

```
SET DELIMITER
```

The private delimiter is also disabled by any macro (multiple characteristic) command: for example, SET VT05, SET LA36, etc.

Limitations of Private Delimiters — As previously mentioned, a character designated as a private delimiter will not echo at the terminal. Thus, making a private delimiter of a common formatting character such as <TAB> may cause confusion in editing a program.

The SET DELIMITER command causes the existing ASCII code for the specified character to be overridden. Therefore, if the character normally has an expanded or alternate function, that function will be eliminated by the command. RUBOUT, for example, will neither backspace (on a display terminal) nor print backslashes (on a hard-copy terminal).

For BASIC programs, private delimiters work only in the GET statement—not in the INPUT or INPUT LINE statement.

6.17.7 SET as a CCL Command

Under a standard RSTS/E system, you can execute TTYSET by the correct Concise Command Language (CCL) command. To execute a CCL command, issue the proper CCL command followed by the desired TTYSET command(s). Multiple TTYSET commands on the same line are separated by semicolons (;). For example:

```
SET LA36;WIDTH 80
```

6.18 Mounting and Dismounting Magnetic Tapes and Private Disks: The UMOUNT Program

A nonprivileged or privileged user can execute the MOUNT or DISMOUNT commands of the UMOUNT system program to logically mount and dismount magnetic tapes, and private disk packs. MOUNT and DISMOUNT are CCL commands and must be installed on the system by the system manager. An attempt to run the UMOUNT program by any other means generates the message PLEASE USE THE 'MOUNT' OR 'DISMOUNT' COMMAND.

6.18.1 Logically Mounting a Disk Pack or Cartridge

To logically mount a private pack or cartridge on the system:

1. Ensure that the drive is available by running the SYSTAT program, then load the pack or cartridge in the available drive.
2. Ensure that the drive is write enabled.
3. Type the MOUNT command followed by the device designator, the appropriate identification label, and any option(s) desired.

The following example illustrates the format of the MOUNT command:

```
MOUNT DK1:MYPACK
```

This command logically associates RK05 drive unit 1 with the identification label MYPACK. (Disk options are described in Section 6.16.2.) The disk is now available for read and write access and in the UNLOCK state.

An attempt to write a file on the disk when the current account is nonexistent generates the ?DISK PACK IS PRIVATE error. Ask the system manager or responsible system programmer to create the current account on the disk.

If any error occurs in the device designator or in the identification label, the program prints the following error message:

```
?DECODE ERROR IN MOUNT - <text>
```

The notation <text> represents the UMOUNT or RSTS/E error encountered. On printing this message, the program terminates.

If any error occurs in mounting the disk on the system, the program prints the following error message:

```
?PROCESS ERROR IN MOUNT - <text>
```

The notation <text> represents the UMOUNT or RSTS/E error encountered. Typical errors include having the device protected against writing (?DEVICE HUNG OR WRITE LOCKED), specifying an incorrect identification label (?PACK IDS DON'T MATCH), and trying to mount a disk that is already mounted (?DISK IS ALREADY MOUNTED).

If the program encounters an error when attempting to unlock the disk to enable write access, it prints the following message:

```
?PROCESS ERROR IN UNLOCK - <text>
```

The notation <text> represents the UMOUNT or RSTS/E error encountered. The pack, however, is mounted and available for read access only. Report any error of this kind to the system manager or responsible system programmer.

6.18.2 MOUNT Options: Disk Pack or Cartridge

The options described in this section can be used with the MOUNT command when mounting a private disk. Options are summarized in Table 6–24.

If the disk to be mounted must be kept locked, append the /LOCK option to the MOUNT command, as in the following example:

```
MOUNT DR1:DATA1/LOCK
```

This command, UMOUNT, runs and logically associates the identification label DATA1 with RM02 unit 1. The disk pack is now mounted in the LOCK state. Only privileged users have read and write access to the disk pack.

If the disk to be mounted is to be read only, append the /READ ONLY option to the MOUNT command, as in the following example:

```
MOUNT DK1:MYPACK/RONLY
```

As a result of this command, the disk may be referred to as DK1: and MYPACK, and can be read but not written.

If a logical name other than the pack identification label is desired for the disk, append the option /LOGICAL:, followed by a logical device name, to the MOUNT command. The following example illustrates:

```
MOUNT DK1:MYPACK/LOGICAL:PACK3
```

As a result of this command, the disk may be referred to either as DK1: or PACK3.

If only the physical device specification is to be allowed for the mounted disk, append the option /NOLOGICAL to the MOUNT command, as in the following example:

```
MOUNT DK1:MYPACK/NOLOGICAL
```

As a result of this command, the disk may be referred to only by its physical specification, DK1:. No logical names are allowed.

The MOUNT command also has a privileged option that allows a qualified user to mount a public disk as private. The /PRIVATE option is appended to the MOUNT command, as follows:

```
MOUNT DK1:PUB001/PRIVATE
```

As a result of this command, the disk is mounted as a private disk, and is not used as part of the public structure. The disk may be referenced as PUB001 and as DK1:.

6.18.3 Mounting a Magnetic Tape

The MOUNT command can be used with a magnetic tape designator to assign a unit to the current job, to set default labeling format, and to set the tape's density and parity. The options described in this section are summarized in Table 6–24.

To mount a magnetic tape:

1. Ensure that the drive is available by running the SYSTAT program.
2. Physically mount the tape on the drive and ready the drive.
3. Type the MOUNT command.

To set the magnetic tape labeling format, append the option /DOS or /ANSI to the MOUNT command. If /ANSI is specified, the appropriate volume identification label should appear following the device designator. The following example shows the procedure:

```
MOUNT MT1:MYVOL/ANSI
```

As a result of this command, the identification label is checked and, if correct, tape unit 1 is assigned to the current job with ANSI labeling default. The /ANSI option sets the labeling default to ANSI standard format.

Under normal conditions, the system checks the tape identification label to ensure that it matches the /ANSI specification. To override this check and cause the system to mount the tape based only on the format specification, specify the /NOCHECK option before the format specification. For example:

```
MOUNT MT1:/NOCHECK/ANSI
```

This command causes the system to assign tape unit 1 to the current job with ANSI labeling format, regardless of the tape's current labeling format. This option is useful when you wish to initialize a new tape (see Section 6.11.6.21).

To set the tape's density and parity, the options /DENSITY:nnnn and /PARITY:nnnn should be appended to the MOUNT command, for example:

```
MOUNT MT1:/DOS/DENSITY:800/PARITY:ODD
```

As a result of this command, tape unit 1 is assigned to the current job with DOS/BATCH labeling format, 800 BPI density, and ODD parity. If the density or parity specified is illegal on the current tape drive or is not that found on the mounted tape, an error message is displayed. System-wide defaults for magnetic tape density, parity and label are set by the system manager during system initialization. If you wish the system to accept a tape with nondefault characteristics, you must ASSIGN the tape before setting those characteristics (with MOUNT or PIP). Note that when the tape is deassigned, the system reverts to the default characteristics set during system initialization.

If the magnetic tape is to be mounted read only, the /RONLY switch should be appended to the MOUNT command. For example:

```
MOUNT MT1:/DOS/RONLY
```

As a result of these commands, tape unit 1 is assigned to the current job with DOS/BATCH labeling format, the currently assigned default density and parity, and the read only option.

If the current job is privileged, the /JOB:n option can be used with the MOUNT command to reassign the unit to job number n. The following example shows the procedure:

```
MOUNT MT1:/JOB:5
```

As a result of this command, tape unit 1 is assigned to job 5 with the currently assigned default labeling format. If job 5 does not exist, an error message is printed. If an unassigned logical name is given for the magnetic tape unit in the MOUNT command, UMOUNT prints the error text ?ERROR IN MOUNT – ?ILLEGAL DEVICE, and exits.

If any error occurs while mounting a magnetic tape, UMOUNT prints:

```
?PROCESS ERROR IN MOUNT - <text>
```

The <text> will be the UMOUNT or RSTS/E error encountered.

When mounting ANSI labeled tapes, the volume ID should be specified. If neither the volume ID nor the /NOCHECK option is specified, UMOUNT prints the warning:

```
%VOLUME ID SHOULD BE SPECIFIED ON ANSI MAGTAPE MOUNT
```

and continues processing.

6.18.4 Logically Dismounting Disk Packs, Cartridges, and Magnetic Tapes

To logically dismount a private pack or cartridge:

1. Determine the number of OPEN files on the device by running SYSTAT. If non-zero, wait until all files are closed before proceeding. If zero, proceed. Note that a device, which contains files currently in a queue or batch stream, must remain mounted until the queue or batch is executed.
2. Type the DISMOUNT command followed by the device designator of the drive and the pack identification label. (The pack identification is only required if you are not privileged.)

The following example illustrates the format of the DISMOUNT command:

```
DISMOUNT DK1:PACKID
```

This command logically dismounts the pack on RK05 drive 1; the pack identification is removed from the system's table of logical names. The drive is now free for other usage and you can safely remove the pack or cartridge from the drive.

If UNMOUNT encounters an error when it attempts the dismount action, it displays a message in the following format:

```
?PROCESS ERROR IN DISMOUNT - <text>
```

The notation <text> represents the UMOUNT or RSTS/E error encountered. A typical error is attempting to dismount a disk that has open files (?ACCOUNT OR DEVICE IN USE).

To rewind a magnetic tape and take it off line, append the /UNLOAD option to the DISMOUNT command, as in the following example:

```
DISMOUNT MT0:/UNLOAD
```

As a result of this command, the tape on unit 0 is rewound and placed off line.

Table 6–24: The UMOUNT Options

Device	Option	Function
disk pack	/LOCK	Keep the mounted disk locked.
disk pack	/LOGICAL	Allow a logical name other than the pack ID for the mounted disk.
disk pack	/NOLOGICAL	Disallow logical name references to the mounted disk; allow only physical specification.
disk pack	/PRIVATE	Allow a public disk to be mounted as private (privileged only).
disk pack	/RONLY	Allow the mounted disk to be read only; prevent all write access.
magnetic tape	/ANSI	Set magnetic tape labeling default to ANSI format.
magnetic tape	/DOS	Set magnetic tape labeling default to DOS format.
magnetic tape	/NOCHECK	Override system check and set magnetic tape labeling default as specified (/ANSI or /DOS).
magnetic tape	/JOB:n	Reassign magnetic tape unit to job number n (current job must be privileged).
magnetic tape	/DENSITY:800	Set density to 800 BPI.
magnetic tape	/DENSITY:1600	Set density to 1600 BPI.
magnetic tape	/PARITY:ODD	Set to odd parity.
magnetic tape	/PARITY:EVEN	Set to even parity.
magnetic tape	/UNLOAD	(With DISMOUNT command) Rewind a magnetic tape and take it off line.

Table 6–25: UMOUNT Error Messages

Message and Meaning
?ILLEGAL LOGICAL NAME - <name> <name> is illegal because it is not a system logical name.
?ILLEGAL OPERAND <operand> <operand> is not one of the legal values.
?1600 BPI ILLEGAL Parity has been specified for this volume; 1600 BPI density is therefore illegal.
?ILLEGAL PARITY SWITCH Density of 1600 BPI has been specified for this volume; parity specification is therefore illegal.
?DISK PACK IS PUBLIC UMOUNT is used for private disks only.
?DISK IS ALREADY MOUNTED Cannot logically mount a disk that is already logically mounted.
?DISK DOES NOT HAVE RSTS FILE STRUCTURE UMOUNT processes RSTS/E file structured disks only.
?PACK ID MUST BE SPECIFIED To dismount a disk, a nonprivileged user must specify the disk's pack ID.
?DENSITY/PARITY SETTING CANNOT BE SET ON THIS DRIVE Cannot use a magnetic tape of the specified density and/or parity on this drive.
?INVALID ANSI MAGTAPE LABEL The value of the volume label identifier and label number is not VOL1.
?VOLUME ID'S DON'T MATCH When mounting an ANSI magnetic tape, volume ID of tape did not match with the ID specified (/NOCHECK was not specified).
?TAPE IS NOT WRITE LOCKED When magnetic tape was mounted, /RONLY was specified, but tape is physically write enabled.
?TAPE IS WRITE LOCKED The magnetic tape must be mounted write-enabled when /RONLY is not specified in mount command.
?ILLEGAL DEVICE Device specified is not disk or magnetic tape.
?ILLEGAL COMMAND Command unrecognizable or incomplete.

(continued on next page)

Table 6–25: UMOUNT Error Messages (Cont.)

Message and Meaning
?NO DISK DEVICE UNIT NUMBER SPECIFIED Disk device unit number must be specified.
?SYNTAX ERROR Illegal field(s) in command string.