RSTS/E
Utilities Reference Manual

Order No. AA-EZ11A-TC

June 1985

This manual describes the functions and uses of the RSTS/E system utility programs.

OPERATING SYSTEM AND VERSION:	RSTS/E	V9.0
SOFTWARE VERSION:	RSTS/E	V9.0

digital equipment corporation, maynard, massachusetts

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.

CONTENTS

TABLES

Preface

Objectives

This manual describes RSTS/E system utility programs. Taken together, the utilities comprise a set of general purpose programs that can assist you in a variety of ways ranging from file transfer to directory listing operations. This manual describes the purpose and use of each utility in detail.

Audience

This *RSTS/E Utilities Reference Manual* is for anyone who uses the RSTS/E operating system. The utilities that return information to the user, such as DIRECT and HELP, are particularly useful to inexperienced users. On the other hand, file transfer utilities such as PIP can greatly assist the experienced RSTS/E user.

Document Structure

This manual has two chapters and one appendix:

Chapter 1	Introduces the system utilities and summarizes their use.
Chapter 2	Describes each system utility in detail.
Appendix A	Describes the OPSER-Based Batch Processor control statements.

Related Documents

This manual references the following manuals where appropriate:

The *RSTS/E Task Builder Reference Manual* tells how to use the Task Builder to link compiled or assembled programs and subprograms into an executable program file to run on RSTS/E:

The *RSTS/E Programmer's Utilities Manual* describes the RSX-based utility programs available to RSTS/E users.

The *RSTS/E RT11 Utilities Manual* describes the RT11-based utilities that MACRO and FORTRAN IV programmers need to develop programs on RSTS/E and RT-11 systems.

The *RSTS/E System User's Guide* explains the use of the RSTS/E operating system and the DIGITAL Command Language (DCL).

Conventions

This manual uses the following conventions:

<RET>	Represents the RETURN key on your terminal.
<LF>	Represents the LINE FEED key on your terminal.
CTRL/x	Indicates a control key combination, such as CTRL/Z or CTRL/W. To enter a control key combination, hold the CTRL key down while you press the indicated key.
^	The circumflex preceding an uppercase character indicates a control character. For example, ^Z indicates the combination of the CTRL key and the Z character and can also be expressed as CTRL/Z. All control key combinations echo on your terminal screen as ^x. For example, if you hold the CTRL key and press the Y key, your screen echoes ^Y.
color	Red print used in examples indicates characters that you type.
UPPERCASE	In statement descriptions, you must type items that appear in capital letters exactly as shown.
[]	Square brackets indicate an optional statement element. For example: /W[ATCH] The required part of the switch is /W.

Most examples in this manual do not show the RETURN key <RET> symbol. If you try examples at your terminal, always press the RETURN key when you finish typing a command unless the example indicates otherwise.

Terms

This manual uses the following terms:

System	This refers to the RSTS/E operating system.
Print and Type	The system prints and you type.
Command	An instruction that directs a system program to perform an operation.
Program	The series of commands that make up the system utility.
Utility	One of the general purpose programs that this manual describes.

Summary of Technical Changes

The following changes have been made to the utilities for V9.0:

- MONEY, QUOLST, SWITCH, TTYSET, and UMOUNT have been deleted. The functions of these utilities have been replaced by DCL commands. See the *RSTS/E System User's Guide* and the *RSTS/E System Manager's Guide* for more information.

- ODT, RESTOR, and TALK have been moved from the *RSTS/E System Manager's Guide* to this manual. RESTOR retains the RESTORE functions previously included in the SAVE/RESTORE system program.

- LOGIN and LOGOUT have been moved to the *RSTS/E System User's Guide*.

- DCL commands duplicate some or all of the functions performed by many of the utility programs documented in this manual. See the *RSTS/E System User's Guide* and the *RSTS/E System Manager's Guide* for more information.

Chapter 1

Introduction

This manual describes utility programs that are available on RSTS/E systems.

Utilities are general purpose programs that DIGITAL supplies to the RSTS/E user. You can run all of the utilities that this manual describes from any command environment that RSTS/E supports.

Chapter 2 describes the RSTS/E utilities, which are arranged in alphabetical order. Appendix A describes the OPSER-Based Batch Processor command statements.

Table 1-1 gives an overview of the utilities, the functions they perform, and their location in the manual.

Table 1-1: Overview of System Utility Programs

Utility	Function	Page Number
BATCH	OPSER-Based Batch Processor control statements	A-1
COPY	Copying between devices	2-1
DIRECT	Listing a directory	2-7
FILCOM	Comparing contents of files	2-25
FIT	Transferring files between devices	2-40
FLINT	Transferring between IBM flexible diskettes and RSTS files	2-52
GRIPE	Sending a message to the system manager	2-68

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

Utility	Function	Page Number
HELP	Obtaining information about system programs	2-73
ODT	Manipulating address space locations	2-79
PIP	Transferring and maintaining RSTS/E files	2-97
PMDUMP	Formatting a post-mortem dump	2-138
QUE	Submitting requests to a spooler	2-149
RESTOR	Restoring pre-V9.0 backed-up files	2-170
SYSTAT	Obtaining a system status report	2-216
TALK	Communicating with other terminals	2-235

Gaining Access to Files and Accounts

Many RSTS/E utilities require you have read, write, or execute access to files or be able to create/rename files. RSTS/E controls access to files and accounts based on the file protection code and your privileges. The file protection code specifies read and write access for the file's owner, for other users within the owner's group, and for all other users on the system. For executable files, the protection code specifies execute and Read/Write access for owner, group, and all other users. Your privileges can override the file protection code in some cases.

Read and Execute Access

Depending on the file protection code, you can have read access to your own files, files in your group, or files not in your group. Regardless of the protection code, you have read access to your own files and files in your group if you have GREAD privilege, and read access to all files if you have WREAD privilege. These same conditions also control execute access.

Write Access

Depending on the file protection code, you can have write access to your own files, files in your group, or files not in your group (except those in [0,*]). Regardless of the protection code, you have write access to your own files and files in your group if you have GWRITE privilege, and write access to all files (except those in [0,*]) if you have WWRITE privilege.

You have write access to files in [0,*] if the file protection code allows world write access and you have SYSIO privilege. Regardless of the file protection code, you have write access to these files if you have WWRITE and SYSIO privileges.

Create/Rename Access

The ability to create new files and to rename existing files does not depend on the file protection code but does depend on the account in which you want to operate and your write privileges. You can always create and rename files in your own account. You can create and rename files in another account if you are in the same group and you have GWRITE privilege. You can create and rename files in any account (except [0,*]) if you have WWRITE privilege.

You can create and rename files in [0,*] if you have WWRITE and SYSIO privileges.

Using CCL Commands to Run System Utility Programs

The Concise Command Language (CCL) lets you type a user-defined command name to run a utility program instead of the traditional RUN command.

The advantages of using a CCL command is that it:

- Invokes the program using a shorter command name. For example, you can define SY to be the CCL command for the SYSTAT utility.
- Lets you bypass program dialogue by including all commands on a single command line. For example, if SW is the CCL command for the SWITCH utility, you can switch to the BASIC-PLUS keyboard monitor by typing SW BASIC.
- Returns you to the system keyboard monitor after running the program.

Your system manager uses the DIGITAL Command Language (DCL) command DEFINE/COMMAND/SYSTEM to define CCL commands. The definition associates a CCL command name (usually an abbreviation or a truncated version of the utility program name) with the utility program executable image. A CCL command name can be any name that complies with the naming rules outlined in the *RSTS/E System Manager's Guide*. To find out what CCL commands are defined on your system, check with your system manager or type the DCL command SHOW COMMAND/SYSTEM/ALL.

Table 1-2 lists some typical CCL commands and describes their function.

Table 1-2: Typical CCL Commands

Utility	CCL Command	Function
DIRECT	DIR	Displays a standard directory listing of your account
	DIR filespec	Displays a directory listing of the file you specify
	DIR/switch(es)	Displays a directory listing according to the switches you specify
FLINT	FLI	Runs the FLINT program
HELP	HE topic	Lists information on system programs and resources (use, command syntax, etc.)
PIP	PIP	Runs the PIP program
	PIP [command line]	Executes [command line] as a standard PIP command
QUE	QUE	Runs the QUE program
	QUE/[command line]	Executes [command line] as a standard QUE command
SYSTAT	SY	Displays a standard system status report
	SY/switch(es)	Displays a system report according to the switches you specify

Using CCL Commands from the DCL Command Environment

RSTS/E lets you type both DCL and CCL commands. CCL commands have been available on RSTS for many years and are specific to RSTS/E. DCL is more recent and is now the default keyboard monitor on RSTS/E systems.

DCL lets you truncate DCL command keywords. You can always truncate DCL keywords to the four leftmost characters, and even to two or three characters, as long as the remaining characters are unique within DCL. For example, you can type DIR or DIRE for the DIRECTORY command.

Some RSTS/E utility program names are the same as (or truncated versions of) DCL keywords (for example, COPY, HELP, and DIRECT). Therefore, you can define CCL command names that DCL accepts as DCL keywords because DCL takes precedence over CCL when operating from the DCL environment. To avoid this conflict, you can:

- Select CCL command names that are not the same as (or truncated versions of) DCL keywords. For example, define HLP as the CCL command name for the HELP utility.
- Add the prefix CCL to the CCL command name. For example, if DIR is the CCL command name for the DIRECT utility, type CCL DIR to run the DIRECT utility. You can always use the CCL prefix with any CCL command from any command environment.
- You can always run the utility by using the RUN command. For example, type RUN AUXLIB$:COPY to run the COPY utility.

Using CCL Commands from a Non-DCL Command Environment

In non-DCL environments there is no conflict between CCL and DCL command names because you must prefix all DCL commands with the dollar sign ($) character. For example, if DIR is the CCL command name for the DIRECT utility, type DIR to run the utility. To use DIR as a DCL command, type $DIR. If DIR is not a CCL command and you type DIR, you receive an error message.

You may have to consider precedence rules to avoid conflicts between CCL command names and keyboard monitor commands. See the *RSTS/E Programming Manual* for more information.

Chapter 2

System Utilities

Copying Between Like Devices: The COPY Program

Use the COPY program to copy all of the information on a device to another device of the same type. You can also use COPY to verify that the information on two like devices is identical.

Note

> Do not confuse the RSTS/E COPY utility with the DCL COPY command. You can use the DCL COPY command to copy individual files or groups of files, and to merge multiple files into a single file. See the *RSTS/E System User's Guide* for more information about the DCL COPY command.

You can use COPY for the following devices: DECtapes, magnetic tapes, and all disks supported by RSTS/E. DIGITAL recommends using COPY for small disks.

The COPY program has the following limitations:

- Use COPY only for magnetic tape to magnetic tape transfer, DECtape to DECtape transfer, or disk to disk transfer. You cannot use COPY to transfer information between dissimilar devices.
- COPY transfers all of the information on a device; you cannot specify individual files.
- COPY does not detect bad blocks on the new (output) device. Therefore, DIGITAL does not recommend using COPY to back up data on devices that allow bad blocks (DB, DM, and the DR series of disks, for example).
- You cannot use COPY to transfer a bootable magnetic tape between tapes of different densities. The bootstrap block on the tape works in only one density.

Privilege Required

To run the COPY program, you need the following:

- o When copying tapes, you need DEVICE privilege if either or both drives is a restricted device.

- o When copying disks, you need RDNFS privilege for the input disk and WRTNFS privilege for the output disk.

Running the COPY Program

To run COPY, type:

```
$ RUN AUXLIB$:COPY
```

COPY displays a header, followed by a number sign (#) prompt, which indicates that the program is ready to receive instructions. For example:

```
COPY  V9.0  RSTS  V9.0
#
```

To copy a device, respond to the prompt using the following command format. Note that you must include the fast copy switch (/FC):

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

For example:

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

In this example, the program copies all of the data, programs and directories block by block, from DECtape unit DT2 to DECtape unit DT1. When the process is complete, the DECtape on DT1 is an exact duplicate of the DECtape on DT2, while the original information on unit 2 remains intact. The next section describes the /FC switch in detail.

After COPY successfully transfers the information from one device to another, the program again prompts for input.

COPY Switches

You can specify one or more switches in the COPY command line.

Table 2-1 lists the COPY switches and briefly describes their meaning.

Table 2-1: COPY Switches

Switch	Format	Meaning
Blocksize	/BL:n	Specifies the blocksize used when copying.
Density	/DE:n	Specifies the density setting when copying magnetic tapes.
Fast copy	/FC	Directs the program to copy the image on the input device to the output device.
Help	/HE	Displays a help message on your terminal.
No copy	/NC	Used in conjunction with /VE, directs the program not to copy the information, but only to verify that the information on two devices is identical.
Parity	/PA:n	Specifies the parity setting when copying magnetic tapes.
Verify	/VE	Used in conjunction with /FC or /NC, directs the program to verify that the information on two devices is identical.

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

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

This command directs COPY to transfer the contents of the disk on drive DK1 to the disk on drive DK0, using 2048-byte blocks, rather than the default of 512-byte blocks.

/DE (density),/PA (parity)
The /DENSITY:d and /PARITY:p switches specify nondefault density and parity settings when copying magnetic tapes. You can abbreviate these switches to /DE:d and /PA:p. In these switches:

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

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.

The following example shows the use of the /DE:d and /PA:p switches:

```
#MT0:/DE:800/PA:EVEN=MT1:/FC
```

In this example, COPY writes the information on tape unit MT1 to tape unit MT0, at 800 bpi, even parity.

/FC (fast copy)
The /FC switch directs the COPY program to transfer the image on the input device to the output device. The information on the input device remains unchanged. You must include the /FC switch in the command line when you want to copy information.

/HE (help)
Type /HE in response to COPY's number sign (#) prompt to display a help message on your terminal.

/VE (verify), /NC(no copy)
The /VE switch directs the program to confirm that the information on two devices is identical. You can use the /VE switch in two ways:

- o When you use the /VE and the /FC (fast copy) switches together, the program first copies the information and then verifies that the copied information is correct.
- o When you use the /VE and /NC (no copy) switches together, the program verifies only that the information on the two specified devices is identical.

The following two examples show the use of the /VE switch:

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

In this example, the program first copies the information from DECtape unit DT2 to DECtape unit DT1. Then the program verifies that the copied information is correct. Because COPY performs the verification block by block, verification requires as much time as copying.

If you only want to confirm that the information on two devices is identical, use the /NC and /VE switches. For example:

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

In this example, COPY does not copy information; the program only verifies that the information on the two DECtapes is identical.

The two sample commands are the only allowable forms for verifying. As with the /FC switch, you cannot specify individual files with the /VE switch; it verifies all of the information on a device.

When COPY begins verification, the program displays the following message:

```
BEGINNING VERIFICATION PASS
```

If the program copies the information correctly, the program displays the following message:

```
VERIFICATION COMPLETE 0 BAD BLOCKS
```

However, if the program does not copy the information correctly, COPY displays the number of blocks (in decimal) 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
```

COPY Error Messages

Table 2-2 lists the COPY error messages and their meanings.

Table 2-2: COPY Error Messages

Message and Meaning
?CANNOT SPECIFY BOTH /FC AND /NC You specified both the /FC and /NC switches in the same command line.
?ERROR IN SPECIFYING DENSITY (or PARITY) You used a density (or a parity) that COPY does not support.
?ERROR IN SPECIFYING SWITCHES You misplaced or mistyped a switch.
?/FC OR /NC MUST BE SPECIFIED You failed to specify the /FC switch in the command line for a copy procedure, or you failed to specify either the /FC or /NC switch in the command line for a verification procedure.
?ILLEGAL BLOCK SIZE You specified an illegal blocksize.
?ILLEGAL DEVICE You specified an illegal or nonexistent device.
?ILLEGAL INPUT (or OUTPUT) SPECIFICATION You incorrectly specified the device in the input (or output) specification.
?MUST HAVE DIFFERENT UNIT NUMBERS You specified the same device unit number twice.
?MUST HAVE SAME TYPE DEVICES You attempted to copy information from one type of device to a different type of device.
?NO HELP FILE EXISTS The file COPY.HLP does not exist or resides in the wrong account.
?SYNTAX ERROR IN COMMAND STRING You typed a command string that COPY cannot interpret.

Listing File Directories: The DIRECT Program

The DIRECT program lists information, stored in a disk directory, about files. DIRECT opens the directory as a file and reads the information by immediately accessing the blocks in the directory.

Privilege Required

To run the DIRECT program, you need the following:

- o Create/Rename access to any output file you specify
- o Read access to any file you display from accounts other than your own

Running the DIRECT Program

To run DIRECT, type:

```
$ RUN $DIRECT
```

DIRECT prints a header line followed by a number sign (#) prompt:

```
DIRECT V9.0 RSTS V9.0
#
```

In response to the prompt, type a command line that has the format:

```
[output=][input[/sw]...][,input[/sw]...],...
```

where:

output = specifies the destination of the directory information. You can specify a device specification or a disk file specification. If you omit the file type in the output file specification, DIRECT assigns the default file type .DIR unless you force a null file type by including a period (.) after the file name. If you omit the output file specification, DIRECT displays the output on your terminal. If you do not specify output, also omit the = symbol.

input specifies the disk files you want listed. You can specify any number of full disk file specifications. The file specification can include a device, file name, file type, and project/programmer-number (PPN) If you do not specify a device, DIRECT assumes the public structure and denotes it by displaying SY: in the directory listing. If you do not specify any input specification, DIRECT lists information about all files in your current account on the public structure. If you specify more than one input specification, separate each with a comma (,).

/sw specifies one or more switches you want to include with each input specification. These switches let you tailor the directory information to your specific needs. If you do not specify any switches for an input specification, DIRECT proceeds as if you had specified the /DI switch. DIRECT includes a heading, file name, file type, size, protection code, creation date, and ending summary line. If you specify more than one switch for an input specification, include a slash (/) with each switch.

The following example shows the results of the DIRECT program:

```
$ RUN $DIRECT
DIRECT V9.0 RSTS V9.0
# LP0:=ESC.BAS,MAIL.MAI


 Name .Typ    Size   Prot    Date       SY:[3,214]
ESC   .BAS       1   < 60> 15-Jun-85

Total of 1 block in 1 file in SY:[3,214]

 Name .Typ    Size   Prot    Date       SY:[3,214]
MAIL  .MAI      21   < 60> 09-Apr-85

Total of 21 blocks in 1 file in SY:[3,214]
```

Items reported are:

Item	**Description**
Name.Typ	File name and type
Size	Size (in 512-byte blocks) of each file
Prot	Protection code
Date	Date of creation

Item	Description
(Disk)	Disk where the file resides, and the user's PPN. In the previous example, DIRECT displays information about the files located on the system disk (SY:) in user account [3,214].
(Summary)	Summary line that includes the total number of blocks used, the number of files listed, and the disk location and PPN of the user's account.

Note

Because the DIRECT program follows pointers through a disk directory, the program may display incorrect information if a user changes the pointers during program execution. For example, this may occur if another user opens a file in the current account while you are running DIRECT.

On completion of the directory listing, DIRECT returns to the command level # prompt. You can specify another directory listing or exit from DIRECT by entering CTRL/Z.

Wildcard Specifications

You can substitute the asterisk (*) character and the question mark (?) character wildcards in the file name and file type fields. In addition, you can substitute the * character in the PPN field.

You can substitute the * character for the entire file name, the entire file type, both the file name and file type, and the entire file specification. For example:

File Name and Type	Meaning
*.TBL	Substitute for file name
FILE.*	Substitute for file type
.	Substitute for file name and type
*	Substitute for file specification

The following shows the default interpretations of the * character wildcard that DIRECT applies to file specifications:

User Types	**DIRECT Interprets As**	**Files Listed**
null(RET)	*.*	All files
*	*.*	All files
.*	*.*	All files
*.	*.	All files having a null file type
.	*.	All files having a null file type
.TYP	*.TYP	All files having .TYP file type
FILE	FILE.*	All files having a FILE file name

You can also use the ? character wildcard to substitute for one or more alphanumeric characters in the file name and/or file type fields. Although you can substitute the ? character for each of the six characters of the file name and the three characters of the file type, it is simpler to use the * character in these cases. Therefore, you would usually mix ? characters with alphanumeric characters of the file name and file type. You can include ? characters in the file name and still use the * character for the file type and vice versa. For example:

User Types	**Files Listed**
DIRE??.DIR	All files having DIRE as the first four characters in their file name and having a .DIR file type
B?????.TBL	All files whose names start with B and have a .TBL file type
FILE.B??	All files with file names of FILE and file types starting with B
PROG??.*	All files with PROG as the first four characters in their file name
*.BA?	All files with BA as the first two characters in their file types

You can substitute the * character for the project number, the programmer number, or both (PPN). For example:

PPN	Meaning
[*,214]	Substitute for the project number
[1,*]	Substitute for the programmer number
[*,*]	Substitute for PPN

DIRECT Switches

You can specify one or more switches with each input file specification. If you do not include any switches for an input file specification, DIRECT uses the /DI switch as the default.

Table 2-3 lists the DIRECT switches and their meaning.

Table 2-3: DIRECT Switches

Switch	Format	Meaning
Individual Switches		
Name	/NA	Lists file name only.
Type Extension	/TY or /EX	Lists file name and type.
Size	/SZ or /SI	Lists file name, type, size (in blocks) of each file. DIRECT lists a C if file is contiguous, P if file is protected (nondeletable), and L if file is located (placed). The /SI switch cannot immediately follow the DIRECT CCL command. For example, assuming the DIRECT CCL command is DIR, you cannot type DIR/SI; however, you can type DIR/SZ.

Table 2-3: DIRECT Switches (Cont.)

Switch	Format	Meaning
Individual Switches		
Allocated	/AL	Lists file name, type, and number of blocks allocated to the file. The number of blocks listed is always a multiple of the cluster size. For example, if the cluster size is 4 blocks and the actual file size is 5 blocks, the allocation is listed as 8 blocks.
Open status	/OP	Lists file name, type, open status, and access count. DIRECT appends a U to the access count if the file is currently open in UPDATE mode, and a W if the file allows others write access. DIRECT lists the access count as two numbers separated by a slash (/). The first number indicates the number of users who currently have the file open in any mode except read regardless mode. The second number indicates the number of users who currently have the file open in read regardless mode.
Run-time	/RT	Lists the name of the run-time system that created the file.
Protection	/PR	Lists the file name, type, and file protection.
Last access	/LA	Lists the file name, type, and date of last access or update, depending on disk initialization.
Date	/DA	Lists the file name, type, and date of creation.
Time	/TI	Lists file name, type, date and time of day of creation.
Cluster size	/CL	Lists file name, type, and cluster size.

Table 2-3: DIRECT Switches (Cont.)

Switch	Format	Meaning
Individual Switches		
Summary	/SU	Lists summary data only to include number of designated files and total number of blocks occupied by designated files.
Position	/PO	Lists file name, type, and position of the file on disk (the device cluster number of the file's first block). See the *RSTS/E System User's Guide*.
Symbolic attribute Attribute	/SA or /AT	Lists file name, type, symbolic file attributes, and caching status. File caching is indicated by one of the following: CACHE:ON:RAN; automatically cached randomly when open CACHE:ON:SEQ; automatically cached sequentially when open CACHE:OFF:SEQ; not automatically cached, but if cached, it is cached sequentially.
Octal attribute	/OA	Lists file name, type, and a series of octal numbers that represent file attributes.
Aggregate Switches		
Brief Fast	/BR or /F	Lists file name and type and a brief ending summary message.
Full	/FU	Lists heading, file name, type, size, protection code, date of last access, date of creation, time of creation, cluster size, associated run-time system, file position, open status, and an ending summary message.

Table 2-3: DIRECT Switches (Cont.)

Switch	Format	Meaning
Aggregate Switches		
Directory,slow Listing,slow Slow	/DI:S or /LI:S or /S	Lists all relevant data to include heading, file name, type, 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 display on the same line), files marked for deletion (see /MD), and an ending summary message.
Directory Listing Carriage return	/DI or /LI or <RET>	Lists most important data to include heading, file name, type, size, protection code, creation date, and an ending summary message.
Marked for deletion	/MD	Lists all files, and flags those files marked for deletion by appending an * character after the file name. Tentative files are marked for deletion until closed.
No match	/N	Lists only those entries that do not match the specified input file specification.
General Switches		
Heading	/HD	Displays heading at top of columns in the listing for those listings that normally do not include a heading. If you specify /HD without another switch, DIRECT displays an error message.
Wide	/W	Lists data across the width of a line rather than one item per line. Useful with large directory listings and individual switches. When used alone, /W lists file names and types across the width of a line and an ending Summary message.

Table 2-3: DIRECT Switches (Cont.)

Switch	Format	Meaning
General Switches		
Help	/HE	Displays the DIRECT.HLP file, which describes the DIRECT program.
Backwards	/BK	Lists 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 you use /BK to list files in the public structure and the public structure includes multiple disks, the listing reflects reverse order for each disk. DIRECT cannot list directories in reverse order if the directory contains more than 200 files.

DIRECT Examples

Consider the following example:

```
        $ RUN $DIRECT

        DIRECT V9.0  RSTS  V9.0

        # /DI:S

Name .Typ    Size   Prot   Access     Date      Time    Clu  RTS     Pos  Op/rr
SY:[2,106]
MAIL  .MAI      4   <188> 13-Jun-85 13-Jun-85 09:53 AM   4 ...RSX  1090   0/0
RF:FIX=512  FO:SEQ  USED:5:0        RECSI:512
                            EX:1
SPR2  .OBJ      3   < 60> 09-Apr-85 09-Apr-85 08:31 AM   4 ...RSX   677   0/0
RF:VAR=128  FO:SEQ  USED:3:44       RECSI:122     CC:IMP
MAIL  .SYS     14   < 60> 08-May-85 08-May-85 09:05 AM   4 ...RSX  6447   0/0
FLINT .LOG      5   < 60> 09-Apr-85 09-Apr-85 02:34 PM   4 ...RSX  8613   0/0
BLDQUE.LOG     24   < 60> 13-Jun-85 13-Jun-85 09:55 PM   4 BAS4F   4955   0/0
BRIAN .NAD      1   < 60> 15-Jan-85 15-Jan-85 09:54 AM   4 TECO   13529   0/0
SNOOP .CMD      1   < 60> 15-Jan-85 15-Jan-85 11:15 AM   4 RT11    3386   0/0

        Total of 52 blocks in 7 files in SY:[2,106]
```

In this example, you run the DIRECT program and select the /DI:S switch. DIRECT lists all relevant information for all files that reside on the system disk in your account. DIRECT then reports these items:

Name.Typ	**File name and type**
Size	Size (in blocks) of each file
Prot	Protection code
Access	Date of last access
Date	Date of creation
Time	Time of creation
Clu	File cluster size
RTS	File's run-time system
Pos	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 in any mode except read regardless (Op); and the number of times the file is currently open in read regardless mode (rr)

Immediately below the headings, DIRECT lists the disk location of the files and the user's PPN.

Some files in this example have additional descriptive data listed on the line following the file data. This data lists the file attributes, which are required by RMS-11 software. For the meaning of the listed attributes, see the section "File Attributes."

Several of the listings that DIRECT produces contain only one or two columns. By default, DIRECT outputs columns in a vertical format. In these cases, you can use the /W switch and obtain a listing that spans the width of your terminal or paper, if you print the listing. For example:

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

SY:[3,214]
NONAME.BAS   ESC   .BAS   SYS6  .BAS   RESCUE.BAS   FIP16 .BAS   RBLOCK.BAS
NUMBER.BAS   WBLOCK.BAS   TD    .BAS   SWAP  .BAS   INDEX .BAS

11 Files 36 Blocks
```

In this example, DIRECT outputs the listing to line printer LP0. The listing includes the file name and type of all files in your current account having BA as the first two characters in the file type. DIRECT formats the listing into six columns. The listing ends with a summary statement showing the number of files and their total size in blocks.

Several of the listings that DIRECT produces do not include column headers. In these cases, you can obtain column headers by using the /HD switch. Repeating the previous listing, but including the /HD switch, you obtain the following listing:

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

Name .Typ     Name .Typ     Name .Typ     Name .Typ     Name .Typ     Name .Typ
SY:[3,214]
NONAME.BAS    ESC   .BAS    SYS6  .BAS    RESCUE.BAS    FIP16 .BAS    RBLOCK.BAS
NUMBER.BAS    WBLOCK.BAS    TD    .BAS    SWAP  .BAS    INDEX .BAS
```

If you specify the /HD switch but do not also include another switch, DIRECT displays this error message:

```
?/HD illegal without other switches
```

You can list the directory of an account other than your own by specifying the PPN field in the command line. However, DIRECT limits your listing to those files whose protection codes permit read, write, or execute access by your account. For example:

```
# PROJ=[120,3]/DI:S
```

DIRECT creates the file PROJ.DIR in your current account and writes all relevant data to the file for all files in account [120,3] for which you have read, write, or execute access. For more information about protection codes, see the *RSTS/E System User's Guide*.

DIRECT as a CCL Command

Your system manager can install a CCL command for DIRECT on your system. When you run DIRECT with a CCL command, you enter the program at the command level. DIRECT does not display an identifying header or a # prompt. Therefore, you must include DIRECT command line specifications in the CCL command line. For example:

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

In this example, DIRECT lists the file name, type, size, protection code and creation date of all BASIC-PLUS source files (.BAS) on line printer LP1.

When you run DIRECT with a CCL command, the defaults remain the same as when you specify RUN $DIRECT. Output defaults to your terminal, input defaults to all files in your user's account, and the switch defaults to /DI.

File Attributes

File attributes are file description data that RMS-11 software requires. For instance, both COBOL and BASIC-PLUS-2 use RMS-11. The User File Directory (UFD) for a given file contains the file attributes 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 file attributes in two formats:

- Symbolically, using the /SA, /AT, and /S switches
- Octally, using the /OA switch

The following example shows the use of the /SA switch to obtain file attributes in a symbolic format:

```
#/SA/HD


 Name .Typ
SY:[2,211]
RMSTST.OBJ
 RF:VAR=128   FO:SEQ    USED:7:22     RECSI:128    CC:IMP
PFILE .DAT
 RF:FIX=56    FO:IDX    USED:8:0      RECSI:56     CC:IMP
CHTST1.OBJ
 RF:VAR=128   FO:SEQ    USED:2:396    RECSI:90     CC:IMP
RMSTST.B2S
PROOF .BAS
RDPLN1.DOC
```

Table 2-4 lists the symbolic file attribute descriptions.

Table 2-4: Symbolic Attribute Fields

Attribute	Meaning	Description
RF	Record format	Displays the maximum record length of the file. The possible formats are: o UDF -- undefined o FIX -- fixed o VAR -- variable o VFC -- variable /fixed control o STM -- stream
FO	File organization	Displays type of file organization: o SEQ -- sequential o REL -- relative o IDX -- indexed
USED	Blocks in use	Displays the number of blocks currently in use, followed by the number of bytes used in the last block.
RECSI	Record size	Displays the size, in bytes, of the largest file record.
CC	Carriage control	Displays the type of carriage control: o IMP -- implied o FOR -- FORTRAN
NOSPAN	No span	If displayed, indicates that records are not allowed to span block boundaries.
BK	Bucket size	Displayed if the file organization is relative or indexed; indicates the bucket size in blocks.
EX	Extension	If you do not use the default, indicates the number of blocks added to the file each time you increase its size.
HS	Header size	If you do not use the default, indicates the fixed header size.

An octal format file attribute listing normally consists of seven 6-digit words. A listing of file attributes for RMS-created files, however, consists of nine or ten words. The additional words contain information required by RMS-11.

The following example shows the use of the /OA switch to obtain an octal format file attributes listing. Following the listing is the annotation describing the octal file attributes for file FILE.DAT.

```
#/OA/HD


 Name .Typ
SY:[2,211]
FILE .DAT 7    < 60> 13-Jun-85  13-Jun-85  02:10 PM   8  RSX  0
 001041 000070 000000 000010 000000 000007 000000 000001 000070 000010
RMSTST.OBJ
 001002 000200 000000 000007 000000 000007 000026 000000 000200
RMSTST.MAP
 000002 000167 000000 000211 000000 000211 000354
PFILE .DAT
 001041 000070 000000 000007 000000 000010 000000 000001 000070
RMSTST.B2S
RMSTST.CMD
RMSTST.ODL
PROOF .BAS
```

Annotation for file FILE.DAT:

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 (00000000 00000001 binary), 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 additional blocks (8 decimal) each time it is extended.

Table 2-5 lists the ten words and their meanings.

Table 2-5: Octal 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 digits -- 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 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 nonzero only if the file is larger than 65535 blocks.
7	The number of bytes in use for the last block used. 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.

Table 2-5: Octal File Attributes (Cont.)

Word	Meaning
8	In binary notation, the bytes are: Low byte — The bucket size, in blocks, in the range 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 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. The content of this word represents the number of blocks added to the file each time it is extended.

Both RMS-11, and all languages that use RMS-11, process file attribute data. BASIC-PLUS and FORTRAN IV do not use RMS-11 and, therefore, do not correctly process files that have attributes. You cannot use programs written in BASIC-PLUS or FORTRAN IV with files produced by RMS-11 unless they are specially programmed to handle file attributes. FORTRAN-77 uses file attributes but does not use RMS-11.

PIP is the only generally used program that preserves file attributes and run-time systems in disk-to-disk transfers within one system. When PIP copies a disk file to a nondisk device, the program performs file format conversions, either automatically, or as you specify with switches. PIP can also perform file format conversions when copying from a nondisk device to a disk file, creating the appropriate file attributes in the process.

When moving files between systems using magnetic tape as the transportation medium, you have two alternatives:

- o Use the RSTS/E BACKUP or PIP programs for transfers between two RSTS/E systems.
- o When the copied files are RMS-11 files and the transfer is between a RSTS/E system and a system in the RSX-11 family, use the RMS BACKUP package.

See the *RSTS/E RMS-11 User's Guide* for more information about RMS-11 and file attributes.

DIRECT Error Messages

When DIRECT finds an error in a command, the program displays an error message followed by the # prompt. You must then retype the entire command correctly.

Table 2-6 lists the DIRECT error messages and their meanings.

Table 2-6: DIRECT Error Messages

Message and Meaning
?DECTAPE ERROR <text> An error occurred while processing the DECtape.
?DEVICE NOT DIRECTORY STRUCTURED Device specified does not use a directory for file access.
?DIRECTORY OF dev:[n,m] IS EMPTY The account [n,m] on device dev: contains no entries.
?DISK PACK IS NOT ON-LINE Either the pack or cartridge you specified is not mounted or it is off-line.
?DISK PACK IS RESTRICTED The device is restricted (SET DEVICE/RESTRICTED) and you lack the DEVICE privilege.
?/HD illegal without other switches. When you specify /HD, you must also specify another switch.
?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 The file specification contains an undefined switch as indicated by text.

Table 2-6: DIRECT Error Messages (Cont.)

Message and Meaning
?INVALID DEVICE SPECIFICATION Either the specified device is invalid or the device does not exist on the system.
?MAGTAPE ERROR <text> An error occurred while processing the magnetic tape.
?MAGTAPE IS NON-FILE STRUCTURED RSTS/E does not understand the tape format.
?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 do not have the necessary privilege.
?OUTPUT MUST BE FILE STRUCTURED <filename> The output device must be file structured; for example, a mounted disk.
?UNABLE TO CREATE OUTPUT FILE <filename> The device may be write protected or there is not enough space in the directory of the account.
?TOO MANY FILES FOR INVERTED DIRECTORY LISTING DIRECT limits use of the /BK switch to accounts with fewer than 200 files.

Comparing Files: The FILCOM Program

The FILCOM (File Compare) program compares two ASCII files, line by line, and identifies differences. When you run FILCOM you specify the files you want to compare and (optionally) one or more switches that direct the comparison.

Privilege Required

To run FILCOM, you need the following:

- o Read access to the files you want to compare
- o Create/Rename access for any output file you create

Running FILCOM

To run FILCOM, type:

```
$ RUN $FILCOM
```

FILCOM then displays this two line identification header:

```
FILCOM V9.0   RSTS   V9.0
File Comparison Program  13-Jun-85 11:59PM
```

Following the identification header, FILCOM displays the prompt:

```
Output to <KB:>?
```

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

- o The device designator or a RSTS/E file specification to which FILCOM writes the comparison data.

  If you specify only a device, a RSTS/E file specification, or press RETURN to accept the user terminal as a default, FILCOM displays additional prompts. These prompts request the files you want to compare and let you set the parameters of the comparison. The next section describes the prompts. In answer to this prompt, you can specify one or more switches listed in Table 2-7. These switches provide an alternative method for specifying comparison parameters.

- o A single command line that contains all of the information that FILCOM needs to make the file comparison. The command line has the format:

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

  If you specify the full command line, FILCOM does not display other prompts 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. The section "FILCOM Single Command Line" describes the command line specification and Table 2-7 describes the FILCOM switches.

- o A command line that contains a DCL command file name. The DCL command file is an ASCII text file you create with PIP or an editor. See the *RSTS/E Guide to Writing Command Procedures*. The file must contain all the information FILCOM needs to make the file comparison. The command line has the format:

  ```
  Output to <KB:>? @file name
  ```

  File name is the name of a DCL command file. The DCL command file can contain a single line command as described in the previous response, or it can contain multiple line answers to the FILCOM prompts as described in the first response. If the DCL command file contains answers to the FILCOM prompts, you must answer each prompt in sequence as described in the section "FILCOM Prompts." You cannot nest another DCL command file within the original response. If FILCOM finds an at sign (@) character in the DCL command file, the program processes it as an assignable account specification.

After FILCOM completes a file comparison, the program returns to the Output to <KB:>? prompt for additional input. To exit from FILCOM, enter CTRL/Z in response to the prompt.

FILCOM Prompts

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

- o The files you want to compare
- o The number of successive lines in those files that constitute a match

- o Whether FILCOM considers BASIC-PLUS continuation lines as part of a numbered line
- o Whether FILCOM compares blank lines

You can specify one or more switches in response to FILCOM's initial prompt. You can use a switch specification to override prompt displays and to provide additional instructions to the FILCOM program.

Following the initial prompt, FILCOM displays prompts for the file names of the ASCII files you want to compare. For example:

```
Output to <KB:>? <RET>
Input File #1? SORT1.BAS
Input File #2? SORT2.BAS
```

In this example, FILCOM compares SORT1.BAS to SORT2.BAS and displays the result of the comparison on your terminal. Include the file type when you specify the files you want to compare. You can use the * and ? wildcard characters to designate the input files. See the section "Wildcards" for information on their use.

The How Many to Match <3>? prompt asks for the number of lines that FILCOM considers a match. As FILCOM compares the lines in a file, it displays the differences until it comes to a series of lines that are identical. Your answer to this prompt determines the number of lines. 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 example, you create a file named SORT.DIF that contains the results of the comparison between files SORT1.BAS and SORT2.BAS. FILCOM examines the files and records the differences until it finds three lines that match. To accept the default of three lines, press the RETURN key.

Following the How Many to Match <3>? prompt, FILCOM asks if you want 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 example, FILCOM considers continuation lines as part of a numbered program line when it compares SORT1.BAS and SORT2.BAS. The default for this prompt, <NO>, causes FILCOM to compare only single lines.

Consider the following program line:

```
100 FOR J%=1% TO 15 &
\   PRINT 3.14*J%/2. &
\   NEXT J%
```

If you answer the BASIC+ Lines <NO>? prompt with Y (for YES), FILCOM compares all three multiline statements (FOR, PRINT, and NEXT) as a single line. If you accept the NO default, FILCOM only compares the first line (FOR J%=1% TO 15% &) in SORT1 to a single line in SORT2.

The last prompt in the series asks if you want 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 example, FILCOM compares SORT1.BAS to SORT2.BAS and writes the differences to SORT.DIF. FILCOM matches five successive lines, considers BASIC-PLUS continuation lines, and ignores the presence of blank lines. If FILCOM finds two blank lines in SORT1 and five blank lines in SORT2, it does not consider this a difference. See the section "FILCOM Examples" for examples of FILCOM comparisons.

FILCOM Single Command Line

In response to FILCOM's initial prompt, you can type a single command line that specifies all files and comparison parameters. This command line has the format:

```
[output=]input1,input2[/switch]
```

where:

output — is one of the following:

- A complete RSTS/E file specification containing device, ppn , file name, and type designators.
- A RETURN key that specifies the default user terminal. If you press RETURN, FILCOM displays prompts. See the section "FILCOM Prompts."

- o A wildcard specification (see the section "Wildcards").

- o No specification. If you do not specify a file or = symbol but rather an input specification only, FILCOM displays its output on your terminal.

=	specifies an output file. If you type an = symbol with no output specification, FILCOM writes the result of its file comparison to a file on the public structure. FILCOM creates this file in the current account using a file name taken from the first input file and using a .DIF file type.
input1, input2	names the two files that FILCOM compares, separated by a comma. You can describe these files with full file specifications, file names and types, or wildcards (see the section "Wildcards").
/switch	names one or more optional switches.

See the section "FILCOM Examples" for examples of the FILCOM single command line.

FILCOM Switches

You can specify one or more switches in the FILCOM command line or in response to the FILCOM prompts.

Table 2-7 lists the FILCOM switches and their meanings.

Table 2-7: FILCOM Switches

Switch	Format	Meaning
Match	/MA[:n]	Specifies the number of lines in the file that constitute a match (see the section "FILCOM Prompts"). If you do not specify the number of lines (n), the default is three.
Blank, No blank	/BL, /NOBL	Specifies whether FILCOM considers blank lines in its comparison (see the section "FILCOM Prompts"). The default is not to consider blank lines (/NOBL).

Table 2-7: FILCOM Switches (Cont.)

Switch	Format	Meaning
BASIC, No BASIC	/BA, /NOBA	Specifies whether FILCOM considers BASIC-PLUS continuation lines as part of a numbered program line (see the section "FILCOM Prompts"). The default is not to consider continuation lines (/NOBA).
Patch	/PA	Directs FILCOM 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.
Append	/AP	Directs FILCOM to open the output file for APPEND (see /PA switch). If you omit this switch, FILCOM deletes any existing output file of the same name before writing differences. With the /AP switch, you can direct FILCOM to write differences into an existing output file.
Summary	/SU	Directs FILCOM to list the total number of differences found in the files without listing individual lines.
Limit	/LI[: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.
Compare	/CO[:m:n]	Converts tabs to the appropriate number of spaces and directs 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.

Use of these switches is not restricted to single line commands. You can specify any of these switches in answer to the Output to <KB:>? 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 are a match: that is, FILCOM does not compare blank lines and does not consider BASIC-PLUS continuation lines as part of a numbered program line (/MA:3/NOBL/NOBA). However, if you specify the /PA switch, FILCOM compares blank lines and considers BASIC-PLUS continuation lines. 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, FILCOM displays the error message:

```
?Illegal command line /x
```

In this error message, /x refers to the illegal switch.

If you specify the /LI switch, FILCOM ceases to compare the files when it finds a set number of differing lines during a single comparison. 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 displays the error message:

```
?Limit (x) reached on last compare, skipping rest of file
```

In this message, (x) refers to the /LI specification or the default of 60. FILCOM sets no limit unless you specify the /LI switch.

You can use the /CO switch to direct FILCOM to ignore characters beyond a certain column. For example, suppose you want to compare a program that contains comments beginning in column 80. Specify /CO:1:79 in the command line to cause FILCOM to convert all tabs to an equivalent number of spaces (in order to properly count character positions) and to compare each line from column 1 to column 79. FILCOM ignores everything after column 79.

Consider the following examples:

```
Output to <KB:>? SORT.DIF=SORT1.BAS,SORT2.BAS
```

FILCOM compares SORT1 and SORT2 and writes the differences to file SORT.DIF. FILCOM sets the /MA:3, /NOBA, and /NOBL switches by default.

```
Output to <KB:>? = SORT1.BAS,SORT2.BAS/LI:10
```

FILCOM compares SORT1 and SORT2 until either the number of differing lines in a single comparison exceeds 10 or the comparison is complete. FILCOM creates a file named SORT1.DIF in the current account to store the results of the comparison. FILCOM sets the /MA:3, /NOBA, and /NOBL switches by default.

```
Output to <KB:>? SORT1.BAS,SORT2.BAS/BA/BL/SU
```

FILCOM compares SORT1 and SORT2 and displays a summary of the differences on your terminal. FILCOM compares blank lines (/BL) and considers BASIC-PLUS continuation lines (/BA).

Wildcards

You can include wildcard characters in both the output and input file specifications in a FILCOM command line. The * wildcard replaces an entire specification field. The ? wildcard replaces a single character within a field.

FILCOM output file specifications can contain a wildcard for the file name or type, but not both. For output, you must specify a file name or type, or default both fields (see the section "FILCOM Single Command Line").

You can specify FILCOM input files using full RSTS/E file specifications or using wildcard characters. When you include wildcards 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:

Specification	Action by 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 with 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, the same character, and the same file type.

Specification	Action by FILCOM
.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.

You can use wildcards to make multiple file comparisons based on a single command line specification. When you make multiple comparisons, FILCOM displays the differences found in each set of files in the standard manner (see the section "FILCOM Examples"). In addition, FILCOM displays a summary at the completion of the comparisons. The summary has the format:

```
?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

z   is the total number of files compared based on the wildcard specification

You can specify wildcard file names only for the first input file if the file resides on disk. If the first input file is not a disk file, FILCOM displays the following error message:

```
?Wildcard filename illegal on device x
```

In this error message, x refers to the specified input device. FILCOM then returns to the Output to <KB:>? prompt. The second input file does not have to reside on disk.

If a wildcard specification causes FILCOM to create an output file name that is syntactically incorrect, (one containing embedded spaces, for example) FILCOM displays the following error message:

```
?Illegal filename x created from y by z
```

where:

x   is the illegal file name

y   is the first input file wildcard specification you typed

z   is the file specification for the second input file

If you specify a wildcard character for both the file name and type for the output file, FILCOM displays the following error message:

```
?Wildcard x illegal for output filename
```

In this error message, x refers to the illegal wildcard specification.

FILCOM Examples

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

TEST1

```
10  REM A
20  REM B
30  REM C
40  REM D
50  REM E
60  REM F
70  REM G
80  REM H
90  REM I
100 REM J
110 REM K
120 REM L
130 REM M
140 REM N
150 REM O
160 REM P
170 REM Q
180 REM R
190 REM S
200 REM T
210 REM U
220 REM V
230 REM W
240 REM X
250 REM Y
260 REM Z
```

TEST2

```
10  REM A
20  REM B
30  REM C
70  REM G
80  REM H
90  REM I
100 REM J
110 REM 1
120 REM 2
130 REM 3
140 REM N
150 REM O
160 REM P
170 REM Q
180 REM R
190 REM S
200 REM T
210 REM U
220 REM V
222 REM 4
224 REM 5
230 REM W
240 REM X
250 REM Y
260 REM Z
```

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

```
$ RUN $FILCOM
FILCOM File Comparison Program  13-Jun-85  11:14 AM
Output to <KB:>? TEST1.BAS,TEST2.BAS
```

This command line directs the FILCOM 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:>? <CTRL/Z>
$
```

FILCOM displays 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 displays line 40 because it has no match in TEST2.BAS. FILCOM continues to display TEST1 until it finds three matching lines (the default), the first line of which is line 70.

As another example, consider the following two programs:

FILE1.BAS

```
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
80    END
```

FILE2.BAS

```
5     EXTEND
1     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 directs FILCOM to create a file named FIL.DIF and to write the results of the comparison of FILE1.BAS and FILE2.BAS to that file:

```
$ RUN $FILCOM
FILCOM V9.0  RSTS  V9.0
File Comparison Program  13-JUN-85 11:54 AM
Output to <KB:>? FIL.DIF=FILE1.BAS,FILE2.BAS/BA/MA:2/SU
Output to <KB:>? <CTRL/Z>
$
```

The specified switches instruct FILCOM to consider BASIC-PLUS continuation lines as part of a numbered program line and to write differences when more than two lines fail to match. The /SU switch directs FILCOM to write a summary of the differences in FIL.DIF. After completion of the comparison, FIL.DIF contains the following:

```
FILCOM V9.0  RSTS  V9.0
File Comparison Program  13-JUN-85 11:54 AM
Comparing: 1) [2,211]FILE1.BAS TO 2) [2,211]FILE2.BAS

?2 Differences Found.
```

The following series of commands directs FILCOM to display the results of its comparison of FILE1.BAS and FILE2.BAS on your terminal. However, the number of lines needed to match is four, and a full description of the differences is requested:

```
$ RUN $FILCOM
FILCOM V9.0  RSTS  V9.0
File Comparison Program  16-DEC-84 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:>? <CTRL/Z>
$
```

FILCOM Error Messages

Table 2-8 lists the FILCOM error messages and their meanings.

Table 2-8: FILCOM Error Messages

Message and Meaning
?A NULL LENGTH FILE? The file examined by FILCOM (the first or second input file) is empty.
?FILE HAS NON-ASCII STREAM ATTRIBUTE - CAN'T READ <filename> The file <filename> contains file attribute data that is not ASCII stream. The FILCOM program can only compare ASCII file data. After FILCOM returns this error, it continues to the next set of files or prints the Output to <KB:>? prompt.
?ILLEGAL COMMAND LINE x The FILCOM command line or file specification contains illegal syntax.
?ILLEGAL FILENAME <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. After FILCOM returns this error, it continues to the next set of files or prints the Output to <KB:>? prompt.
?LIMIT n REACHED ON LAST COMPARE, SKIPPING REST OF FILE <filename> The number of differing lines in a single compare (n), specified by the /LI switch, is exceeded. FILCOM continues to the next set of files or prints the output to <KB:>? 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.
?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. FILCOM aborts the operation.

Table 2-8: FILCOM Error Messages (Cont.)

Message and Meaning
?WILDCARD FILENAME ILLEGAL ON DEVICE x You typed a wildcard specification for a first input file that does not reside on a disk device.
?WILDCARD '*.*' ILLEGAL FOR OUTPUT FILENAME <filename> FILCOM requires either a file name or type in the output file specification. You can use a wildcard for one of these fields (but not both), or you can default both fields.
?WILDCARD PPN SPECIFICATION ILLEGAL You cannot use wildcard characters in the PPN field.

Transferring Files Between Devices: The FIT Program

The FIT (FIle Transfer) program transfers files between RSTS/E directory-structured devices and RT11 directory-structured devices. FIT can also transfer files from DOS directory-structured disks to RSTS/E directory-structured devices, and list device directories. FIT can perform the following tasks:

- Transfer files between RSTS/E structured devices and RT11 structured devices. This includes transfers between RSTS/E devices and RX01 or RX02 flexible diskettes that are maintained in the RT11 directory-structured format.
- Transfer files from a DOS disk to a RSTS/E device.
- List the directory of a RSTS/E, RT11 or DOS structured device, including RX01 and RX02 flexible diskettes.
- Delete files on an RT11 structured device.
- Initialize (zero) an RT11 structured device.
- Compress (squeeze) the files on an RT11 structured device.

Privilege Required

To run the FIT program and use its various features, you need the following:

- Read access to input files and files included in a directory listing
- Create/Rename access to any output files you create
- Write access to any files you delete
- DEVICE privilege for restricted tape drives and restricted RX01/RX02 disk drives
- RDNFS privilege for non-RSTS/E input disk devices
- WRTNFS privilege for non-RSTS/E output disk devices
- SYSIO privilege to set the protection/privilege bit when transferring files to a RSTS/E system

Running the FIT Program

To run FIT, type:

```
$ RUN AUXLIB$:FIT
```

FIT displays an identifying header followed by the FIT> prompt:

```
FIT V9.0 RSTS V9.0
FIT>
```

The prompt indicates that FIT is ready to accept a command line.

You can obtain help on using FIT by typing /HE in response to the FIT prompt.

To exit from FIT, enter CTRL/Z in response to the prompt.

Transferring Files with FIT

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

```
[output[/sw]=]input[/switch]
```

where:

input and output	are file specifications.
/switch	is an optional switch that specifies the structure of the device on which the file resides. Specify: o RSTS o RT11 o DOS o /WATCH to cause FIT to log its file transfers

The input and output file specifications can be full RSTS/E specifications of the following format:

```
dev:[proj,prog]filename.type/PROT:n
```

where:

dev:	specifies a device. If you do not specify a device, the default is the public structure. Legal output and input devices are explained later in this section.
proj,prog	is the project-programmer number (PPN). If you do not specify one, the default is your current account. FIT ignores the PPN if the device has an RT11 directory structure. You can also use wildcards (see the PIP utility section "Wildcard Specifications").
filename.type	is the file name consisting of one-to six-alphanumeric characters and, optionally, a file type consisting of one-to three-alphanumeric characters. If you omit the file name and type for output, FIT uses the input file name and type. For input, however, you must specify a file name. You can also use wildcards (see the PIP utility section "Wildcard Specifications").
/PROT:n	is the file protection code. If you omit the protection code on output, FIT assigns your current default protection code. FIT ignores any input file protection code specification. If you have SYSIO privilege, you can set the protection privilege bit.

The legal input devices for a FIT file transfer are:

- o Any disk with a RSTS/E directory structure.
- o Any disk, diskette, or DECtape with an RT11 directory structure. This includes RT11 disks, RX01/RX02 flexible diskettes, and TU56 DECtape.
- o Any disk with a DOS directory structure.

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

- o Valid RSTS/E devices. These include RSTS/E disks, magtape, TU56 DECtape, keyboards, and line printers.
- o Devices with an RT11 directory structure. These include RT11 disks, RX01/RX02 flexible diskettes, and TU56 DECtape.

If the input device is a DOS disk, the legal output devices are all valid RSTS/E devices. These include RSTS/E disks, magtape, TU56 DECtape, keyboards, and line printers.

You cannot use FIT to directly transfer files from a DOS disk to magtape or to an RT11 structured device. The algorithm that FIT uses to output to these devices requires that it know the file length ahead of time. Since DOS disks do not contain this information, FIT does not support direct transfers from DOS disks to RT11 devices.

If you need to perform such a transfer, you can do it in two steps:

1. Transfer the files from the DOS disk to a RSTS/E disk
2. Perform a second transfer from the RSTS/E disk to the RT11 device

The specifications you can use for /switch in the command line for file transfer operations are:

Switch	Abbreviation	Description
/RT11	/RT	The file is on an RT11 directory-structured device. When you specify this switch, FIT ignores files with a .BAD file type on transfers and deletions. When you also specify the /SQ switch (see the section "Compressing an RT11 Structured Device"), FIT displays a warning message and aborts the compression when it finds a .BAD file type.
/DOS	/DO	The file is on a DOS format disk. This switch is legal only for an input file specification.
/RSTS	/RS	The file is on a RSTS/E disk or device. This switch is illegal with RX01 and RX02 flexible diskettes.
/WATCH	/W	When you specify this optional switch, FIT logs file transfers on your terminal.

The following example shows copying a file and using the /W switch:

```
$ RUN AUXLIB$:FIT
FIT      V9.0    RSTS V9.0
FIT> DY1:=[1,2]COMMON.MAC/W
[File SY:[1,2]COMMON.MAC copied to DY1:[COMMON.MAC]
```

Listing the Directory of a Device

Use the /LI or /DI switch to list the directory of a device on your terminal. The command line has the following format:

```
dev:[proj,prog]filename.type/L[I][/sw]
```

where:

dev:	specifies the device designator. The default is the system disk (SY:).
proj,prog	specifies the project-programmer number (PPN) of the directory you want to list. The default is the current account. This field can contain wildcards. See the PIP utility section "Wildcard Specifications."
filename.type	specifies the file names and types of the files you want to list. The default is all files in the current account. This field can contain wildcards.
/LI	specifies the directory listing switch. You can also use /DI in the command line. You can abbreviate these switches to /L and /D.
/sw	specifies an optional switch for the directory structure of the device.

The following example shows the use of the /LI switch:

```
$ RUN AUXLIB$:FIT
FIT     V9.0    RSTS V9.0
FIT> DY1:/LI
Directory of DY1:?????.???

Name   .Typ    Size      Date         RTS     Pos
COMMON.MAC       41    13-Jun-85    TECO      26
 RF:STM       FO:SEQ    USED:41:165      RECSI:85

<UNUSED>      427                              67

Total of 41 blocks in 1 file in DY1:
Total of 427 free blocks in DY!:
```

For RT11 file-structured devices, FIT also displays file attributes if the directory has 13 extra words for each entry (see the section "Initializing an RT11 Structured Device"). Those sections of the directory that have free space are flagged as UNUSED in the listing.

You can include one of the following optional switches with the /LI or /DI switch:

Switch	Meaning
/RT11	Indicates that the device must have an RT11 directory structure. If you include this switch and the device is not an RT11 device, FIT returns an error message. You can abbreviate this switch to /RT.
/DOS	Lists the directory of a DOS disk. This operation generates a listing for each file that includes account number (UIC), file name, file type, file length, protection code, date of creation, the first block number of the file, and whether or not the file is contiguous. If you include the /DOS switch and the device is not a DOS disk, FIT returns an error message. You can abbreviate this switch to /DO.

If you do not include an additional switch with the /LI operation, FIT lists the directory of the device whether it is DOS, RSTS/E, or RT11 directory structured.

Deleting Files on an RT11 Structured Device

You can use the /DE switch to delete a file on an RT11 directory-structured device. To delete a file, type a command in the following format in response to the FIT> prompt:

```
dev:filename.type/DE[/switch]
```

where:

dev:	specifies the device designator. It must be an RT11 structured device.
filename.type	specifies the file names and types of the files you want to delete. This field can contain wildcards (see the PIP utility section "Wildcard Specifications").
/DE	specifies the file deletion switch.
/WATCH	directs FIT to log each file deletion on your terminal. You can abbreviate this switch to /W.

If the specified device is not RT11 directory structured or if the file does not exist, FIT displays an error message. If FIT finds an error as it accesses the device, it reports a RSTS/E error.

Initializing an RT11 Structured Device

You can initialize (zero) an RT11 structured device by typing a command in the following format in response to the FIT> prompt:

```
dev:/ZE:m[/switch]
```

where:

dev: specifies the device you want to initialize.

/ZE is the switch that tells FIT to initialize the specified device. If you specify a value for m, FIT adds m extra words of directory information for each file to the minimal RT11 directory structure. If you omit m, FIT adds 13 extra words.

Note

> RT-11 systems currently ignore these extra directory words. However, if future releases of RT-11 attach significance to these words, you may have to specify the /ZE:0 switch for RT11 files that are to be 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 specify only values of 0 or 13 for m with the /ZE switch.

The following example shows initializing a device:

```
$ RUN AUXLIB$:FIT
FIT     V9.0     RSTS V9.0
FIT> DY1:/ZE
Really zero DY1:? Y
```

Note

> Never use FIT to initialize an RT11 disk that isolates bad blocks in .BAD files. Initializing such a device destroys the bad block information. If you want to delete all files except the .BAD files, perform a wildcard delete operation using the /RT11 switch (see the previous section).

You can also use the /N switch with the /ZE switch to create more directory segments on the initialized device. The switch has the format:

```
/N:n
```

where:

/N: is the switch that tells FIT to add segments to the directory. You can use /N to control the size of the directory on the device, and therefore, the number of files stored on that device.

n specifies the number of segments you want to add to the directory. Each segment consists of two 512-byte blocks. The value of n must be in the range 1 to 31. Use the following formula to determine the maximum number of files that can fit on a device:

```
Max Files = n*INT(507/(m+7))
```

In this formula, 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). You may have to compress the device with the /SQ switch (see the following section) before the maximum number of files is reached. If you do not specify n, FIT applies defaults based on the device capacity (in blocks).

Table 2-9 lists the defaults and the maximum number of files allowed on a particular device as determined by your specification in /ZE:m.

Table 2-9: FIT Initialization Defaults

Device Capacity (Blocks)	Device	/ZE:0 Directory Segments	/ZE:0 Maximum Files	/ZE:13 Directory Segments	/ZE:13 Maximum Files
< 1000	RX01 RX02 RX50	4	288	10	250
> 1000 and < 10000	RK05 RK05F	16	1152	31	775
> 10000	*	31	2232	31	775

* See the *RSTS/E System Manager's Guide* for specific devices with a capacity greater than 10,000 blocks.

Compressing an RT11 Structured Device

You can compress (squeeze) an RT11 structured device by typing a command in the following format in response to the FIT> prompt:

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

The /SQ switch compresses the files that are currently on the device you specified in dev:. Use the /SQ switch to make room on an output device for additional files that you want to transfer to the device.

The space on RT11 devices allocated to the directory of files can become fragmented. If this is the case, an attempt to add another file to the device can result in the error message ?Directory overflow or ?Device full. You can use the /SQ switch to compress the contents of the device and attempt to provide enough room for the additional files. If you interrupt a compress operation, data may be lost on the device you are compressing. Thus, FIT ignores any CTRL/C you enter at the keyboard during a compress operation.

If you also specify the /RT11 switch, FIT first inspects the device to see if it contains any files with the .BAD file type. If it does, FIT displays an error message and aborts the compress operation.

Because of the way that RT11 directory entries are made, you may need to compress the device to use the directory space, even if you add files contiguously without deletions.

Note

> RT11 format DECtape (TU56) is much slower to access than RSTS/E format DECtape. You should use the /SQ switch only when necessary to transfer between RSTS/E and RT-11 systems. For example, a FIT compress operation (/SQ) on an RT11 DECtape can take an hour or longer.

Maintaining RT11 Format with Bad Block Files

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

1. First initialize a disk that contains bad blocks on an RT-11 system before FIT can process it. The RT-11 system locates the bad blocks and allocates them to files with a .BAD file type.

2. Include the /RT11 switch (see the previous section) with the file specification whenever you reference the disk in the FIT command line.

This procedure directs FIT to ignore any files with a .BAD type on transfer to and from RT11 devices. FIT also ignores .BAD files on delete operations. During compress operations (see the previous section) FIT displays an error message and aborts the operation if it detects a .BAD file.

FIT Error Messages

If FIT finds an error during the open or transfer phase of a RSTS/E file operation, it returns the appropriate RSTS/E error (see the *RSTS/E System User's Guide*) and aborts the transfer.

FIT generates errors related to an RT11 directory structure before transferring any data to the RT11 structured device. Because FIT does not finalize the directory entry for the RT11 file until the transfer is complete, an error during transfer frees the space reserved for the file.

Table 2-10 lists the FIT error messages and their meanings.

Table 2-10: FIT Error Messages

Message and Meaning
Common Directory Structure Errors
?ARGUMENT NEEDED FOR You specified the /N switch without an argument.
?CANNOT OUTPUT TO DOS A DOS format disk is valid for input only.
?CAN'T ZERO A MOUNTED RSTS/E DISK You cannot zero a RSTS/E mounted disk.
?CONFLICTING SWITCH You specified an invalid switch combination.
?DEVICE <device-type> HAS <text> STRUCTURE You specified an /RT11, /RSTS/E, or a /DOS device switch and the device does not have that structure.

Table 2-10: FIT Error Messages (Cont.)

Message and Meaning
Common Directory Structure Errors
?DEVICE NOT ALLOWED FOR INPUT The specified input device is not a legal device.
?ILLEGAL ARGUMENT IN You specified an illegal switch argument.
?ILLEGAL OPERATION FOR DEVICE The specified operation is illegal for the specified device.
?ILLEGAL WILDCARDS FOR OUTPUT You cannot use the ? character in the output file specification.
?NO FILES MATCHING The specified file name for the directory listing does not exist.
?NO INPUT FILE You did not specify a file name or a device.
?NOT A RECOGNIZED DIRECTORY STRUCTURE The specified device is not an RT11, RSTS/E, or a DOS structure.
?OPERATION DOES NOT TAKE OUTPUT SPEC You cannot specify an output file specification using the /LI, /DE, /ZE, /SQ, and /HE switches.
?PROGRAM MUST BE PRIVILEGED The program has insufficient privileges to run properly. Set the privilege bit in the protection code.
?RSTS/E DISK NOT MOUNTED The specified disk is not mounted.
?UNRECOGNIZED SWITCH(ES) You specified an invalid switch.

Table 2-10: FIT Error Messages (Cont.)

Message and Meaning
RT11 Directory-Structured Errors
?DEVICE FULL Either there is not enough space in the directory or you are attempting to transfer a file larger than 65,535 blocks to an RT11 structured device.
?DEVICE HAS .BAD FILE(S) You cannot compress an RT11 structured device that has .BAD files
?DEVICE TOO LARGE You cannot initialize a device that is larger than 65,535 blocks.
?DIRECTORY CORRUPTED The specified RT11 directory structure is corrupted.
?DIRECTORY OVERFLOW The output device has no more space for directory entries. It may be possible for you to use the /SQ switch (see the section "Compressing an RT11 Structured Device") to free more space for the directory. If this is not successful, you must delete some of the files on the device before youcan add more. 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 the section, "Initializing an RT11 Structured Device") when you initialize the device.
?FILENAME AND TYPE NEEDED FOR /DE You must specify a file name and a file type when deleting a file from an RT11 structured device.
?ILLEGAL INPUT FOR RT11 OUTPUT You cannot transfer files from a DOS disk to an RT11 structured device.
?/ZE MUST BE SPECIFIED WITH /N When creating additional segments, you must include the /ZE switch when using the /N option.

Transferring Diskette Information: The FLINT Program

The FLINT program (FLexible diskette INTerchange) transfers information between IBM flexible diskettes and standard RSTS/E disk files. FLINT can only transfer information between single density IBM flexible diskettes, which are mounted in RX01/RX02 drives, and RSTS/E disks. FLINT cannot process double density IBM diskettes. You can use FLINT to perform the following operations:

- Transfer information from IBM flexible diskettes to standard disk files that RSTS/E can read (for example, DK:, DP:, SY:). FLINT can also transfer files that reside on multiple diskettes.

- Transfer information from RSTS/E readable disks to the diskettes that an IBM system can read. FLINT can also transfer large files that require multiple diskettes.

- List the directory of an IBM diskette. This directory contains information related to the unique data format of an IBM diskette.

- Initialize the directory of a diskette.

- Initialize the directory and erase all data on a diskette.

To perform these operations, FLINT communicates with you by dialogue. There are two different dialogues for the transfer operations: IBM-to-RSTS/E and RSTS/E-to-IBM. There are also dialogues for listing an IBM directory, initializing, and erasing a diskette.

Privilege Required

To run the FLINT program and use its various features, you need the following:

- Read access to input files and files included in a directory listing

- Create/Rename access to any output files you create

- Write access to any files you delete

- DEVICE privilege for restricted RX01/RX02 disk drives

- RDNFS privilege for non-RSTS/E input disk devices

- WRTNFS privilege for non-RSTS/E output disk devices

Running FLINT: The Initiation Commands

To run FLINT, type:

```
$ RUN AUXLIB$:FLINT
```

FLINT displays an identifying header, followed by a # prompt. This indicates the program is ready to accept one of five initiation commands. Each of the following commands initiates a different dialogue:

Command	Initiates the dialogue to:
/DIRECTORY	List an IBM diskette directory
/TORSTS	Transfer from an IBM diskette to a RSTS/E disk file
/TOIBM	Transfer from a RSTS/E disk file to an IBM diskette
/ZERO	Initialize the directory of a diskette
/ERASE	Initialize the directory and erase all data on a diskette

You can abbreviate each command (at the minimum) to its first three letters: /DIR, /TOR, /TOI, /ZER, and /ERA.

After FLINT completes a file transfer, it returns to the # prompt. To exit from FLINT, enter CTRL/Z.

Listing the Directory of an IBM Diskette

Type /DIR in response to the # prompt to begin the two-question dialogue for listing the directory of an IBM diskette. In the following dialogue description, explanations follow FLINT's questions.

```
Output to?
```

This question asks you where you want FLINT to output the directory of the diskette(s). Press RETURN to display the output on your terminal. If you want to write the output to a file, include an output file specification. The full file specification has the following format:

```
dev:[proj,prog]filename.type/PROT:n
```

where:

dev:	specifies the device. You must specify the device name.
[proj,prog]	specifies the project-programmer number (PPN). The default is the current account.
filename.type	specifies the file name and file type for the output file. You can use the * and the ? wildcard characters in the file name and file type fields. You must specify at least the file name.
n	specifies the file protection. The default is 60.

FLINT then prompts for the location of the IBM diskette(s):

```
Directory of?
```

This question asks you to specify the location of the IBM diskette(s) whose directory you want listed. Press RETURN to specify diskette DX0. Type any other disk specifications in the form DXn: or n, where n is an integer between zero and seven. You must separate multidiskette specifications with commas.

The following example displays the directory of an IBM diskette on your terminal:

```
$ RUN AUXLIB$:FLINT
FLINT  V9.0  RSTS  V9.0
#/DIR
Output to? <RET>
Directory of? DX1:

DIRECTORY OF DX1:

DSN    BRL   BOE     EOE     EOD     BI   MVI   VSN

DATA   080   01001   73026   01001

TOTAL OF 1 DATA SET ON DX1:
```

Table 2-11 lists and describes the abbreviated headings for the IBM diskette directory.

Table 2-11: FLINT Abbreviated Directory Headings

Heading	Name	Meaning
DSN	Data Set Name	The one- to eight-character name you have given the data set (corresponds to file name in RSTS/E).
BRL	Block/Record Length	The number of positions (bytes) containing data in each 128-position (byte) sector.
BOE	Beginning Of Extent (BOE)	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.
EOE	End Of Extent (EOE)	The address of the last sector reserved for this data set (in the same format as BOE).
EOD	End Of Data (EOD)	The address of the next unused sector within the data set extent.
BI	Bypass Indicator	A space in this field means the data set is intended for processing; a B means it is not.
MVI	Multivolume Indicator	A space 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.
VSN	Volume Sequence Number	The sequence of volumes in a multivolume data set. Blanks indicate that volume sequence checking is not to be performed.

Transferring IBM Diskette Data to RSTS/E

Type /TOR in response to the # prompt to begin the dialogue for transferring IBM diskette data to a standard RSTS/E disk file. In the following dialogue description, explanations follow FLINT's questions.

```
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 that has the format:

```
dev:[proj,prog]filename.type/PROT:n[/NH]
```

where:

dev:	specifies the RSTS/E disk device. The default is the system disk.
[proj,prog]	specifies the project-programmer number (PPN). The default is the current account. You cannot use the * and the ? wildcard characters.
filename.type	specifies the file name and file type of the file to receive the output of the diskettes.
n	specifies the file protection. The default is 60.
/NH	specifies the optional switch that eliminates the 6-byte header.

FLINT then prompts for translation:

```
Translate from EBCDIC to ASCII <Yes>?
```

This question asks you if you want FLINT to translate the diskette data from its current EBCDIC format (unreadable by RSTS/E) into ASCII format, which is readable by RSTS/E. Pressing RETURN directs FLINT to translate the data. If you type NO, FLINT does not translate the data.

```
Input from?
```

This question asks you to specify the location of the IBM diskette(s) from which FLINT will transfer data. Press RETURN to specify diskette DX0. You can specify a different location by typing a response in the format:

```
DXn:
or
n
```

where DXn refers to the location of the input diskette(s) and n is an integer from zero to seven.

FLINT then proceeds with the transfer of the first data set from the specified diskette. FLINT displays a message in the format:

```
dsn is the Data Set Being Transferred
```

where dsn refers to the name of the first data set on the first diskette.

If you want FLINT to transfer a specific data set, give a response in the format:

```
DXn:dsn
```

where:

DXn — refers to the location of the input diskette(s) and n is an integer from zero to seven.

dsn — specifies the data set name, containing one to eight ASCII characters, including blanks (spaces and tabs). FLINT recognizes these blanks as part of the IBM data set name. If FLINT cannot find the specified data set, the program displays the informational message ?File not found, followed by the Input from? prompt.

FLINT examines the data set label, which is a header with statistical information, to determine if the data set resides on more than one diskette. If the transfer requires more than one diskette but you specify only one diskette, FLINT displays this message:

```
dsn Resides on more than one diskette -
Do you Wish to CONTINUE <NO>?
```

If you type YES, FLINT proceeds to its next question. If you press RETURN to accept the default response, FLINT aborts the transfer and displays the # prompt.

FLINT next computes the blocking factor it uses in the transfer of the current data set and displays a message expressing that factor as the ratio of IBM to RSTS/E records. This message has the format:

```
N XXX Character IBM Records = 1 RSTS Record
```

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. FLINT divides the RSTS/E record size by the IBM record size and displays the quotient, rounded down to the nearest integer, as N in the previous message.

For example, if each IBM record contains 80 data characters, FLINT places 6 full IBM records in a 512-byte RSTS/E record (512/80 = 6.4). FLINT displays the message:

```
6 80 Character IBM Records = 1 RSTS Record.
```

After displaying the blocking factor message, FLINT transfers the data from each specified input device.

When FLINT copies data from IBM diskettes whose record size does not divide equally into the RSTS 512 byte record size, it transfers N data records and then fills the extra space with null characters. In the previous example, FLINT transfers six 80-byte records from the IBM diskette, using 480 bytes of storage in the RSTS/E file. FLINT then fills the remaining 32 bytes with null characters before proceeding with the next record transfer.

If the specified input device does not contain the entire data set, 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 transfer data and display Next input device? requests until the current diskette contains the end of the specified data set.

On multivolume input, FLINT continually checks data set names against the one you specify and, if possible, also checks volume numbers for correct sequence. For example, if FLINT is transferring the data set IDAT and the third input volume specified contains no IDAT, FLINT displays the message:

```
File Not Found - DXn:IDAT
```

FLINT then displays the message:

```
Next Input Device?
```

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

```
Volume #[m] cannot follow Volume #[n]
```

FLINT then displays the message:

```
Next Input Device?
```

If no fatal errors occur, FLINT completes the IBM-to-RSTS/E transfer and displays a summary message that has the format:

```
Exchange Completed
X IBM Records Read
Y Logical (Z Physical) RSTS Records Written
```

where:

X indicates the number of IBM records read.

Y indicates the number of logical RSTS/E records written.

Z indicates the number of physical RSTS/E records used.

Specifying the Known Diskettes of a Data Set

Before you begin the transfer of a multivolume data set, you may know on which diskettes the data set resides. Respond to the Input from? question by specifying the diskette on which the data set begins, then the data set's name, and finally the other diskette(s) on which the data set is continued. For example:

```
Input from? DX1:DATASETA,DX2:,DX3:,DX4
```

In this example, you tell FLINT to mount (in order) the diskettes containing DATASETA on consecutively numbered drives. You can mount and specify diskettes out of sequence but there is a chance for confusion. If the unsequenced order of specifications does not match exactly the unsequenced order of mounting, FLINT displays the message:

```
?Volume #[m] cannot follow Volume #[n]
```

Format of RSTS/E Disk File Header Records

The first logical record on the RSTS/E output file is reserved for a statistical header. This header is six bytes long, and contains the following information (all fields are 2-byte integer fields in standard CVT%$ format; see the *BASIC-PLUS Language Manual*):

Bytes	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.

When FLINT begins the data transfer, it copies the statistical header into the first six bytes of the first RSTS/E logical record. It then fills the rest of the logical record with null characters before continuing with the data transfer. Note that if the logical record size of the IBM diskette is less than six bytes, the FLINT program requires two RSTS/E logical records in which to store the statistical header.

Use the /NH switch in response to the Output to? dialogue question to stop the transfer of this statistical header. For example:

```
Output to? DK0:INFO.DAT/NH
```

In this example, you tell FLINT to transfer data to the file INFO.DAT on disk drive DK0:. FLINT does not transfer the statistical header, and begins the data transfer at the first logical record on the RSTS/E disk file.

Transferring RSTS/E Files to IBM Diskettes

Type /TOI in response to the # prompt to begin the dialogue for transferring a RSTS/E disk file to IBM diskette(s). In the following dialogue description, explanations follow FLINT's questions.

```
Output to?
```

This question asks you to specify the IBM data set that is to receive data from the RSTS/E disk. Respond with a data set name in the following format:

```
DXn:dsn
```

where:

DXn: specifies the diskette you want to receive the data set and n is a device number from zero to seven.

dsn specifies the output data set name composed of one to eight characters including blanks. If you omit dsn, FLINT assumes the name RSTS.

You must separate multivolumes with commas. For example:

```
DX0:DATSETC,DX2:
```

FLINT then prompts for translation:

```
Translate from ASCII to EBCDIC <Yes>?
```

This question asks if you want FLINT to translate the RSTS input data from its current ASCII format into EBCDIC format. Pressing RETURN tells FLINT to translate the data. If you type NO, FLINT does not translate the data.

```
Input from?
```

This question asks you to specify the location of the RSTS/E file from which FLINT will transfer the data. Specify an input file in RSTS/E format:

```
dev:[proj,prog]filename.type
```

where:

dev:	specifies the RSTS/E disk device. The system disk is the default.
[proj,prog]	specifies the PPN. You cannot use the * and the ? wildcard characters. The default is the current account.

FLINT then prompts for the record length:

```
Record Length <128>?
```

This question asks you to specify the size (in bytes) of each IBM output record. As the default of 128 indicates, press RETURN to specify output records that contain 128 characters each. IBM records contain 1 to 128 characters and are fixed length. To specify a shorter record length, type any number that is a power of 2 within the range 0 to 128 (that is, 2, 4, 8, 16, 32, or 64).

Note

> To make sure that no data is lost in transfer, you must specify a record size that divides equally into the RSTS/E 512-byte record size. You can specify any number that is a power of 2.
>
> If you specify any other record size, FLINT transfers data from each 512-byte record to N IBM records (see the following blocking factor information) ignoring any left over data in the RSTS/E record. It then proceeds to the next RSTS/E record for transfer.
>
> For example, if you specify an IBM record length of 80, FLINT transfers six 80-byte records to the IBM diskette, for a total of 480 bytes. FLINT then ignores the 32 data bytes remaining in the RSTS/E record and proceeds to the next record.

FLINT next computes the blocking factor to use 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 format:

```
1 RSTS Record = N IBM Records
```

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. FLINT divides the RSTS/E record size by the IBM record size and displays the quotient, rounded down to the nearest integer, as N in the previous message.

After computing the blocking factor, FLINT proceeds with the transfer.

If FLINT determines that the RSTS/E file will not fit on the number of diskettes you specify, FLINT displays the question:

```
Volume #(n + 1)?
```

This question asks you to specify an additional diskette for output. Type a response in the format:

```
DXn:
or
n
```

where DXn specifies the diskette you want to receive the data set and n is a device number from zero to seven.

FLINT continues to ask the Volume #? question and to accept one of these responses until it can complete the transfer. For all diskettes used in the transfer, FLINT labels each diskette with volume sequence numbers and, where necessary, with continuation markers.

Note

> You cannot use more than 99 diskettes for one data set. If you attempt to use more, FLINT displays the error message:
>
> ```
> ?Maximum number of volumes exceeded
> ```
>
> FLINT then displays the # prompt.

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

```
Exchange Completed
data set name Resides on x Diskettes
```

where:

data set name	indicates the name of the data set
x	indicates the number of diskettes that comprise the named data set.

Initializing and Erasing a Diskette

Type /ZERO in response to the # prompt to begin the dialogue for initializing the directory of a single density diskette. In the following dialogue description, explanations follow FLINT's questions.

```
Zero which diskette?
```

This question asks you to specify the location of the diskette whose directory you want to initialize. The drive specification has the format:

```
DXn:
or
n
```

where DXn specifies the location of the diskette and n is a device number in the range zero to seven.

```
Really Zero DXn: <No>?
```

FLINT asks this question to make sure that you specify the correct diskette. DXn: indicates the diskette you want to initialize. If you answer YES or Y to this question, FLINT initializes the directory of the diskette. If you accept the default by pressing RETURN, FLINT displays a message, followed by the # prompt.

Type /ERASE in response to the # prompt to begin the dialogue for erasing a single density diskette. In the following dialogue description, explanations follow FLINT's questions.

```
Erase which diskette?
```

This question asks you to specify the diskette drive of the data set you want to erase. The drive specification has the format:

```
DXn:
or
n
```

Where DXn specifies the diskette drive and n is a number in the range zero to seven.

```
Really Erase DXn: <No>?
```

FLINT asks this question to make sure that you specify the correct diskette. DXn: indicates the diskette you want to erase. If you answer YES or Y to this question, FLINT initializes the directory of the diskette and overwrites all data with zeros. This operation can take considerable time. If you accept the <NO> default by pressing RETURN, FLINT displays a message, followed by the # prompt.

FLINT Dialogue Examples: /DIRECTORY, /TORSTS, /TOIBM, /ZERO, and /ERASE

In the first example, FLINT displays the directory listing of an IBM diskette on your terminal:

```
$ RUN AUXLIB$:FLINT
FLINT     V9.0    RSTS    V9.0
#/DIR
Output to?
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:
```

In the next example, FLINT transfers data from an IBM diskette to a RSTS/E disk file:

```
$ RUN AUXLIB$:FLINT
FLINT  V9.0  RSTS  V9.0
#/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 this /TORSTS dialogue, FLINT transfers data set PHONE DB from diskette DX0: to the RSTS/E output file PHONE.DAT. By default, FLINT translates the data into ASCII. FLINT computes and displays the blocking factor (4 IBM records to 1 RSTS/E record).

In the next example, FLINT transfers a RSTS/E disk file to an IBM diskette:

```
$ RUN AUXLIB$:FLINT
FLINT    V9.0   RSTS   V9.0
#/TOIBM
Output to? DX1:MYDATA
Translate from ASCII to EBCDIC <Yes>? <RET>
Input from? MYDATA.DAT
Record Length <128>?

1 RSTS Record = 4 IBM Records

Exchange Completed
MYDATA Resides ON 1 Diskette
```

In this /TOIBM dialogue, FLINT transfers the RSTS/E file MYDATA.DAT to the IBM data set MYDATA on diskette DX1:. By default, FLINT translates the data to EBCDIC. FLINT computes and displays the blocking factor (1 RSTS/E record to 4 IBM records).

The last example uses the /ZERO operation to initialize the directory of a diskette on unit 0 (DX0:), and the /ERASE operation to initialize and erase a diskette on unit 1 (DX1:):

```
$ RUN AUXLIB$:FLINT
FLINT V9.0  RSTS   V9.0
#/ZER
ZERO which diskette? 0
Really ZERO DX0: <No> ?  Y
#/ERA
ERASE which diskette? 1
Really ERASE DX1: <No> ?  Y
#  <CTRL/Z>
$
```

FLINT Error Messages

FLINT displays both fatal and nonfatal error messages. If the error is nonfatal, FLINT repeats the prompting question. If the error is fatal, FLINT aborts the transfer. See the *RSTS/E System User's Guide* for a list of error messages specific to RSTS/E.

Table 2-12 lists the FLINT error messages and their meanings.

Table 2-12: FLINT Error Messages

Message and Meaning
Recoverable Errors
?DATA SET NAME TOO LONG The IBM data set you specified name has more than eight characters, including spaces and tabs.
?DATA SET NOT FOUND FLINT cannot find on the current diskette the data set you specified.
?ILLEGAL DATA SET NAME You specified a data set name where none is permissible.
?ILLEGAL FILE NAME The file name you specified contains unacceptable characters or violates the specification format.
?ILLEGAL PPN You specified wildcards in the account number, or you typed an account number that does not exist.
?LOGICAL DEVICE NAME NOT ASSIGNED You specified a logical device name that was not previously assigned.
?OUTPUT DEVICE MUST BE DISK In a /TORSTS transfer, you specified an output device other than a RSTS/E disk (the only permissible type).
?VOLUME #[m] CANNOT FOLLOW VOLUME #[n] In a multivolume /TORSTS transfer, you specified an input volume number [m] that is not one greater than the previous volume number [n] you specified.

Table 2-12: FLINT Error Messages (Cont.)

Message and Meaning
Nonrecoverable Errors
?BOE = EOD - TRANSFER ABORTED In the specified data set, the Beginning Of Extent address is the same as the End Of Data address; that is, the data set has no length. ?DATA SET LABEL MUST BE HDR1 - TRANSFER ABORTED The label ID of a data set you specified is not HDR1, which IBM format requires. ?TRACK 00 DOES NOT CONTAIN ERMAP FIELD - TRANSFER ABORTED There is no ERMAP field in track 00 of the diskette you specified; according to IBM format, this field must be present.

Communicating with the System Manager: The GRIPE Program

The GRIPE program sends messages to the system manager. GRIPE writes your comments to a common file where they are held for system manager review.

Privilege Required

You do not need any special privilege to run the GRIPE program and send a message to the system manager. You do, however, need WACNT privilege to use the *LIST and *RESET commands.

Sending a Message

To run GRIPE, type:

```
$ RUN AUXLIB$:GRIPE
```

GRIPE displays an identifying header, followed by a $ prompt to indicate that it is ready to accept comments:

```
GRIPE 9.0   RSTS 9.0

Enter message below (<CTRL/Z> to end):
(*LIST to list gripes, *RESET to reset file)

Message: Jack:
Message: Please request that all users return the system
Message: line printer ON LINE after removing their printout.
Message: Thanks, Ann
Message:  <CTRL/Z>
Your message has been recorded
$
```

Following the Message: prompt, type the text of your comment, which GRIPE enters into the common file. End all text lines by pressing RETURN. If you end a text line by entering CTRL/Z, GRIPE exits and you lose the last line of text.

GRIPE indicates its acceptance of the text and its termination by displaying the message:

```
Your message has been recorded
```

When you end your message, GRIPE attempts to open comment file GRIPE.TXT with write access. If another user already has the file open with write access, GRIPE displays the warning message:

```
%GRIPE file in use - please wait...
```

GRIPE tries to open the file every 5 seconds up to a limit of 60 seconds. If GRIPE can open the file with write access within the time limit, GRIPE writes your message to the file and displays the Thank You message. If GRIPE cannot open the file within the time limit, GRIPE displays the error message:

```
?Unable to write to GRIPE file
?Please try again later
```

If comment file GRIPE.TXT does not have enough free space to store your message, GRIPE displays the error message:

```
?GRIPE file is full
?Please notify system manager
```

The *LIST Command

The following section shows how to display the contents of the GRIPE.TXT file using the *LIST command. You can:

- o Display the messages on your terminal
- o Send the messages to a line printer
- o Write the messages to a file

Each comment in the GRIPE.TXT file consists of:

- o An identification line, which contains:
  - Project-programmer number (PPN) of the user
  - Account name or (no name) of the user
  - Terminal keyboard number of the user
  - The date and time the message was entered
- o The comment text

An example using the *LIST command is:

```
$ RUN AUXLIB$:GRIPE
GRIPE V9.0 RSTS V9.0

Enter message below (<CTRL/Z> to end):
(*LIST to list gripes, *RESET to reset file)

Message: *LIST
Output to <_KB:GRIPE.LOG>?

*** From user [1,214] (Hogan) on KB1:  31-Jan-85 04:01 PM ***

Jack:
Please request that all users return the system
line printer ON LINE after removing their printout.
Thanks, Ann
$
```

After GRIPE displays the Message: prompt, type *LIST and press RETURN. You must include the leading asterisk (*); otherwise, GRIPE interprets the command as a comment line.

If file GRIPE.TXT is empty, GRIPE displays the informational message:

```
No GRIPES found
```

GRIPE THEN exits to your keyboard monitor. Otherwise, GRIPE displays the prompt:

```
Output to <_KB:GRIPE.LOG>
```

Press RETURN to display the comments on your terminal or type a line printer device or a file specification.

If the GRIPE.TXT file is more than 90 percent full, GRIPE displays the warning message:

```
%GRIPE file almost full -
please *RESET it.
```

The *RESET Command

The following section shows how to delete the GRIPE.TXT file using the *RESET command.

An example using the *RESET command is:

```
$ RUN AUXLIB$:GRIPE
GRIPE V9.0 RSTS V9.0

Enter message below (<CTRL/Z> to end):
(*LIST to list gripes, *RESET to reset file)

Message: *RESET
GRIPE file reset
Message: <CTRL/Z>
$
```

After GRIPE displays the Message: prompt, type *RESET and press RETURN. You must include the leading asterisk; (*) otherwise, GRIPE interprets the command as a comment line. After deleting the file, GRIPE returns you to your keyboard monitor.

GRIPE Error Messages

Table 2-13 lists and describes the GRIPE error messages.

Table 2-13: GRIPE Error Messages

Message and Meaning
?GRIPE FILE IS FULL PLEASE NOTIFY SYSTEM MANAGER The GRIPE text file is full. List the file and then RESET it.
??PROGRAM FAILURE IN GRIPE There has been an unexpected failure in the GRIPE program.
?UNABLE TO CREATE GRIPE FILE Cannot create the GRIPE text file.
?UNABLE TO OPEN GRIPE FILE Cannot open the GRIPE text file.
?UNABLE TO OPEN OUTPUT FILE <file name> Cannot open the specified output file or device to list or store GRIPE text.
?UNABLE TO RESET GRIPE FILE Cannot reset the GRIPE text file.
?UNABLE TO WRITE TO GRIPE FILE PLEASE TRY AGAIN LATER Cannot gain write access to the GRIPE text file.
?WACNT PRIVILEGE REQUIRED You must have WACNT privilege to list and reset the GRIPE text file.

Obtaining Help: The HELP Program

The system library program HELP contains files that describe other system programs and system resource commands. HELP is an interactive program that prompts you for topics, subtopics, and sub-subtopics. To obtain information on system programs or commands, you type the desired topic, and optional subtopic and sub-subtopic in response to the HELP prompts. Your response directs HELP to display the specified information.

Privilege Required

To run the HELP program and use its various features, you need Create/Rename access to any file you create using the /OUTPUT switch.

Running the HELP Program

To run HELP, type:

```
$ RUN HELP$:HELP
```

HELP then displays a header and prompts you for the desired topic. For example:

```
HELP V9.0 RSTS V9.0

Topic?
```

In response to the prompt, type or press one of the following:

- o The name of the topic about which you want information. If you abbreviate the name, HELP displays information on all topics that match the abbreviation. In addition to the topic you can specify one or more subtopics (separated by spaces) associated with that topic. The level to which you can nest subtopics depends on the particular topic about which you want information.
- o A * wildcard character, which directs HELP to display information on all available topics.
- o The ? character, which directs HELP to display a list of available topics.

For example:

```
Topic?

Help is available on:

/OUTPUT     /PROMPT
ADVANCED    ASSIGN      ATTACH      BASIC       BYE

                 .
                 .
                 .

Topic?
```

To exit from the HELP program, enter CTRL/Z at any prompt. You can also exit HELP by pressing RETURN at the Topic? prompt.

If any level of HELP text contains more than 24 lines, HELP displays 24 lines and displays the prompt:

```
Press RETURN for more...
```

If you type any characters at this prompt, HELP ignores them and does not echo them on your terminal. HELP waits for you to press RETURN, or enter CTRL/Z to exit. HELP continues this process as long as there is more text to display.

HELP describes many of the topics by displaying additional subtopics. Depending on the particular topic, HELP may provide several layers of subtopics. The prompt for subtopics contains the original topic name and all subtopic names down to the current subtopic name. If there are no further subtopic layers, HELP prompts at the final subtopic layer. For example:

```
Topic? SYSTAT

SYSTAT

The SYSTAT program provides current system information. It may be
run with the command RUN $SYSTAT or invoked with the CCL command,

              .
              .
              .

Additional help is available on:

/A              /B              /C              /D

              .
              .
              .

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:

              .
              .
              .

SYSTAT Subtopic? <RET>

Topic? <RET>
$
```

Each time you press RETURN at the Subtopic ? prompt, HELP displays the next higher subtopic level until it reaches the the top Topic? prompt. If you press RETURN at this prompt, HELP exits.

If you specify a HELP topic that does not exist, HELP displays the list of available topics, followed by the Topic? prompt.

HELP as a CCL Command

Your system manager can define a CCL HELP command on your system. If the CCL HELP command name conflicts with the DCL HELP command, use the CCL prefix when you type the CCL command. The examples in this section assume that the CCL command is HELP and include the CCL prefix.

You can use CCL HELP with prompting or without prompting. To direct HELP to include prompting, include the /PROMPT switch immediately after the CCL HELP command. You can abbreviate the switch to /P. You cannot use the /P switch if you include the /OUTPUT switch (see the next section for a description of the /OUTPUT switch). When you include the /P switch, HELP displays descriptive text about how to use HELP, displays a list of available HELP topics, and displays a prompt.

For example:

```
$ CCL HELP/P
HELP V9.0 RSTS V9.0

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.)
       .
       .
       .

Topic?
```

From this point on, CCL HELP with prompting functions the same way as it does when you use the RUN HELP$:HELP command.

When you include the /P switch, you can also specify a topic and optional subtopics in the command line. In this case, HELP displays the information and displays the topic name Subtopic? prompt if there are subtopics. If there are no subtopics, HELP displays the Topic? prompt. For example:

```
$ CCL HELP/P PIP SWITCHES /IN

PIP

  SWITCHES

    /IN
    /Q

       Inspects eligible files; type 'Y' to transfer or
       anything not beginning with 'Y'  to  omit.  Type

               .
               .
               .

Topic?
```

If you do not include the /P switch but do include a topic and optional subtopics in the CCL command line, HELP displays the information and exits. For example:

```
$ CCL HELP QUE

QUE

  The QUE program creates requests, or jobs, to be executed by
  spooling programs. QUE can also list, modify, or kill pending
  requests.

               .
               .
               .

$
```

By default, CCL HELP displays the HELP text on your terminal. If you want to write the text to an output file, specify the /OUTPUT: switch. You can abbreviate the switch to /O:. If you include the /O: switch, you cannot include the /P switch. Immediately after the colon (:), specify the output file specification, a space, the topic, and optional subtopics. If you do not include a file type, HELP uses a default file type of .LST. HELP does not display the HELP text on your terminal. For example:

```
$ CCL HELP/O:SYSHLP.SAV SYSTAT
$
```

Manipulating Address Space Locations: The ODT Program

The ODT (Octal Debugging Tool) program opens a file, a peripheral device, or memory as an address space and allows you to examine and change word or byte locations within that address space. You can also list the contents of certain table locations in the operating system.

Privilege Required

To run the ODT program and use its various features, you need the following:

- o RDMEM privilege to read memory
- o RDMEM and SYSMOD privileges to read and change memory
- o RDWFS privilege to read a non-file-structured device
- o WRTWFS privilege to write to a non-file-structured device
- o Read access to read a file
- o Read and write access to read and change a file

How ODT Works

The ODT program immediately interprets and executes each character as you type it. This interpretive mode is called ODT submode or no delimiter character input mode and differs from the procedure other utilities use. Other programs interpret input only after you enter an entire line of characters. Because ODT performs processing based on single characters typed at the terminal, its language is highly interpretive and interactive.

Note

> ODT was designed for expert system programmers having a thorough knowledge of file formats and operating system tables. Use extreme care with ODT; mistakes can cause data loss or corruption.

The program accesses and manipulates data in word and byte locations based on octal values. The word is the 16-bit PDP-11 word and can have an octal value in the range of 0 to 177777, the maximum value obtainable with 16 bits. A word has high-order (odd address) and low-order (even address) bytes. A byte can have an octal value in the range of 0 to 377, the maximum value obtainable with 8 bits. The following symbols express the octal values that ODT prints:

Symbol	Meaning
n	Represents an octal integer between 0 and 17. The use of 8 or 9 generates an error.
k	Represents an octal value up to 6 digits in length. If you specify more than six digits or a value greater than 177777 (octal), ODT truncates the value to the low order (rightmost) 16 bits. If a minus sign (-) precedes the octal value, ODT uses the two's complement value of the number.

For example, ODT interprets the following values as shown:

Value	Interpretation
1	000001
-1	177777 (two's complement)
400	000400
-177730	000050 (two's complement)
1234567	034567 (truncated to low-order 16-bits)

You can represent a location within the permissible address space by typing an octal value or an expression that evaluates to an octal value. The correct forms and their interpretations are:

Symbol	Meaning
k	The six-digit octal value of k.
n,k	The resultant address is the value of k added to the contents of the relocation register specified by n. Relocation registers are numbered from 0 to 17 (octal).

Table 2-14 lists the special characters and symbols that ODT recognizes.

Table 2-14: ODT Characters and Symbols

Character Symbols	Meaning
/ k/	Opens the previously open location as a word, or opens the location designated by k as a word.
' k'	Opens the previously open location as a byte, or opens the location designated by k as a byte.
" k"	Gives the ASCII representation of the currently open or last open location, or of the location specified by k.
% k%	Gives the ASCII representation of the Radix-50 value in the currently open or last open location, or in the location specified by k.
<RET> k<RET>	Closes the currently open location, or modifies the contents of k followed by the currently open location with the value k and closes it.
<LF> k<LF>	Closes the currently open location and opens the next sequential location, or modifies the contents of the currently open location with the value of k before closing it and opening the next sequential location.
^	Closes the currently open location and opens the preceding sequential location.
_	Takes the contents of the currently open location as a program counter relative offset and calculates the next location to be opened; closes the currently open location and opens the location thus evaluated.
@ k@	Takes the contents of the currently open location as an absolute address, closes the currently open location; opens and prints the contents of the location thus evaluated. If k precedes @, the value k replaces the contents of the currently open location before it is closed.

Table 2-14: ODT Characters and Symbols (Cont.)

Character Symbols	Meaning
> k>	Takes the low order byte of the currently open location as a relative branch offset and calculates the address of the next location to be opened. Closes the currently open location and opens and prints the contents of the relative branch location thus evaluated. If k precedes >, the value k replaces the contents of the currently open location before it is closed.
<	Closes the currently open location and opens the last location explicitly open. Returns ODT to the origin of a sequence of relative locations determined by _, @, and > character operations.
,	Separates a relocation register number from an octal value. ODT adds the contents of the specified relocation register to the octal value following the comma, and forms a relocatable addres.
;	Separates multiple values in a list request using the L character and in a register operation using the R character.
.	Specifies the last explicitly open location similar to that used by the < character.
+ space bar	Adds the preceding value and following value and uses the result.
-	Subtracts the following value from the preceding value and uses the result.
R	Resets all relocation registers to -1 (177777).
nR	Resets relocation register n to -1 (177777).
k;nR	Sets relocation register n to the value k.
F	Sets relocation calculation for list requests using the L character.
1F	Disables relocation calculation set by the F character.

Table 2-14: ODT Characters and Symbols (Cont.)

Character Symbols	Meaning
C	Prints out monitor table symbolic names and memory addresses.
$S	Prints out the processor status word.
Q	Uses the last quantity printed by ODT.
k1;k2L	Prints the contents of locations k1 through k2 at the terminal.
1;k1;k2L	Prints the contents of location k1 through k2 on line printer unit 0.
2;k1;k2L	Prints the contents of location k1 through k2 on another device. ODT asks for a device designator by printing the DEVICE question.

Running and Terminating ODT

To run ODT, type:

```
$ RUN AUXLIB$:ODT
```

ODT displays an identification line and the FILE question, and waits for your response:

```
ODT V9.0  RSTS V9.0
FILE<MEMORY>?
```

Your response to this question determines the way ODT runs and what address space ODT accesses.

Table 2-15 lists the possible responses to the FILE<MEMORY>? prompt.

Table 2-15: ODT Question Responses

Response	Meaning
Press RETURN only.	Allows read access to memory only if you have RDMEM privilege.
Type a file specification, then press RETURN.	Allows read access to the file on the device specified. If you specify no device, ODT uses the system disk.
Press LINE FEED only.	Allows read and write access to memory only if you have RDMEM and SYSMOD privileges.
Type the file specification, then press LINE FEED.	Allows read and write access to the file specified.

ODT determines the address space by your response to the FILE<MEMORY>? prompt. ODT then displays the * prompt, indicating it is ready to accept commands. For example:

```
FILE<MEMORY>? ABC.DAT
*
```

ODT opens, for read and write access, file ABC.DAT on the system disk under the current account. To exit from ODT, enter CTRL/Z in response to the * prompt. For example:

```
* <CTRL/Z>
$
```

ODT closes any file currently open and returns control to your keyboard monitor.

Accessing Locations in the Address Space

Typing the slash (/) character access the address space as a word. For example:

```
*1000/ 004100
*
```

In this example, ODT:

- o Opens the location as a word
- o Generates a space
- o prints the six-digit octal contents of the word
- o Generates another space
- o Leaves the location open for change

To close the location, press RETURN. ODT:

- o Closes the location
- o Displays the * prompt
- o Does not open a new location

Typing the backslash (\) character opens a location as a byte. For example:

```
*1000\ 100
*
```

In this example, ODT:

- o Opens the location as a byte
- o Generates a space
- o Displays the six-digit octal contents of the byte
- o Generates another space
- o Leaves the location open for change

To close the location, press RETURN. ODT:

- o Closes the location
- o Displays the * prompt
- o Does not open a new location

To change location 1000, type a new six-digit octal value and press the RETURN key. For example, if location 1000 is open as a word:

```
*1000/ 004100 004000
*
```

In this example, ODT:

- o Replaces the current contents (octal 004100) with 004000
- o Closes the location
- o Displays the * prompt

Typing the slash (/) character at the * prompt displays the contents of the current word location. For example, if the current location is 1000:

```
*/ 004000
```

In this example, ODT:

- o Opens the current location
- o Generates a space
- o Displays the six-digit contents
- o Generates another space

If you press LINE FEED while a word location is open, ODT opens the next sequential location. For example, if location 1000 is open and you press LINE FEED:

```
*1000/ 004000<LF>
001002/ 012345
```

In this example, ODT:

- o Closes the current location
- o Generates a carriage return and line feed
- o Opens the next sequential location
- o Prints the address of the next location
- o Displays a slash (/) and space
- o Displays the contents of the word

If you press LINE FEED while a byte location is open, ODT performs the same actions as for a word location, except that the next location is treated as a byte.

Opening the Preceding Location

Typing a circumflex (^) character when a location is open causes ODT to close the location currently open and to open the location immediately preceding the one just closed.

For example, if you open two sequential locations in succession and then type the ^ character, ODT closes the last location open and opens the location you opened first. For example:

```
*1000/ 002345<LF>
001002/ 012740^
001000/ 002345
```

In this example, ODT:

- o Closes location 1002
- o Opens location 1000
- o Displays the contents of location 1000

If you type a value followed by the ^ character, ODT replaces the current contents with the specified value before closing the location.

If you type the ^ character and a location is not currently open, ODT opens and prints the contents of the word or byte location that was last open. For example:

```
*1000/ 002345
*^
001000/ 002345
```

In this example, ODT:

- o Displays the address of the location that was open last
- o Displays the contents of word location 1000

Opening a PC Relative Location

If you type the underscore (_) character when a location is currently open, ODT:

- o Closes the currently open location
- o Adds 2 to the address of the location just closed
- o Adds the resultant address sum to the contents of the location just closed
- o Opens the location specified by the contents of the location just closed
- o Displays the contents of the currently open location

For example:

```
*1000/ 000040_
001042/ 012345
```

In this example, ODT:

- o Closes location 1000
- o Adds 2 to the address (1000)
- o Adds the resultant sum (1002) to the contents (40) of location 1000
- o Opens location 1042 and prints its contents (12345)

This method of calculating the next location to open is similar to that used in relative addressing by the PC in the PDP-11 computer. Note that this method of address calculation is for position-independent code.

If the contents of the current location is an odd value, ODT opens and prints the contents of the low-order byte of the PC relative location. If you type a value followed by an underscore character, ODT modifies the current contents and uses the new value to calculate the PC relative address.

Opening an Absolute Location

Typing the at sign (@) character when a location is currently open causes ODT to use the contents of the current location as the address of the next location to open. As a result, ODT:

- o Closes the current location
- o Opens the calculated location
- o Prints its contents

For example:

```
*1006/ 001024 @
001024/ 000500
```

In this example, ODT:

- o Opens location 1024
- o Displays the contents

If you type a value followed by the @ character, ODT changes the contents of the current location to the new value and uses this value to determine the next location to open. The method is equivalent to absolute addressing on the PDP-11 computer, where the contents of the location following an instruction is taken as the address of the operand. The address is absolute because it remains constant regardless of where in memory the assembled instruction is executed.

Opening a Relative Branch Offset Location

Typing the greater than (>) character when a location is currently open causes ODT to use the signed value of the low-order byte of the current location to determine an offset from the current location. As a result, ODT:

- o Closes the current location
- o Opens the calculated location
- o Displays the contents of the new location

For example:

```
*1032/ 000407 >
001052/ 001456
```

In this example, ODT:

- o Multiplies the low-order byte (007) of the contents (000407) of the current location by 2, giving a result of 16 (octal)
- o Adds 2 to the address (1032) of the current location, giving a result of 1034
- o Sums the results of the two calculations (1034 + 16), giving the address (1052) of the next location to open
- o Closes the currently open location
- o Opens the new location and prints its address and contents

If you specify a value followed by the > character, ODT modifies the contents of the currently open location and uses the low order byte of the new value to calculate the relative branch offset location. For example:

```
*1032/ 000407 301>
000636/ 000010
```

In this example, ODT:

- o Interprets byte value 301 as a negative value because its high order bit is 1.
- o Multiplies the absolute value of 301 (77 octal) by 2, giving 176 (octal)
- o Subtracts the relative branch offset (176) from the address plus 2 of the current location, giving 636 as the address of the next location to open.
- o Opens the new location
- o Displays the contents of the new location

Returning to an Interrupted Sequence

Typing the less than (<) character causes ODT to close the currently open location and open the last explicitly open location. This command is useful when you have typed the _, @, and > characters, or any sequence thereof, and want to open the locations from which ODT calculated subsequent relative locations. For example:

```
*1032/ 000301 >
000636/ 000010 @
000010/ 123456 <
001032/ 000301
```

In this example, ODT:

- Opens location 10 after you type the > and @ characters
- Displays the contents of location 10
- Opens the last explicitly opened location (1032) after you type the < character
- Displays the contents of location 1032

Printing the Contents of a Range of Locations

You can display the contents of a range of locations using the format:

```
[device;]start-loc;end-locL
```

where:

device — is the number of the device to which you want ODT to send the contents. If you do not specify a device, ODT displays the contents on your terminal. If you specify 1, ODT sends the contents to line printer LP0:. If you specify a number other than 1, ODT prompts you for the line printer device number. If you specify a device number, include the semicolon (;).

start-loc — is the starting location of the range.

end-loc — is the ending location of the range.

L — is the letter, which indicates that you want ODT to output the contents of a range of locations.

To display the contents of locations 0 through 776 on your terminal, respond to the * prompt as follows:

```
*0;776L
```

To print the contents of locations 0 through 776 on line printer LP0:, respond to the * prompt as follows:

```
*1;0;776L
```

To print the contents of locations 0 through 776 on line printer LP1:, respond to the * and DEVICE? prompts as follows:

```
*2;0;776
DEVICE?LP1:
```

Enter CTRL/O to turn the display on and off. If ODT is displaying a listing and you enter CTRL/O, the display stops; when you enter another CTRL/O, the display resumes.

Printing ASCII Format

If you type quotation marks (") when a location is currently open, ODT prints the ASCII representation of the contents.

If the current location is open as a byte, ODT prints a single ASCII character representing the byte. For example:

```
*1000 101 " A
```

If the current location is open as a word, ODT prints two ASCII characters representing the word. For example:

```
*1032/ 034567 " w9
```

If you type the quotation mark and a location is not currently open, ODT prints the ASCII representation of the previously open location. For example:

```
*1032/ 034567 <LF>
*" w9
```

Printing Radix-50 Format

If you type the percent (%) character when a location is currently open as a word, ODT prints the three-character ASCII representation of the Radix-50 word. For example:

```
*1000/ 034567 % IG1
```

If you type a value before the % character when a location is currently open as a word, ODT interprets the value as the location whose contents you want to display. For example:

```
* 1032/ 000301 1000% IG1
```

If you type the % character and a location is not currently open, ODT prints the three-character ASCII representation of the previously open location. For example:

```
*1032/ 034567
*% IG1
*
```

Relocation Registers

ODT has 16 relocation registers that you can use to specify relative addresses. ODT initially sets the relocation registers to -1 (177777 octal), the highest possible address, to prevent inadvertent errors in address calculation. To set a relocation register, type the relative address, followed by a semicolon and the specification of one of the 16 relocation registers. For example, to set relocation register 0 to 1000, type:

```
*1000;0R
*
```

You can subsequently use the value in relocation register 0 as an offset or a base address in specifying a location. For example, to open location 1032 as a word, type:

```
*0,32/ 000010
```

ODT adds offset 32 to the contents of relocation register 0 to open location 1032.

To reset the contents of all relocation registers to -1 type R in response to the * prompt. For example:

```
*R
*
```

To reset the contents of any one register to -1, specify the register number followed by R. For example:

```
*1R
*
```

When used with disk files, ODT treats registers 0 through 7 differently from registers 10 (octal) through 17 (octal). For registers 0 through 7, the leftmost three digits specify a block number and the rightmost three digits specify a byte location within the block. For example, 000017 designates byte 17 in block 0 of the file. The value 003412 designates byte 412 in block 3 of the file. These registers are helpful in accessing data that is partitioned in 512-byte blocks.

For registers 10 through 17, the value specifies an absolute block address on the disk file. For example, the value 1000 (octal) specifies block 1000 (octal) on the disk file.

You can use relocation registers 10 through 17 to access blocks in large files. That is, files larger than 65K blocks and beyond block number 65,535. Each block contains 1000 (octal), 512 (decimal), bytes. Using the number stored in one of these registers as a base block address, you can then examine the contents of any one of the next 177777 (octal) bytes offset from the base base block address. This allows you to examine the contents of any byte in the next 128 (decimal) blocks, referenced from the base block address. To examine the contents of a byte offset from a base block address block, you must:

1. Store the base block address in one of the relocation registers 10 through 17
2. Specify the relocation register and the byte offset (in octal) in a separate command

For example:

```
*303240;10R
*10R,30/ 000234
```

In the first line, you specify that you want to use block address 303240 (octal); 100,000 (decimal); as the base block address and store this address in relocation register 10.

In the second line, you specify you want to examine the contents of the byte that is offset from the base block address by 30 (octal); 24 (decimal); bytes. Depending on the byte offset you specify, ODT appends zeros to the least significant end of the base block address. Because blocks are comprised of 1000 (octal) bytes, the minimum number of zeros appended is three. In this example, the base block address is changed to 303240000. ODT then adds the 30-byte offset to 303240000 to obtain the byte address of 303240030 and displays the contents, 234 (octal).

You can print the contents of locations based on a fixed offset from the relocation registers. For example:

```
*F
*1;0;3000L
*
```

The first line instructs ODT to calculate relocated addresses for the display. The second line instructs ODT to add offsets from 0 to 3000 to the value of relocation register 0 and print the contents of those locations on line printer 0. You can repeat the procedure for values of relocation registers 1 through 7. The printout contains a listing of the contents of addresses n,0 through n,3000 where n is in the range of 0 to 7 (the relocation registers).

To turn off calculation of relocation addresses for a printout, type 1F. For example:

```
*1F
*
```

Subsequent listing requests print out actual addresses rather than relocated addresses.

Interpretive Address Quantities

ODT stores the value of the last location value displayed on your terminal in a variable named Q. You can request that ODT perform any valid arithmetic procedure on Q. For example, if you want to increase the value of an open location by a certain increment, type:

```
*1342/ 173214 Q+10
*/ 173224
```

The first line opens location 1342 as a word, displays the value 173214, and adds 10 to that value stored in variable Q. In the second line, you display the current value stored in Q (173224) by typing a slash (/). ODT opens location 1342, prints its new contents value, and updates the value in Q.

The period (.) character indicates the currently open or last explicitly open location. It indicates the current address for ODT operations. This is the same address used by the < character (see the section "Returning to an Interrupted Sequence"). In most cases, the period character value is the address used by ODT when you type the /, \, ", or % characters and press LINE FEED. For example, to open as a word a location 16 bytes from the last explicitly open location, type:

```
*.+16/ 012345
```

ODT adds 16 (octal) to the address given by the character, opens the resulting address, and prints its contents.

Error Procedures

If you type an invalid or unrecognized character, ODT prints a ? character, generates a CARRIAGE RETURN and a LINE FEED, and displays the * prompt. For example:

```
*1008?
*
```

ODT indicates that the character 8 is an error. Retype the number correctly.

If ODT finds an error while performing output to a device, it prints the error message %I/O Error, followed by the * prompt. You must correct the device error and type the command again.

Device Transfer: The PIP Program

PIP (Peripheral Interchange Program) is a file transfer and maintenance utility that can perform these operations:

- Copy files from one RSTS/E device to another
- Concatenate, delete, and rename files
- Change protection codes and file attribute data
- Initialize accounts
- List directories

Privilege Required

To run the PIP program, you need the following:

- Read access to any input files you read
- Create/Rename access to any output files you create
- Write access to any output files you write to
- Create/Rename access to any input files you rename
- TUNE privilege if you use the /LOCK or /PRIOR switches
- DEVICE privilege for any restricted devices you access

Running the PIP Program

To run PIP, type:

```
$ RUN $PIP
```

In response, PIP prints an asterisk * prompt to indicate it is ready to accept input.

If your system manager has defined a CCL command for PIP, you can also use it to run the program. If you use the PIP CCL command, you can include PIP commands on the same line.

To terminate PIP, enter CTRL/Z in response to the PIP * prompt:

```
$ PIP
* <CTRL/Z>
$
```

A CTRL/Z directs PIP to complete the current operation and exit in an orderly fashion. If you enter CTRL/C in response to the * prompt, PIP exits but does not complete the current operation. If you enter CTRL/C during a PIP operation, PIP halts the operation and displays the PIP * prompt. Note that you may lose some data.

PIP Command Line Specifications

The PIP command line specifies the actions you want PIP to perform 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	specifies an output file specification.
input	specifies an input file specification.
/sw	specifies one or more switches to direct the action PIP performs on the files.

An output file specification has the format:

```
dev:[proj,prog]filename.type/PROT:n
```

where:

dev:	specifies a device. 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]	specifies a ppn. The default is your current account.

filename — specifies a file name of one to six alphanumeric characters. PIP allows you to specify only one output file specification. If you omit the file name, the default is an * wildcard character; see the section "Wildcard Specifications." For output-only (non-file-structured) devices such as KB:, LP:, and PP:, PIP ignores the file name and type specifications.

.type — specifies a file type of one to three alphanumeric characters. If you omit the type, the default is an * wildcard character see the section "Wildcard Specifications." If you specify only the period (.), the type is null.

/PROT:n — specifies the file protection code see the *RSTS/E System User's Guide*. 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.

- PIP recognizes a file type as executable by comparing it against the file types 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.

- For bits other than the executable bit, PIP uses the user-assigned protection code, if there is one. (You can establish a protection code with the DCL SET PROTECTION/DEFAULT command; see the *RSTS/E System User's Guide*.) 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 (for example a magnetic tape file), PIP uses the default protection code, normally 60.

- o PIP does not allow you to set the privileged bit (128) for an output file unless you have SYSIO privilege. If you try, PIP does not set the bit and does not return an error message.

If you do not type any output specification, PIP directs output to your terminal (KB:).

An input file specification has the format:

```
dev:[proj,prog]filename.type/PROT:n
```

where:

dev:	specifies a device. If you omit the device, the default is DK: as described for output. If the previous input file contains a device specification, that device becomes the default for the current command line until you override it with another specification.
[proj,prog]	specifies a PPN as described for output.
filename	specifies a file name of one- to six-alphanumeric characters. PIP allows you to specify up to six input file specifications, separated by commas. If you omit this specification, the default is an * wildcard character; see the section "Wildcard Specifications." Note that you must specify an input file in a delete operation; see the section "File Deletion Switches." For input-only (non-file-structured) devices such as KB:, PR:, and CR:, the system ignores a file name and type specification.
.type	specifies a file type of one- to three-alphanumeric characters as described for output.
/PROT:n	specifies a protection code. PIP ignores input file protection codes unless the file is the target of a rename operation; see the section "File Rename Switch."

If you do not type any input file specification, PIP displays the following error message:

```
?No input file
```

A PIP switch specification always begins with a slash (/) character; see the section "PIP Switches." When you specify more than one switch in the command line, you must separate them using the / character. When you omit the switch specification, PIP performs a block mode transfer; see the section "Block Switch."

Both the input and output file specifications can contain the RSTS/E file switches /PROTECT, /FILESIZE, /CLUSTERSIZE, /MODE, /RONLY, and /POSITION; see the *RSTS/E Programming Manual*. However, file switches must immediately follow the file specification otherwise, PIP returns an error message. Also, PIP ignores /CLUSTERSIZE on nondisk output file specifications. See the section "PIP Switches" for a description of file specification switches and their format.

If you do not type an input or an output file specification, but press the RETURN key in response to an * prompt, PIP displays its current version and revision level numbers.

Wildcard Specifications

PIP allows you to use the * and ? wildcard characters in input and output file specifications. The use of wildcards saves you typing time and permits you to designate multiple files in a single file specification without the PIP restriction of six maximum input file specifications.

Specify wildcards 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 four- to six-character names, the first four of which are FILE, and a file type of TYP.
FILE??.T??	All files with four- to six-character names, the first four of which are FILE, and any type whose first character is T.

Specification	Result
AB?.TYP	All files with two- to three-character names, the first two of which are AB, and a file type of TYP.
FILE??.*	All files with four- to six-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 PIP substitutes wildcards for the missing elements; see the section "PIP Command Line Specifications." You cannot use default input specifications for delete operations. The defaults PIP applies to missing elements are:

Specification	Default	Meaning
<RET>	*.*	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 ? wildcard character in output file specifications, but if such use causes PIP to try to create a file name with embedded spaces, PIP prints the error message "Illegal output filename", ignore the request, and proceed to the next input file. See the section "File Transfer Operations" for an example.

PIP allows you to specify an * wildcard character in a ppn. However, if you specify wildcard accounts for input files, the files 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 refers to all files in your account, not just those with a protection code of 40.

DCL Command Files in PIP

PIP allows you to specify a DCL command file in response to its * prompt. A DCL command file is a user-created ASCII file that contains all of the information PIP needs to perform an operation. You can also create DCL command files using an editor; see the *RSTS/E Guide to Writing Command Procedures*.

You specify a DCL command file to PIP in the following manner:

```
@dev:[proj,prog]filename.type
```

where:

@	tells PIP that the specified file is a DCL command file.
dev:	specifies a device. The default is the public structure.
[proj,prog]	specifies a PPN. The default is the current account.
filename.type	specifies the file name and type for the DCL command file. You must specify a file name. The default file type is .CMD.

If a line in the file begins with a semicolon (;), PIP interprets the line as a comment.

If the file causes the execution of a /ZE or /IN switch operation, PIP displays confirmation questions at your terminal.

A DCL command file, in addition to containing PIP commands and file specifications, can contain references to other DCL command files. This process is called nesting. The number of files you can nest depends on the amount of memory available to you. If there is not enough memory, PIP displays the following error message:

```
?No buffer space for command file
```

PIP then halts the operation, and returns to the prompt level. A file containing references to nested DCL command files must be resident on a file-structured disk.

Most errors during DCL command file processing cause PIP to return to prompt level. On the following errors, however, PIP displays the following warning message:

```
%No files matching <spec>
```

where:

<spec> indicates the file specification. PIP returns this message during /DE or /LI operations; see the sections "File Deletion Switches," and "Directory Listing Switches."

Pip displays the following message during a /RE operation; see the section "File Rename Switch."

```
?File or account already exists
```

The following command shows the procedure to execute a DCL command file:

```
@ABC
```

PIP reads the file named ABC.CMD in the current account on the public structure. If you specify KBn: following the at sign (@KB3:, for example), PIP reads the specified keyboard for commands. You terminate the keyboard command set with a CTRL/Z.

Because you can use the at sign (@) to designate a logical account assignment (see the *RSTS/E System User's Guide*) and a PIP DCL command, placement of the @ character in the PIP command line is important. When the command line contains an @ character, PIP scans the line for validity as a DCL file command. If PIP finds more than one file specification, wildcard file specifications, or a PIP switch, PIP assumes that the @ character is a logical account and processes the command line accordingly.

PIP Switches: An Overview

You specify the operations that PIP is to perform with one or more switch specifications in the command line; see the section "PIP Command Line Specifications." Always precede a switch with a slash (/) character, and separate multiple switch specifications by slashes.

Table 2-16 lists the PIP switches, their format, and the operations that they perform. See the section "PIP Switches" for a detailed description of their use.

Table 2-16: PIP Switches

Switch	Format	Meaning
Informational Switches		
Help	/HE	PIP prints a text message at your terminal that describes file specifications, switches, and options.
File Transfer Switches		
none	<RET>	See the description of /BL.
Access	/AC	For disk input only, PIP changes the file's last access date to the current date. If you do not specify this switch, PIP preserves the last access date.
Append Extend	/AP or /EX	For disk output only, PIP appends the input file to the output file (extends the output file). If no output file exists, PIP displays a warning message and creates the file.
ASCII	/AS	Formatted ASCII transfer; PIP discards nulls, parity bits, and rubout characters.
Block	/BL	PIP performs a block-by-block transfer with no data translation.

Table 2-16: PIP Switches (Cont.)

Switch	Format	Meaning
File Transfer Switches		
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.
Cluster size	/CL:n	For disk only, PIP sets the output file cluster size to n. To have effect, you must place this switch adjacent to the output file specification. For an explanation of cluster size, see the *RSTS/E Programming Manual*. If you omit this switch from a disk-to-disk transfer, PIP preserves the current cluster size.
Ignore Go	/IG or /GO	PIP ignores ?Data error on device errors.
Mode	/MO:n	Sets the mode (n) in which PIP opens the file. To have effect, you must place this switch adjacent to the file specification; see the *RSTS/E Programming Manual* for an explanation of modes.
New file	/NE	For disk output only, PIP creates a new file with the current date of creation and access.
Retain	/RET	For disk output only, PIP retains creation and access date of the input file.
No attributes	/NOA	PIP transfers the file without writing attributes to the output file.

Table 2-16: PIP Switches (Cont.)

Switch	Format	Meaning
File Transfer Switches		
No supersede	/NOS /NOS:NOWARN /NOS:Q /NOS:IN	PIP displays an error message and ignores the operation if an output file already exists. If you include the :NOWARN option, PIP does not display 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. DIGITAL recommends NOS to avoid conflict with protection code option.
Run-time system	/RTS:name	Sets the output file's run-time system name to that specified in name. If you omit the output file specification, PIP renames the input file's run-time system; see the section "Run-Time System Name Switch."
Update	/UP	PIP opens a preexisting disk file and overwrites 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, PIP translates RMS format to formatted ASCII or formatted binary. For output disk files, PIP translates formatted ASCII or formatted binary to RMS variable length records.

Table 2-16: PIP Switches (Cont.)

Switch	Format	Meaning
Files with Attributes, Translation Switches and Options		
		For nondisk transfers, PIP's translation depends on current format; see the section "File Operations Involving Attributes." If you do not specify the /RMS switch for disk files, PIP performs no translation. You cannot apply the switch 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, PIP translates FORTRAN carriage control to formatted ASCII. For nondisk output files, translation is automatic.
	/RMS:IM	For input files only, PIP performs the same as /RMS but does not transfer attributes but does remove RMS variable length header information.
	/RMS:PRN	For input disk files, PIP translates RMS print files to formatted ASCII.
Date Switches		
After	/AF:dd-mmm-yy	PIP includes only files that were created after, but not on, the given date.

Table 2-16: PIP Switches (Cont.)

Switch	Format	Meaning
Date Switches		
Before	/BE:dd-mmm-yy	PIP includes only files that were created before, but not on, the given date.
On	/ON:dd-mmm-yy	PIP includes only files that were created on the given date.
Since	/SIN:dd-mmm-yy	PIP includes only files that were created on or after the given date.
Today	/TO	PIP includes only files that were created on the current date.
Until	/UN:dd-mmm-yy	PIP includes only files that were created on or before the given date.
Date of last access	/DLA	For disk only, PIP uses the date of last access in file operations.
Creation	/CRE	PIP uses the date of creation in file operations.
File Operation Switches - General		
Halt	/HA	PIP halts a magnetic tape wildcard search as soon as it detects a file that does not match.
Inspect Query	/IN or /QU	PIP displays 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. Enter CTRL/Z to end the display and operation.

Table 2-16: PIP Switches (Cont.)

Switch	Format	Meaning
File Operation Switches - General		
	/IN:S	PIP prints file name, type, filesize, protection code, creation date, and last access date.
Log Watch	/LO or /WA	PIP displays a report on all actions taken during execution. PIP prints the report at your terminal.
No log	/NOLO	PIP displays nothing for actions taken during execution.
No rewind	/NO or /RW:NO	PIP does not rewind the magnetic tape before an input file search.
Version Identify	/VE or /ID	PIP displays the program's current version number.
File Operation Switches - Deletion		
Delete	/DE	For disk and DECtape only, PIP deletes the specified file.
	/DE:NO	PIP does not display an error message if the file to be deleted does not exist.
Erase Wipe out	/ER or /WIPE or /WO	For disk, PIP overwrites the file with zeros prior to deletion. These switches are synonymous; you must use one of them in conjunction with the /DE switch.
File Operation Switch - Rename		
Rename	/RE:option	For disk and DECtape only, PIP renames the input file to that of the output file.

Table 2-16: PIP Switches (Cont.)

Switch	Format	Meaning
File Operation Switches - Listing		
Brief Fast	/BR or /F	PIP displays a brief directory listing.
Directory List Slow	/DI or /LI or /S	PIP displays a full directory listing. The listing switches accept option specifications that modify the directory listing; see the section "Directory Listing Switches."
File Operation Switches - Zeroing		
Zero	/ZE	PIP zeros (initializes) the directory of an account or device or initializes the labels on a magnetic tape.
Density	/ZE/DEN:n	For magnetic tape only, PIP sets the tape density before the zero operation.
Parity	/ZE/PAR:ODD /ZE/PAR:EVEN	For magnetic tape only, PIP sets the parity of the tape before the zero operation.
Privileged-Only Switches		
Lock	/LOCK	The system locks PIP in memory for the duration of the current operation. Note that you cannot abbreviate /LOCK.
Priority	/PRIOR	The system runs PIP at special priority for the duration of the current operation. Note that you cannot abbreviate /PRIOR.

File Transfer Operations

File transfer operations have the following command format:

output/sw=input/sw

where:

output — specifies a single file specification.

input — specifies from one to six file specifications. If you specify one file specification, PIP copies that file to the output file. If you specify more than one file specification, PIP concatenates copies of the input files and writes them to the output file. PIP does not alter input files.

/sw — specifies one or more switches that determine how PIP modifies the file transfer. If you do not specify a switch when performing disk-to-disk transfers, PIP performs a block mode (/BL) transfer by default. In a /BL transfer, PIP copies the input file data, block-by-block, and any input file attributes into the output file. PIP continues the transfer until the last of the input file data has been copied.

If you do not specify a switch on disk to nondisk file transfers, PIP performs a file transfer with format translation by default; see the section "File Operations Involving Attributes." When transferring RMS files with attributes to non-disk-structured devices, PIP translates the file to ASCII stream or formatted binary, depending on the file attributes. 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 performs a /BL 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 American National Standard Institute (ANSI) 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.

Use the following procedures to make sure PIP transfers information correctly.

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 the section "Initializing Directories". For example:

   ```
   PIP MT0:TAPIT/ZERO/DEN:800/PAR:ODD
   ```

   In the example command line, MT0: specifies the device, TAPIT specifies the volume ID, and the DEN and PAR switches specify a density of 800 bpi and odd parity. If you omit the volume ID when initializing an ANSI magnetic tape, PIP displays the following error message:

   ```
   ?Illegal file name -- file mmx:.
   ```

3. Use the /BL switch; see the section "Block Switch", to copy the files to a tape. For example:

   ```
   PIP MT0:FORMS.DAT=FORMS.DAT/BL
   ```

   PIP creates the output file in U (undefined) format and retains all attribute information in the file header label. Use this type of transfer only 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 the section "File Operations Involving Attributes." For example:

   ```
   PIP MM0:REPORT.DAT/RMS:FA=REPORT.DAT
   ```

   To transfer FCS (RSX File Control Services) or RMS files with attributes, you do not have to specify any switch. However, you can copy only RMS sequential files. Therefore, to transfer RMS relative and indexed files to ANSI labeled tape, you must first use the RMS conversion utility (RMSCNV) to convert them to sequential organization. PIP translates sequential files with attributes to their appropriate ANSI record format; see the section "File Operations Involving Attributes."

Tape to disk transfer:

1. Mount the input tape and use the ASSIGN command to specify ANSI label format. 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 file attributes from tape and restores the file to its former format.

   If the file you are transferring 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 is an RMS variable length record file with implied carriage return.

   You can use the /CL switch to copy files with the correct cluster size; see the section "Cluster Size Switch." Use RMSCNV to convert files (originally converted from RMS relative or indexed files to sequential files) back to their original format. If the file resides 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. See the *RSTS/E RMS-11 User's Guide* for information about these RMS utility programs.

3. Dismount the tape to release the assignment made in step 1. In addition, you can include the /UNLOAD switch. For example:

   ```
   DISMOUNT MT0:/UNLOAD
   ```

This manual does not discuss RSTS/E disk-to-RSTS/E disk transfers because 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 stores files on the public structure so you can access them efficiently. Copying within the public structure does not improve this efficiency and can cause problems if you use wildcards. An exception condition such as an I/O error or CTRL/C may cause PIP to delete both the old and the new copies of the file you are transferring.

If you find that it is necessary to transfer files between public disks in the same account, there are three safe ways to transfer the files:

- o Mount either the source or the destination disk as a private disk. This prevents RSTS/E from deleting the source file when the destination file is opened. After you confirm that the operation is successful, delete the source files, dismount the private disk and remount it as a public disk.

- o Assign a temporary name to each file you want to transfer, and then delete the old files after a successful transfer operation. For instance, to move all *.ODL files from DB1: to DB0:, you can 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:]
```

- o Use two transfers; go first to tape, or to a private disk, or to another account.

If you specify a non-file-structured device in the input file specification, PIP ignores the file name and type even if you use wildcards.

If you transfer data to the line printer and the printer goes off-line, PIP displays the following message:

```
Line printer hung - put online to continue or type CTRL/C
```

Following the message, PIP rings the terminal bell every ten seconds until you put the printer on line or enter CTRL/C. If you enter CTRL/C, PIP returns to the * prompt.

Wildcard Usage in File Transfer Operations

Input and output file specifications can contain wildcard characters; see the section "Wildcard Specifications." For example:

```
DL1:*.*/CL:4=[2,2]*.MAC
```

PIP copies all files with a .MAC type in account [2,2] on the public structure to your account on RL01 unit 1. PIP preserves protection codes and creates all output files with a cluster size of 4 (see the section "Cluster Size Switch"). 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
```

The * wildcards are legal for output file accounts and for input file accounts if the input file resides on disk or magnetic tape.

You can also use the ? wildcard in input or output file specifications. If a corresponding character exists in the input file specification, PIP supplies it for the ? character in output file specifications. For example, suppose files MYFIL1.TXT and MYFIL2.TXT reside in your account on the public disk structure and you use the following command:

```
HS????.TXT=MYFIL1.TXT, MYFIL2.TXT
```

PIP creates files HSFIL1.TXT and HSFIL2.TXT consisting of files MYFIL1.TXT and MYFIL2.TXT, respectively.

Suppose files M1.TXT and M2.TXT reside in your account on the public disk structure and you use the following command:

```
HS????.TXT=M1.TXT,M2.TXT
```

PIP performs two copy operations. First PIP copies file M1.TXT to HS.TXT, and then it copies file M2.TXT to HS.TXT. The final result is one file named HS.TXT consisting of the contents of file M2.TXT.

If the output specification is such that PIP tries to create a file name with embedded spaces, PIP displays the error message "?Illegal output filename" ignores the request, and proceeds to the next input file, if any. For example:

```
????M?.TXT=M1.TXT,M2.TXT
```

PIP displays two error messages. First, PIP tries to create a file named M1 M .TXT. Because this file name contains embedded spaces, PIP displays the error message "?Illegal output filename". PIP then proceeds to the next input file, attempting to create a file named M2 M .TXT. Again, PIP displays the error message "?Illegal output filename".

The Help Switch

The /HE switch directs PIP to display information about input and output file specifications, switches, and options on your terminal.

The Access Switch

The /AC switch applies 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 before copying the file to the output file. If you do not specify this switch, PIP does not change the file's date of last access and preserves it during the transfer. Even if the input disk is initialized for an access update on date of last write, the /AC switch causes PIP to change the access date.

The Append and Extend Switches

The /AP and /EX switches 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 the data 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.DAT,C.DAT/AP
```

In this example, A.DAT contains its original data followed by the data transferred from B.DAT and C.DAT. DIGITAL does not recommend using these switches if the output file has attributes and was created under control of the RSX File Control Services or RMS, because PIP does not update the end-of-file attribute.

If the specified output file does not exist, PIP returns the error message ?Can't find file or account, and creates the file. You cannot append a file to itself. If you try, PIP returns the error message ?Protection violation.

The ASCII Switch

The /AS switch applies to output files that you want 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 the previous section). 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.

Do not use the /AS switch to transfer files that contain 8-bit binary data. The binary data uses the parity bit, which PIP discards. The bytes in a virtual array or block I/O file can take any pattern of eight bits, one of which can be CTRL/Z. Because PIP terminates the transfer when it finds CTRL/Z, some data can be lost. To transfer such a file, use the /BL (Block) switch (see the next section).

The Block Switch

The /BL switch directs PIP to perform a block-by-block image copy of the input file to the output file. PIP performs no translation on the data and preserves all file attributes. Block transfer is the default file transfer operation.

If you are transferring an RMS sequential file to a nondisk device and want to preserve the RMS format, specify the /BL switch. (You can also use the /RMS switch; see the section "File Operations Involving Attributes." If you do not include either the /RMS or /BL switch in the command line, PIP automatically converts the RMS format to formatted ASCII or formatted binary. Also use the /BL switch to correctly transfer RMS relative and indexed files between RSTS/E systems.

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.

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, it creates it noncontiguously and displays the warning message % File created noncontiguous.

When PIP performs a block mode transfer (either by default or because you include the /BL switch) from a nondisk device to a disk, it examines the output file type. If the the file type indicates an executable image (.BAC, .TSK, .SAV, and so on.), 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 you invoke PIP. If PIP detects conflicting file types (for example RSX uses .TSK) PIP assigns the name of the first run-time system on its list whose file type matches the output file type (see the section "Run-Time System Name Switch").

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, PIP transfers each DECtape block to a disk block and fills the excess bytes in the last block with null characters. For a disk to DECtape transfer, PIP places the two extra bytes in a new block. PIP does not pad excess bytes with nulls until it reaches the last output block.

The Block Size Switch

The /BSIZE switch specifies a block size for output to magnetic tape. If you do not use this switch, PIP transfers data to an output magnetic tape using a default block size of 512 bytes. You can use the /BSIZE:n switch, where n is an even integer in the range of 18 to 4096 bytes, to create a magnetic tape file whose block size is larger or smaller than the 512-byte default. If you specify an illegal value for n, PIP displays the error message ?Bad block size.

Use of the /BSIZE switch has the following restrictions:

- Use the switch for magnetic tape only; PIP ignores its use on disks.

- The switch applies only to output on magnetic tape; PIP automatically handles input magnetic tape with block sizes other than 512 bytes.

- If the output tape is in ANSI format and is intended for interchange on another operating system, the block size specification must be an even integer between 18 and 2048 bytes.

- If the output tape is intended for use on the RT-11 operating system, the block size must be 512 bytes.

- Using the /BSIZE switch can increase the amount of buffer space PIP uses. Thus, when reading or writing block sizes larger than 512, invoke PIP with the CCL /SIZE:n switch or, if under the RT11 Run-Time System, with the SIZE n immediate mode command.

The Cluster Size Switch

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. You can specify the value of n in two ways:

- Specify a power of 2 that is greater than or equal to the pack cluster size of the disk involved in the transfer, and less than or equal to 256.
- Specify a 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.

If you use this switch, place it immediately after the output file name on the PIP command line, unless you also include the /MO switch. In the latter case, place the /MO switch immediately after the output file name, followed by the /CL switch (see the section "Mode Switch"). If you do not include the /CL switch in the command line, PIP preserves the cluster size of the input file during the transfer. PIP ignores the switch for nondisk devices. For a complete discussion of cluster size, see the *RSTS/E Programming Manual.*

The Go or Ignore Switches

Specify either the /GO or /IG switch if you want PIP to ignore the error message ?Data error on device or any magnetic tape record length errors. If this error occurs in a record other than a label record, and you include one of these switches, PIP ignores the error and continues the operation. With these switches, you can use PIP to partially salvage files from damaged or compromised media. However, PIP cannot retrieve files whose label records are unreadable.

The Mode Switch

RSTS/E allows you to specify modes in OPEN statements that set special characteristics for devices. You can use the /MO:n switch in the PIP command line to specify the mode PIP uses to open the files. When you include this switch, place it immediately after the output file name. See the *RSTS/E Programming Manual* for information about mode values and how to use them.

The New File and Retain Switches

The /NE switch, specified for disk output files, directs PIP to create a new file and use the current date for both the creation date and last access date. The /RET switch, also specified for disk output files, directs PIP to retain the creation date and last access date of the input file. The /RET switch is the default, unless your system manager has installed the optional feature patch that makes the /NE switch the default.

The No Attributes Switch

The /NOA switch disables the transfer of disk file attributes during PIP copy procedures. If the input file contains attribute information that you do not want to copy to the output file, include the /NOA switch. If you do not include this switch, PIP automatically transfers attributes.

The No Supersede Switch

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 deletion and overwriting. The action of this switch is independent of the file protection code, which can also prohibit write access.

When you specify /NOS and the output file does not exist, PIP creates the file and transfers the data. PIP does not display an error message.

When you specify /NOS and the output file does exist, PIP displays the error message ?Name or account already exists 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 message.

If you use the QUERY or INSPECT option (/NOS:Q or /NOS:IN), PIP displays the message "OK to replace existing file?" if the output file already exists. Type Y, YE, or YES to replace the file. Type N, NO, or press RETURN to cancel the data transfer.

The Run-Time System Name Switch

For a file structured disk transfer, use the /RTS switch 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, PIP transfers a run-time system name to the output file based on the following criteria:

- If the input file resides on a file-structured disk, PIP transfers the input file's run-time system name to the output file.
- If the input file does not reside on a file-structured disk, PIP searches for a run-time system whose executable file type matches the output file type. PIP assigns a name to the output file based on the first such run-time system it finds. PIP also assigns an executable protection code to the output file.
- If neither the first nor second case applies, PIP assigns the run-time system name under which it is running to the output file.

The Update Switch

Use the /UP switch for output disk files. It directs PIP to attempt an update in existing files of the same name before it creates a new file. When you include the /UP switch, 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 cannot locate a matching output file, it creates a new file.

If PIP finds a matching output file that is larger than the corresponding input file, PIP writes the input file data to the output file without disturbing the output file excess data. For example, a 34-block input file changes only the first 34 blocks of a 48-block output file. If you also specify the /LO or /WA switch (see the section "File Operation Switches"), PIP reports that the input file was "Copied into prefix of" the output file.

If PIP finds a matching output file that is shorter than the corresponding input file, PIP extends the output file. For example, if the input file is 48 blocks and the output file is 34 blocks, PIP extends the output file to 48 blocks and replaces the output file data with the input file data.

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 BASIC-PLUS files do not have attributes.

In addition, RMS files and non-RMS files have different formats. RMS files can have a 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, 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.

You can direct PIP to use the /RMS switch and its options to translate from one format to another as it transfers input disk file data to output disk files.

Use the /RMS switch only on input or output files that are resident on disk devices, or on ANSI or DOS labeled magnetic tape. The switch forces PIP to translate the file data format when transferring the file. You cannot use this switch for both input and output files in the same command line.

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.

When you specify the /RMS switch, PIP determines whether or not to translate between RMS format and formatted ASCII or formatted binary. However, PIP allows you to include one of five options with the switch to specify a particular translation. The available options are:

Option	Meaning
:FA	For input disk files, translate from formatted ASCII. For output disk files, translate to formatted ASCII.
:FB	For input disk files, translate from formatted binary. For output disk files, translate to formatted binary.
:FTN	For input disk files, translate RMS FORTRAN carriage control to formatted ASCII.

Option	Meaning
:IM	For input disk files, translate from RMS format to formatted ASCII or formatted binary (same as /RMS 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.

To translate a Task Builder .MAP file to formatted ASCII, you must use the /RMS:FA switch.

Use of the /RMS switch options to transfer files with attributes has the following restrictions:

- PIP cannot concatenate two or more RMS data files. An attempt to do so results in a file with invalid attributes. All the data in the file is not accessible. To concatenate RMS files, use the RMS utility RMSCNV (see the *RSTS/E RMS-11 User's Guide*).
- If a file contains variable length records or fixed length records with RMS FORTRAN carriage control, PIP can only translate the file to stream (formatted ASCII) records.
- If a file is an object module, PIP can only translate the file to formatted binary.
- If you do not specify the /RMS switch on a transfer of disk files to DOS labeled magnetic tape, the following default transfers occur:
  - If the input RMS file contains implied or RMS FORTRAN carriage control, or if it is a print format file, PIP creates an output file in stream (formatted ASCII) format.
  - In all other cases, PIP creates a formatted binary file.
- If you do not specify the /RMS switch on a transfer of disk files to ANSI labeled magnetic tape, the following default transfers occur:
  - If the input RMS file contains fixed length records, PIP creates an output magnetic tape file in ANSI F record format (fixed length format).

- If the input RMS file contains variable length records, PIP creates an output magnetic tape in ANSI D record format (variable length format).

- If the input file contains no attributes, PIP creates an output magnetic tape file in U (undefined) record format. Note that you may not be able to transfer tapes with U format to non-RSTS/E systems.

o If you do not specify the /RMS switch on an ANSI magnetic tape to disk transfer, the following default transfers occur:

  - If the input magnetic tape file is in ANSI F record format, PIP creates an output disk file with RMS fixed length records.

  - If the input magnetic tape file is in ANSI D record format, PIP creates an output disk file with RMS variable length records.

  - 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 the section "Block Switch").

Date Related Switches

Use the date switches to specify the inclusion or exclusion of certain files in a PIP transfer based on the creation date or date of last access. These switches are especially useful for modifying wildcard file specifications.

By default, PIP transfers files based on the creation date. However, the system manager can install an optional feature patch that allows PIP to use the date of last access as the default.

You can specify the date of last access (/DLA) or the creation date (/CRE) switch in the command line to override the installed default. The /DLA switch directs PIP to use the date of last access; the /CRE switch the date of creation. The switch you use determines whether the /TO, /ON, /AF, /BE, /SIN, and /UN switches use the creation date or the date of last access. These switches do not require an argument.

The today switch (/TO) directs PIP to operate only on those files with a date that matches the current date. This switch does not require an argument.

The /ON:date switch directs PIP to operate on those files with a date that matches the switch date. This switch requires a single argument that specifies a date in the format:

/ON:dd-mmm-yy

where:

dd specifies the day of the month.

mm specifies the first three letters of the month.

yy specifies the last two digits of the year. The default is the current year.

For example:

/ON:13-Jun-85

When used in combination, the After (/AF), Before (/BE), Since (/SIN), and Until (/UN) switches, allow you to specify a range of dates. Only one switch is necessary in the PIP command line; however, you can use two switches to specify an upper boundary and a lower boundary. When you specify a range, PIP processes only those files whose date falls within the range and ignores those files whose date is outside the range. Because of a conflict between the /SIZE and /SINCE switches, the minimum abbreviation for the /SINCE switch is /SIN.

To specify a date with these switches, use the same format as the /ON switch.

The following examples show the use of these switches:

Switches	**Lower Boundary**	**Upper Boundary**
/SIN:09-APR-85 /UN:01-MAY-85	09-APR-85	01-MAY-85
/AF:15-JAN-85 /UN:01-MAY-85	16-JAN-85	01-MAY-85
/SIN:13-JUN-85 /BE:20-JUN-85	13-JUN-85	20-JUN-85
/AF:15-DEC-85 /BE:09-APR-85	16-DEC-85	08-APR-85

The File Operation Switches

You can apply the following file operation switches to the four major operations that PIP performs: copy, delete, rename, and directory listing. Use these switches to improve the performance of PIP operations and as auxiliaries to those operations.

You can specify the /HA switch when PIP conducts wildcard searches on magnetic tape devices. When you include this switch, PIP stops processing when it finds 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 use the /IN switch when the PIP operation involves wildcard specifications. You can include the switch at any point in the command line. When present, it directs PIP to display the file name of each file that matches the wildcard specification. PIP displays the file names one at a time and allows you to respond with one of the following:

Response	Meaning
Y (for Yes)	Directs PIP to execute the operation on this file and search for the next matching file.
Any other response	Directs PIP to skip this file and search for the next matching file.
CTRL/Z	Directs PIP to terminate processing of the current wildcard specification without processing this file.

When you include the /IN switch, it applies to every input file wildcard specification.

The /IN switch accepts an option (/IN:S) that directs PIP to print the file name, file type, file size, protection code, creation date, and date of last access of each file that matches the wildcard specifications.

The /LO or /WA switches are synonymous. You can include them anywhere in the command line. When you specify /LO or /WA, PIP displays a report of all actions taken during execution. Note that the system manager can install an optional feature patch that causes PIP to log all actions by default.

Use the /NOL switch to suppress the display of the execution report where logging is enabled as the system default.

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 switch anywhere in the command line to direct PIP to suppress the automatic tape rewind. When you specify one of these switches, PIP begins its file search at the current tape position.

You can specify the /VE or /ID switch anywhere in the command line to direct PIP to print information about the version and current configuration on the terminal.

The File Deletion Switches

When you specify the /DE switch in a command line input file specification, PIP deletes all files that match the specification. You can apply the switch only to a disk or DECtape input file specification. If you do not specify a file name or a type, PIP prints the following error message:

```
?Filename and type must be specified
```

You can use wildcard characters in file deletion operations. If you do specify wildcards, you can also include the /IN switch (see the previous section) in the command line to prompt for confirmation of all file deletions.

The /DE switch applies to all input file specifications on the command line in which it appears. Therefore, do not attempt to specify any other PIP operation in a command line that contains the /DE switch.

You can modify the /DE switch with the /NO option (/DE:NO). When you include this option, PIP does not display an error message if the file you want to delete does not exist.

If the file you want to delete has the privileged (128) bit set in its protection code, PIP writes zeros in the file, renames it to a nonprivileged code, and deletes it. When PIP deletes a file with the privileged bit from the system by means of the UNSAVE command or KILL statement, the File Processor performs the same actions as the PIP delete operation. If you want such a delete operation, use PIP because it is faster than the File Processor, which is subject to greater system demand.

You can specify the /ER, /WO, or /WIPE switch with the /DE switch. When you include one of these switches, PIP writes zeros in the file before deleting the file. As with the /DE switch, these switches apply to every input file specification on the command line.

The File Rename Switch

When you apply the /RE switch to a disk or DECtape input file specification, PIP changes the input file name to that of the output file specification. During the /RE operation, PIP does not transfer any data.

Use the following format when renaming files with the /RE switch:

```
newname.type/PROT:n=oldname.type/RE
```

PIP renames file oldname.type to newname.type and changes its protection code according to the value you assign to n. The data in file newname remains unchanged.

You can use wildcard specifications with the /RE switch to perform multiple file renames. PIP requires that you specify only those elements of the output file specification that you want to change. If you do not include an input file specification, PIP renames all files in the current account on the public structure.

The following example shows the use of the /RE switch:

```
PIP *.C??=STAT?.B?S/RE
```

PIP changes the file type of every file whose file name begins with STAT and whose type begins with B and ends with S. In each case, PIP changes the first character of the type to C.

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 message ?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 message ?Name or account now exists, skips that operation, and continues processing. You can use the NOWARN option (/RE:NOWARN) to suppress the display of the ?Name or account now exists error message.

The Directory Listing Switches

PIP provides five switches and several options that allow you to display a directory listing. This listing is similar to the one you can obtain using the DIRECT system program (see the DIRECT utility, section "DIRECT Switches.")

The /DI and /LI switches direct PIP to display a full directory listing and allow you to modify the listing. Table 2-17 lists the available options. Specify options in the following format:

/DI:option

or

/LI:option

The /BR and /F switches do not accept an option specification. When you include these switches, PIP displays an abbreviated directory listing that includes only the file names and types. The /S switch directs PIP to display a full directory listing that is similar to the slow listing obtained with the DIRECT system program; (see the section "DIRECT Switches").

The /DI, /LI, /S, /F, and /BR switches allow PIP to display the directory listing of any file-structured device. When you include one of these switches in a file specification, which can contain wildcards, PIP displays only those files that match that specification. PIP displays the listing on your terminal by default; however, you can output the listing to another device or file by including an output specification.

Table 2-17 lists the options that you can specify with the /DI or /LI switches.

Table 2-17: PIP Directory Listing Options

Option	Format	Meaning
Allocated	:AL	List file name, type, and the number of blocks allocated to the file.
Cluster size	:CL	List file name, type, and file cluster size.
Date	:DA	List file name, type, and file creation date.
Directory	:DI	List file name, type, size, protection code, creation date, and column headers.
Extended	:EX	List file name and type same as :TY (provided for compatibility with previous versions of RSTS/E).

Table 2-17: PIP Directory Listing Options (Cont.)

Option	Format	Meaning
Full	:FU	List heading, file name, type, 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.
Header	:HD	Include column headers in the listing.
Last	:LA	List file name, type, and date of last access or date of last write (depending on disk initialization).
Name	:NA	List file name.
Octal attribute	:OA	List file name, type, and octal representation of file attributes.
Protection	:PR	List file name, type, and protection code.
Run-time	:RT	List file name, type, and associated run-time system.
Slow	:S	List all information.
Symbolic attribute Attribute	:SA or :AT	List file name, type, symbolic file attributes (see the "File Attributes" section in the DIRECT utility)and caching status. Caching is indicated by one of the following entries: o CACHE:ON:RAN; file will be automatically cached randomly when open. o CACHE:ON:SEQ; file will be automatically cached sequentially when open. o CACHE:OFF:SEQ; file will not be automatically cached, but if cached, it will be cached sequentially.

Table 2-17: PIP Directory Listing Options (Cont.)

Option	Format	Meaning
Size	:SI or :SZ	List file name, type, and file size.
Summary	:SU	List only summary data to include number of designated files and total number of blocks occupied by designated files
Time	:TI	List file name, type, and date and time of creation.
Type	:TY	List file name and type.
Wide	:W	List file name and type across the width of the page or screen.
Wide	:WI	List file name only across the width of the page or screen.

The :OA, :SA, :AT, and :S options cause PIP to display file attributes in octal or symbolic representation. See the DIRECT utility, section "File Attributes," for a description of the format and meaning of these attribute representations.

The Initializing Directories Switch

You can include the /ZE switch in the PIP command line to direct PIP to initialize (zero) a device or account directory. When you specify this switch, PIP deletes all of the files stored in a specified account on disk or, if you specify a magnetic tape or DECtape, initializes the volume mounted on the specified drive.

To use the /ZE switch, specify a command line of the following format:

```
dev:[proj,prog]/ZE
```

where:

dev:	specifies a device. You can specify only one device. The default is the public disk structure. To initialize an ANSI-labeled magnetic tape, specify a device and a file name that represents the volume label.
[proj,prog]	specifies a ppn. The default is the current account.
/ZE	is the initializing switch.

When PIP detects the /ZE switch in its command line, it prints the following prompt:

```
REALLY ZERO dev:[proj,prog] ?
```

Type Y in response to the prompt to confirm that you want PIP to initialize the device or account. Any other response aborts the operation and returns you to the PIP command response level.

You can also use the /ZE switch to specify a change in the density or parity of a magnetic tape prior to initialization. System-wide defaults for magnetic tape density, parity, and label are set by the system manager during system initialization. If you want the system to accept a tape with nondefault characteristics, you should ASSIGN the tape prior to setting those characteristics (with MOUNT or PIP). 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

/ZE/PAR:parity
```

The /ZE/DEN:n switch combination directs PIP to set the density of the tape to the value specified by n before it initializes the tape. You can specify these densities:

800	bpi for 9-track tape.
1600	bpi for 9-track tape.

The /ZE/PAR:parity switch combination directs PIP to set the parity of the tape to the type specified before it initializes the tape. You can specify these parities:

/ZE/PAR:ODD	For odd parity.
/ZE/PAR:EVEN	For even parity.

A 1600 bpi tape can only support odd parity. Also, unless you process the tape on a system that requires even parity, DIGITAL recommends that you specify odd parity for all output tapes.

DIGITAL recommends that you use an initialize operation, which contains a volume label specification, for all new tapes.

The /LOCK and /PRIOR Switches

To use the /LOCK and /PRIOR switches, you must have the TUNE privilege.

When you specify the /LOCK switch in the command line, the system locks PIP into memory for the duration of the current operation.

When you specify /PRIOR in the command line, the system runs PIP at a special priority for the duration of the current operation. Normally PIP runs at the current user's default priority. When you include /PRIOR, the system boosts the priority of PIP by four.

Multivolume ANSI Magnetic Tape File Transfer

You can use PIP to transfer files from any input device to one or more ANSI labeled magnetic tapes to create a volume set. You can also begin a file on one volume of the set and end the file on another, assign identification labels to the volumes of the set, and transfer the contents of the volume set to any output device. You cannot use PIP to transfer multivolume magnetic tapes in DOS format. DOS does not permit multivolume transfers because it does not allow file segmentation across tape volumes.

To perform a multivolume transfer, you must first initialize the output magnetic tapes with the /ZE switch (see the section "Initializing Directories Switch"). The first tape of the output set may contain data but each succeeding tape must be data free. PIP allows you to specify volume identifications for each output tape. This enables you to create an identifiable volume set containing the transferred data. The system manager can install an optional feature patch that causes PIP to require a volume ID specification.

When you initiate a multivolume transfer, PIP displays a series of messages that notifies you of the end of the output tape. PIP then prompts you to specify the device name and unit number of the second output tape (the second volume). PIP repeats this process for each end-of-tape until it has transferred all of the specified input. After PIP reaches the end of an output tape volume, it rewinds the tape and takes it off line.

To make sure it retrieves files in the correct order, PIP assigns section and sequence numbers to the files it transfers. PIP checks these numbers when it transfers files from the tape volumes. The numbers are internal and are not user specified.

You can use file transfer switches on multivolume transfers. The directory switches return directory listings only on a single volume. For a file segmented across volumes, you must list a directory of all volumes to show the true state of the file.

The following example initializes three tapes in preparation for a file transfer, and then uses PIP to transfer the contents of a user account to three magnetic tape volumes:

```
$ RUN $PIP
* MM2:TAPE3/ZE
* MM1:TAPE2/ZE
* MM0:TAPE1/ZE
* 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

                                    ?MM2:TAPE3
*
```

In this example, you specify a 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 prompts for the next volume. In response to the prompt, you type the device name and unit number (MM1:) and the optional volume ID (TAPE2). PIP checks the output volume ID, if specified, and confirms that the tape contains ANSI labels and has been initialized. If the ID is valid and the tape is properly initialized and free of data, PIP continues with the transfer. PIP repeats this process each time it reaches the end of a tape, until it has transferred all of the specified files. When the transfer is complete, PIP displays the * prompt.

The following example uses PIP to transfer 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

                                  ?MM2:TAPE3


*
```

In this example, you specify a 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 prompts for the next volume. In response to the prompt, you type the device name and unit number (MM1:) and the optional volume ID (TAPE2). PIP checks the input volume ID, if specified, and makes sure that the file is in the correct sequence by checking 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 file is correct, and the tape has ANSI labels, PIP continues with the transfer. PIP repeats this process each time it reaches the end of a tape until all of the files on the input tape volumes are transferred.

When the transfer is complete, PIP displays the * prompt.

If you respond to the volume prompt (?) with a CTRL/Z, PIP interrupts the transfer and terminates.

PIP Error Messages

PIP can return two types of error messages:

- o RSTS/E system error messages (for example, ?Can't find file or account). See the *RSTS/E System User's Guide* for descriptions of the RSTS/E errors and the proper recovery actions.

- o PIP program error messages indicate problems within PIP itself and the only recovery is to try the operation again (the error may be due to a transient condition). If the error continues, notify your system manager.

  ?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 try the operation again (the error may be due to a transient condition). If the error continues, notify your system manager.

Formatting a Post-Mortem Dump: The PMDUMP Program

The system library program, PMDUMP, formats the contents of a post-mortem dump into 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. A system programmer or software specialist can use the information contained in the dump to view the actual state of the program at the time it aborted.

When you run PMDUMP, the program performs the following operations and writes the information to a user-specified output file:

- Formats the binary dump and produces a human-readable dump of the contents of low-core memory
- Produces an octal dump of the user's task

When the program completes its operation, it returns control to RSTS/E command level.

Privilege Required

To run the PMDUMP program, you need the following:

- Read access to the input file
- Create/Rename access to the output file

When to Use PMDUMP

Post-mortem dumps are used to determine the state of a task at the time the task aborted. Post-mortem dumps do not return an online snapshot of the state of a job. The following conditions must be true for PMDUMP to be successful in formatting the dump:

- The run-time system that controls your job must be RSX or RSX-based.
- The task to be dumped must have already aborted (terminated abnormally).
- You must specify the request for a post-mortem dump when you input the program to the Task Builder; that is, you must use the /PM switch as described in the *RSTS/E Task Builder Reference Manual*.

Running the PMDUMP Program

To run PMDUMP, type:

```
$ RUN AUXLIB$:PMDUMP
```

The program displays a header line followed by a number sign (#) prompt, which indicates it is ready to accept command input.

In response to the prompt, type one of the following:

Command	Result
<RET> or <LF>	Accepts a complete default.
outfil.typ=infil.typ	Specifies the output and input files used by PMDUMP.
CTRL/Z	Terminates PMDUMP.

If you do not specify an input file, the program defaults the file name to the following:

```
PMDnnn.PMD
```

In the default file name, nnn refers to the current job number. Note that this file name 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 file name to that specified or defaulted for input, and the type to .LST.

The program defaults for all elements of the file specification are as follows:

Device	**SY:**
account	The current account
file name	The input file name
type	.LST for output, .PMD for input

If you press RETURN in response to the prompt, the program applies the following defaults:

- The program places the files in the current account on the public disk structure (SY:)
- The input file name is PMDnnn.PMD
- The output file name is PMDnnn.LST

Contents of the Post-Mortem Dump

The following description of the PMDUMP listing is keyed to the example that follows:

Item	Description
1	The name and account of the task being dumped and the date and time that the dump was generated. This information appears 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 either GEN or the name you specified in the Task Builder PAR option.
3	The size of the task in octal and decimal words. PMDUMP gives the task size at the time it was initially loaded and also at the time of the dump.
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 PPN, default protection code, and logical name table entries assigned by the user job at the time the dump was made.

12 The program counter (PC) 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 displayed in the previous portion of the listing.

The information shown from octal byte 1000 to the value of the initial stack pointer (Item 13) represents the current contents of the stack region. In this example, the contents of the used portion of the stack are shown in the region from octal byte 1760 to octal byte 1777. Octal byte 001760 contains the value 003770, which is the stack pointer value shown in Item 14. Note that the PDP-11 stack starts at the high address and works toward the lower address.

The octal bytes following the stack display the task image at the time of the dump. This display, which represents the state of the program at the time it aborted, is especially useful to software specialists.

Post-Mortem Dump Example

An example of a post-mortem dump is:

```
1 Post-Mortem Dump of SY:[1,201]PMD030.PMD on 13-Jun-85 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

  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.)

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

 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

 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 13-Jun-85 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
```

```
           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
002240 / 000000 000000 000000 000000 000000 000000 000000 000000
002260 / 000000 000000 000000 000000 000000 000000 000000 000000
002300 / 000000 000000 000000 000000 000000 000000 000000 000000
002320 / 000000 000000 000000 000000 000000 000000 000000 000000
002340 / 000000 000000 000000 000000 000000 000000 000000 000000
002360 / 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

Submitting Requests to the OPSER-Based Spooler: The QUE Program

The QUE system program submits requests (jobs) to the OPSER-based spooling programs. QUE also performs auxiliary operations such as listing, killing, or modifying pending requests. QUE checks the syntax and validity of each request and, if the request is valid, passes it to another program for further processing. If the request is invalid, QUE rejects the entire request, displays an error message, and allows you to try again.

To execute a request submitted to QUE, the system must contain three additional programs:

Program	Description
QUEMAN	The queue manager program. It processes a request created by QUE and places it in the system queue file.
OPSER	The OPerator SERvices program, which provides communication between the operator and the spooling programs.
spooling	The particular spooling program executes requests on its related device when QUEMAN passes a pending request to it. Valid spooling devices and their accompanying standard RSTS/E system programs are:

Device	Name	RSTS/E program
LP:	Line Printer	SPOOL
BA:	Batch	BATCH
RJ:	RJ2780	RJ2780 (optional)

Privilege Required

To run QUE, you need the following:

- o Read access to input files.
- o Write access to files specified with the /DE (delete) switch
- o GACNT or WACNT privilege to submit requests under a PPN other than your own

Running QUE at a Terminal

To run the QUE program, type:

```
$ RUN OPSER$:QUE
```

QUE displays an identifying header followed by a number sign (#) prompt to indicate its readiness to accept commands. QUE also displays the # prompt after processing each command.

If your system manager has defined a CCL command for the queue program, QUE for instance, you can invoke the queue program by typing $QUE.

If QUE runs, and finds an error when accessing the queue file, the program displays the following error message:

```
?Queue not initialized
```

If this occurs, ask your system manager run the QUEMAN program to initialize the queue file.

You can terminate QUE by typing E or entering CTRL/Z in response to the # prompt as follows:

```
#E
$
```

QUE returns control to the keyboard monitor command level and the corresponding prompt (in this case, $).

The following sections explain the commands to queue, list, kill, and modify jobs.

Table 2-18 summarizes the QUE commands.

Table 2-18: QUE Program Commands

Command	Format	Description
QUE	Q device:jobname=file spec(s)	Specifies one or more files you want to process. Use a comma to separate multiple file specifications.
LIST	L device:=jobname(s)	Displays a list of the currently pending requests for the spooling program on your terminal. If you omit device:, QUE assumes LP0:. If you omit device: but specify jobname(s), you must include the = symbol. Valid devices are: LP: LP0:-LP7: BA: BA0:-BA7: RJ: Specifying LP: or BA: indicates all units of that device. If you do not include a jobname, QUE displays all jobs for the specified device.
SHORT	S device:=jobname(s)	Same as the L (LIST) command, except that QUE displays a short form list of the currently pending requests for the spooling program on your terminal.

Table 2-18: QUE Program Commands (Cont.)

Command	Format	Description
KILL	K device:=jobname(s)	Removes one or more jobs you indicate in the device specification and jobname. If you omit device:, QUE assumes LP0:. If you omit device: but specify jobname, you must include the = symbol.
MODIFY	M device:jobname/switch(es)	Modifies the parameters of a job already in the queue. With the M (MODIFY) command you can modify these parameters: o Priority o Forms name o After date or time o Forms control type o Number of job copies o Job status o Mode
FLUSH	F dev: F BA:	Directs QUE to abort all current jobs in the queue. BA: specifies the OPSER-based Batch Processor, dev: specifies a line printer unit (LP0: is the default). If you specify LP: with no unit number, QUE aborts all jobs in any device queue. The F (FLUSH) command is available only to the users defined in the OPSER operator table.

Table 2-18: QUE Program Commands (Cont.)

Command	Format	Description
HELP	H	Displays an informational message on your terminal.
EXIT	E	Terminates QUE and returns control to system command level.
CTRL/Z	Enter CTRL/Z	Terminates QUE and returns control to system command level (same as E [EXIT] command).

Using the Q Command

Type Q in response to the number sign (#) prompt to request a spooling program. The Q command has the following general format:

```
Q device:jobname/sw=file spec, file spec,..., file spec
```

where:

device: designates the device and unit number that the spooling program services. The device can be:

    LPn:
    BAn:
    RJ:

    where n is an integer between zero and seven.

If you do not specify a device, LP0: is the default. LP: and BA: direct QUE to send the request to any available spooling program for that device type. If you specify a device, you must also specify the = symbol.

If you specify a unit number with device LP: or BA:, QUE designates the request for that unit. QUE executes the request only if a spooling program is running on that unit. If you do not specify a unit number with device LP: or BA:, QUE sends the request to the first available device.

The general OPSER-based Batch Processor BA: processes only those jobs queued for BA: and not jobs queued for BA0: - BA7:. However, BATCH processors BA0: - BA7: process requests queued for their respective units as well as requests queued for BA:.

jobname — identifies your request in the queue. Jobname can be a maximum of six alphanumeric characters. If you do not specify jobname, QUE uses the name of the first file you specify in the request.

If you have GACNT or WACNT privilege, you can specify an account with jobname. The account must be a valid user account on the system disk. If you do not specify an account, the program queues the request under your current account. Both your account and the account you specify must have access to any files you include in the request.

/sw — specifies one or more switches that determine the output characteristics of the entire request. See Table 2-19.

= — associates the specified device and jobname with the files you want to process. If you do not specify a device and jobname, the = symbol is optional.

file spec — specifies the file QUE processes from the queue and can include any combination of Q command switches. You can include up to 11 files in a request. No file can be larger than 32767 blocks. Use a comma (,) to separate each file specification. See the section "Q File Specifications" for a complete description of specification options and available switches.

Table 2-19 lists the Q command output switches.

Table 2-19: Q Command Output Switches

Switch	Format	Description
Mode	/MO:n	The value n specifies the MODE that the spooling program uses in its OPEN statement for the line printer (see the *RSTS/E Programming Manual*). You can use the following specific MODE switches instead of /MODE:n).

Table 2-19: Q Command Output Switches (Cont.)

Switch	Format	Description
Length	/LE:nnn	Set form length to nnn (1 to 127) lines per page.
Convert	/CO	Change number 0 (zero) to letter O (oh).
Truncate	/TR	Truncate lines longer than unit's configured length.
Line printer form	/LP	Enable software formatting.
Uppercase	/UP	Translate lowercase to uppercase characters.
Skip	/SK	Skip six lines at bottom of each form.
After	/AF:spec	Initiate the current request after the value given in spec. Spec can be one or more of the following: dd-mmm-yy hh:mm yy.mm.dd hh:mm:am/pm +n D[AYS] +n H[OURS]
Priority	/PR:n	QUE sets the priority to the value of n, which can be in the range 0 and 255. The default priority is 128. If you have EXQTA privilege or are defined in the OPSER operator table, you can specify a priority between 0 and 255. Otherwise you 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.

Table 2-19: Q Command Output Switches (Cont.)

Switch	Format	Description
Type	/TY:xxx	Use the format specified by xxx for printing the file. Value can be: FTN FORTRAN forms control EMB Embedded forms control IMP Implied forms control IMP (LF and CR printed before each record).
Forms	/FO:<form name>	String <form name> specifies the name of the form on which the job is printed; <form name> cannot exceed six characters, and the first character cannot be a digit. The default is NORMAL.
Job copies	/JC:n	The value of n specifies is the number of job copies you want to print. If you omit this switch, QUE prints one copy of the job.

Note

You can also specify file name switches /CO:nn, /NH, /DE, and /BI as job switches. When used as job switches, they apply to all files in the job. See Table 2-20 for a description of these switches.

The /AFTER switch directs the queue manager to initiate a job request after a particular date and time. The /AFTER switch arguments that specify the date and time have one or more of the following formats:

Name	Format	Description
Calendar date	/AF:dd-mmm-yy	dd is the date, mmm are the first three letters of the month, and yy is the year. If you omit the year specification, the default is the current year.
Numeric date	/AF:yy.mm.dd	yy is the year, mm is the number of the month, and dd is the day.

Name	Format	Description
Relative date	/AF:+n D[AYS]	n is the number of days from the current date.
Military time	/AF:hh:mm	24-hour time, where hh is the hour and mm is the minute.
AM/PM time	/AF:hh:mm am/pm,	hh is the hour, mm is the minute, and am or pm designates before or after noon.
Relative time	/AF:+n H[OURS]	n is an integer in the range of 0 to 24.

The date and time specifications you use with the /AFTER switch are subject to the following restrictions:

- When you use a date specification (calendar, numeric, or relative) without a time specification, the job begins after midnight on the specified date.
- You can use only one specification of each type. For example, you cannot use a relative date with a calendar date or a relative time with a military time.
- You can concatenate a date specification and a time specification. For example, /AF:dd-mmm-yy:hh:mm.
- When you use a relative time of 0 with a relative date specification, the job begins after the current time on the date specified. For example, assume you type the following option at 3 PM on September 12:

  ```
  /AFTER:+2 DAYS:+0 HOURS
  ```

  Que initiates the job after 3 PM on September 14.

Q Command File Specifications

The file specification part of the Q command line can be a full RSTS/E file specification of the following format:

```
device:filename.type[proj,prog]/sw/sw...
```

where:

device:	specifies a valid disk device. The default is the system device in the public structure. If you assign a logical name before you run QUE, the program successfully interprets the logical name: otherwise, you receive the error message ?Illegal input file. See the *RSTS/E System User's Guide* for a discussion of logical names.
filename.type	specifies the file name and type. The default file type is .LST for print queues and .CTL for batch queues. You can specify the * and ? wildcard characters for the file name and type. See the section "Using Wildcard Character."
[proj,prog]	specifies a PPN of the file you want to print. The default is the current account. You cannot use wildcard characters in the PPN.
/sw	specifies one or more Q command switches. See Table 2-20 for a description of the switches.

Using Wildcard Characters in File Specifications

When you specify a wildcard character for a file name or file type field:

- The * character replaces an entire field
- The ? character replaces a single character within the field

Specify wildcard characters 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.

Specification	Result
FILE??.TYP	All files with four- to six-character names, the first four of which are FILE, and a file type of TYP.
FILE??.T??	All files with four- to six-character names, the first four of which are FILE, and any type whose first character is T.
AB?.TYP	All files with two- to three-character names, the first two of which are AB, and a file type of TYP.
FILE??.*	All files with four- to six-character names, the first four of which are FILE, and any type.
*.TY?	All files of any name whose type begins with TY.

Table 2-20 lists and describes the Q command file name switches.

Table 2-20: Q Command File Name Switches

Switch	Format	Meaning
Copies	/CO:n	SPOOL prints the number of copies indicated by the decimal integer n. If you do not specify /CO:n, SPOOL prints one copy.
No Header	/NH	SPOOL suppresses printing of the file header.
Delete	/DE	SPOOL deletes the file after it is printed, if the protection code of the file permits. The QUEMAN program checks the protection code of the file. You cannot use /DE with file specifications that include wildcards.
More	/MORE	Indicates to QUE that you have more text to type on the next physical line. Used only at the end of a command line.
Binary	/BI	Indicates a binary file to the RJ2780 program, which must be in transparent mode.
Restart	/RE	SPOOL restarts the file or job if a malfunction occurs in printing.

Q Command Examples

Consider the following Q command:

```
#Q ABC.DAT
```

QUE sends file ABC.DAT, stored in your current account, to the queue manager for spooling to LP0:. 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
```

Que sends file A.BAS on disk DK1: to line printer unit 0. SPOOL generates two copies of the file under job header A.

To specify a job name for the request, type the name and the equal (=) symbol in the command. For example:

```
#Q SORT=DK1:FILEA,FILEB/CO:2
```

In this example, 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 = symbol in the command. For example:

```
#Q LP1:=DK1:FILOUT.001
```

In this example, QUE creates a request for FILOUT.001 on line printer unit 1, and assigns the job name FILOUT.

To specify a request that includes an account other than your own, type the PPN after the output device. For example:

```
#Q LP0:[1,10]=ABC.DAT
```

In this example, QUE creates a request to print file ABC.DAT, located in your current account, and queues the request under account [1,10]. You must have WACNT privilege, or GACNT privilege if the PPN you specify is in the [1,*] group. Account [1,10] must have Read access to file ABC.DAT.

To specify a request longer than one physical line, type the /MORE switch 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 displays the MORE > prompt, 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; QUE interprets the completed command as if it were typed on one line. Therefore, you must place all punctuation (commas, colons, and so on) in the appropriate positions.

When you specify a device designator 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:

```
#Q LP1:SORT=DK2:FILEA,FILEB
```

When QUE executes the previous command, the program sets the device to DK2: for both FILEA.LST and FILEB.LST. Because you did not specify a device designator for FILEB, DK2: is the default device.

To assign a priority to a job when queuing a file, specify the /PR:n switch with a value for n in the range of 0 to 255. Without the /PR switch, 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. 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. Scheduling is a function of priority; if you submit a high-priority batch job that runs for a long time, the job delays other jobs submitted to the same spooler.

Using the L and S Commands

The L command displays information on your terminal about pending requests for a specified device. The command line has the format:

```
L device:=jobname(s)
```

For a complete description of the device: and jobname specifications, see the section "Using the Q Command."

The following example shows the use of the L command:

```
#L LP0:

LP0 QUEUE LISTING          13-Jun-85      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
```

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 to which the file has been queued, 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
```

QUE displays jobs in the order in which QUEMAN accesses them. In the first line of each job description, the jobname precedes its account number and the sequence number (/SE:n) assigned to the job by QUEMAN. The /FO[RMS]:n switch follows the sequence number. Argument n refers to the name you specify, or the default NORMAL.

The status appears under the heading S in the second line of each job description. The status indicators are:

Status	Meaning
0	Job is in queue waiting to be processed.
S	Job has been sent to spooler.
A	Job is waiting for an /AFTER datetime 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 QUEMAN restarted and 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.

P for Priority (0 to 255) appears next on the second line; 128 is the default. Following the priority are any job switches you specify.

Appearing under the lines describing a job are the descriptions of individual files in that job--the full file specifications followed by any file switches you specify.

Note

> Because both QUE and QUEMAN can access a queue list simultaneously, QUE tests the linkage of the queue list to verify that QUEMAN did not change the queue list during QUE's search. If QUE detects that the linkage has changed, QUE displays the following message and restarts the listing from the beginning of the queue list:
>
> *****QUEUE CHANGING--WILL RESTART LISTING*****

To list jobs on all line printers, type:

```
#L LP:
```

QUE displays the related unit number in the UNIT column for each queued job. To display jobs queued for the Batch Processor, type the L BA: command; to display jobs queued for the RJ2780 program, type L RJ:.

To list only information about one request or several requests, include the = symbol followed by the jobname(s) in the L command. In the following example, QUE displays the header lines and the data for the jobname specified:

```
#L LP0:=REW[200,57]
LP0 QUEUE LISTING       09-Jun-85     03:22 PM
UNIT    JOB             S / P      FILES
 0      REW[200,57]/SE:2775/FO:NORMAL
                        S 128TY:EMB
                                   SY :[200,57]EXPENS.BAS
                                   SY :[200,57]VISITS.LST/C:3
```

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      13-Jun-85     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 on your terminal, information about pending requests for a specified device. The command line has the same format as the L command line, described earlier in this section.

The following example shows the command and its result:

```
#S LPO:
LPO SHORT QUEUE LISTING 15-Jun-85        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 directs QUE to display 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:

- o Jobname
- o Job account number
- o Sequence number (/SE:)
- o /FORMS switch, argument (/FO:n)
- o Job status
- o Job priority
- o /AFTER time specified

Using the K Command

The K command removes one or more jobs from the queue of a specified device. To remove a job, specify a command in the following format:

```
K device:=jobname(s)
```

You must be defined as a valid operator to remove any job other than your own.

For a complete description of the device: and jobname specifications, see the section "Using the Q Command."

The following examples show the use of the K command:

```
#K LP1:=PRINT1,BATCH1
```

This command instructs QUE to delete from the queue on line printer LP1: each job with the name PRINT1 or BATCH1 under your account if the spooling program is not currently processing the job.

In the case of matching jobnames, specify the job you want to remove from the queue by including the /SE:nnnnn switch 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 (see the previous section). This example shows the use of the /SE switch:

```
#K LP1:=PRINT1/SE:1475
```

Using the M Command

The M command lets you modify the parameters of a job that is already in the queue. By including the M command with appropriate switches, you can modify the job parameters originally set in the Q command; for example, priority, mode, date/time, number job copies, form, and type. However, if the job has already been sent to a spooling program, QUEMAN does not allow any modifications.

You must be defined as a valid operator to modify any job other than your own.

To specify modifications to the job, type the device designator followed by the jobname and the desired modification switches in the following format:

```
M device:jobname/switch(es)
```

The following list contains job modification switches that you can specify with the M command. Except where otherwise indicated (for example /HOLD and /UNHOLD), these switches correspond in name, function, and format to those of the Q command.

Switch	Format	Description
Hold	/HO	Halt the job's advancement in the queue, and do not send it to a spooling program until the /UNHOLD switch is specified.
Unhold	/UNHO	Continue the job's advancement in the queue.
Sequence	/SE:nnnnn	Modify only the job with sequence number nnnnn (assigned by QUEMAN; see the section "Using the L and S Commands"). Use this switch when there are matching jobnames.

See Table 2-19 for explanations of the following switches:

```
/PR:n

/MO:n or
    /LE:nnn     /UP
    /CO         /SK
    /TR
    /LP

/AF: dd-mmm-yy    hh:mm
     yy.mm.dd     hh:mm am/pm
     +n D[AYS]    +n H[OURS]

/TY:xxx

/FO:<form name>

/JC:n
```

Running QUE Using CCL Commands

You can run QUE using a CCL command if your system manager has defined one for the QUE program on your system. For example, if the CCL command is QUE, the command format is:

```
QUE/<command line>
```

The <command line> can be any valid QUE program command. If QUE does not find a slash (/) character immediately following the word QUE, the program assumes the command is a Q command. For example, the following two commands are identical:

```
QUE/Q JOB1=ABC.DAT

QUE JOB1=ABC.DAT
```

These commands are the same because both enter JOB1 in the queue for line printer 0. In either case, QUE runs, executes the command, and exits. An error in the command causes QUE to display the related error message on a subsequent line, and then exit.

To include an output switch when queuing a job, use /Q in the QUE command line or specify an explicit jobname in the QUE command. For example:

```
QUE/Q/MODE:2048=DK1:DISK.BAS
```

The / character before the Q character marks the distinction between the Q command and the jobname Q.

To list or kill jobs, type QUE, followed by a / character and the appropriate command. For example:

```
$ QUE/L DISPLY[1,253]
LP0 QUEUE LISTING    13-Jun-85      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

$
```

QUE runs and executes the command indicated following the / character, after which it returns control to the system command level.

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 that describes the error condition.

When QUE finds an error, it does not execute any part of the command. You must retype the entire command.

QUE returns the error message ?Illegal input file when it finds any RSTS/E system error. When running by means of a CHAIN operation from a BASIC-PLUS program, QUE reports an error by displaying an error code when it passes control.

Table 2-21 lists both the messages and codes and describes the related error conditions.

Table 2-21: 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 You specified a switch on the output side of the QUE command line that is reserved for the input 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 You specified more than eleven files in the Q or K command.	5
?NULL FILE SPEC You must give at least one file specification in a Q or K command and none was found; or you included two commas with no file specification between them.	6
?ILLEGAL INPUT FILE-<text---filespec> You specified a file that caused this RSTS error. Retype the request.	8
?WILDCARDS NOT ALLOWED IN PPN You typed an * character in the project-programmer field.	9
?QUEUE FULL The queue file is temporarily full. Try again after 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>.	11

Table 2-21: QUE Error Messages and Codes (Cont.)

Message and Meaning	Error Code
?'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
?QUEUE NOT INITIALIZED The queue file does not exist in the system library account. You 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 The specified switches are illegal for the commands used. For example, /HOLD and /UNHOLD are illegal for Q.	18

Restoring Pre-V9.0 Backed-Up Files: The RESTOR Program

The RSTS/E RESTOR program lets you return files you backed up before RSTS/E V9.0 (using the BACKUP program) to online use. The RSTS/E V9.0 RESTOR utility includes the restore features previously included in the BACKUP utility.

As you read the following sections, keep in mind that:

- RESTOR, printed in all uppercase letters, refers to the entire package; Restore, with an initial capital only, refers to a particular RESTOR operation.

- RESTOR checks for files that were contiguous when you backed them up, and attempts to restore them to their original contiguous state. If there is not enough contiguous space on the disk, the program does not restore the file. The only way to restore the file is to make enough contiguous room for it.

Privilege Required

To run RESTOR, you need WACNT, WREAD, and WWRITE privileges.

RESTOR and System Management

RESTOR provides three operational modes:

- Restore, to rebuild Backup-created files onto a RSTS/E disk

- Loadindex, to copy the primary index file from the last volume of the Backup Set to a RSTS/E formatted disk

- List, to print directory information contained in a Backup index file

How RESTOR Works

Important characteristics of the RESTOR package are:

- o User dialogue
- o File selection
- o Account creation (optional)
- o File transfer from the backup medium
- o File comparison (optional)
- o List file generation

The following sections outline the functions RESTOR performs within each of these categories. The discussion is limited to Restore operations because you should perform new backup operations using the DCL command BACKUP.

User Dialogue

The dialogue begins when you run RESTOR. The first question asks which of the three operational modes you want to perform. If you choose, you may create a DCL command file. See the *RSTS/E Guide to Writing Command Procedures*.

The dialogue then asks you where to create the work file into which it will write the directory and error information. Additional questions ask for information on which files and accounts to transfer. When the dialogue ends, the program writes a summary of your commands into the listing file.

File Selection

Next, RESTOR searches for all the files and accounts you specified during the dialogue phase. In Restore mode, the program looks up files in one of three index files:

- o Primary index file
- o Secondary index file
- o Auxiliary index file

The indexes are directories of all the tape or disk volumes of the Backup Set. BACKUP writes a primary index file and a secondary index file as the last two files on the Backup Set. The third index remains in your account; it is called the auxiliary index file.

The auxiliary index file can be in one of two formats. If the date format (selected during system installation) is alphabetic, then the format is:

Bddmmm.Jnn

where:

B	indicates Backup
dd	is the date
mmm	is the month
nn	is the job number

For example, a file named B11JUN.J08 was created for job number 8 on June 11.

If the date format is numeric, the format is:

Bymmdd.Jnn

where:

B	indicates Backup
y	is the last digit of the year
mm	is the number of the month
dd	is the date
nn	is the job number

For example, a file named B40611.J08 was created for job number 8 on June 11.

If you have a multivolume Backup Set, you must load the last volume of the set before you load the volumes in succession for the Restore operation. RESTOR must have access to the index before it can begin restoring a disk. To avoid the need to load the last volume of the Backup Set each time you restore a disk, use the Loadindex mode of RESTOR. Loadindex places the index on another disk where RESTOR can access it. See the section "Load Index File Example" for more information.

As it makes the selections, RESTOR writes information about each selected file and account into the work file. Therefore, the work file can become quite large as the selection phase proceeds. The size of the work file depends on the number and size of files selected for transfer. During the dialogue, you can specify that the work file be placed on a private disk to make sure that enough space is available.

Do not specify a contiguous work or listing file. RESTOR expands both these files and the system displays the error message ?Protection violation if RESTOR tries to expand a contiguous file past the file's initial size. You must make sure enough room is available on the disk you specify for the work and list files. In addition, both files must have different names.

To estimate the size of the work file for RESTOR, consider the following guidelines:

- The work file has a fixed overhead of two to four blocks.
- Each file you specify requires one-eighth of a block.
- Each exception also requires one-eighth of a block. See the section "File Specification" for a description of the exceptions.
- Errors and file attribute blocks require an additional one-eighth of a block each.

Therefore, for a system of 512 accounts with an average of 16 files each (no exceptions) and 16 files with attributes, you can expect the work file to contain at least 1094 disk blocks. DIGITAL recommends that you add approximately 10% as error space when backing up a large system. In this example, therefore, the total is 1204 blocks.

Account Creation (Optional)

The Backup Set may contain accounts that do not exist on the disk (destination disk) you are restoring. RESTOR asks a dialogue question that allows you to reproduce these accounts on the destination disk. When you complete the dialogue, RESTOR first creates the selected accounts that do not exist on the destination disk and then transfers the files (see the following section). Accounts recreated on the destination disk retain the same password, quota, and cluster size; however, they do not retain the same account names.

The ENTER ACCOUNTS question does not allow you to selectively create accounts. RESTOR either creates all the selected accounts on the destination disk or creates none of them. If you want to reproduce all accounts, accept the default (YES); otherwise, type NO. Note that RESTOR does not reproduce any account that did not contain files when you backed up that account.

Note

> Because all account passwords are on the Backup Set, protect your Backup Set from access by unauthorized personnel.

File Transfer from the Backup Medium

When the selection process is complete, RESTOR transfers files in the order it finds them on the Backup Set.

If you include a file specification with the SUPERSEDE dialogue question, RESTOR replaces files on the destination volume with the backup versions. Accept the NONE default to supersede no files. While the default to the SUPERSEDE question is NONE, you must include a file specification to override the default. Note that the words ALL and YES only make RESTOR try to supersede files named ALL and YES. When RESTOR needs a new volume or finds an error, it prints a prompt on the job's terminal or (if the job is detached) on the operator services console (OSC). RESTOR also logs all errors in the listing file.

File Comparison (Optional)

After it copies the files and accounts, RESTOR compares the original files with their duplicates on the Backup Set (if you requested a comparison). RESTOR records any differences between the two files in the listing file and on the job's console terminal.

While the default to the COMPARE FILE(S) question is NONE, you must include a file specification to override the default. Do not enter the words ALL or YES.

List File Generation

As the last step, RESTOR generates the remaining listing file text. RESTOR lists each file selected for transfer, whether or not the program transfers that file. For example, RESTOR can select a zero-length file for transfer, but it does not transfer the file's data. The generation of the listing file can be time-consuming because RESTOR copies ASCII data from the work file to the listing file (which can be output on a terminal or written to disk). The listing file can grow quite large for a large system Restore operation. If the listing file consumes all free disk space, you can abort the RESTOR run. See the section "Interruption Commands" for a description of the ABORT command. Aborting the run while RESTOR is generating the listing file does not harm the Backup Set or the restored files in any way: it merely leaves you with an incomplete record of the Restore operation.

Running RESTOR

To run the RESTOR program, type:

```
$ RUN RESTOR$:RESTOR
```

RESTOR displays the RES[TORE], LOA[DINDEX], or LIS[T] question. If you want to operate in one of the three RESTOR modes from a DCL command file, type one of the following commands in response to the question (see the *RSTS/E Guide to Writing Command Procedures*). Substitute the name of the DCL command file for filename.type. For example:

```
RES @filename.type

LOA @filename.type

LIS @filename.type
```

The defaults for DCL command file names are RESTOR.CMD, LOADIN.CMD, and LIST.CMD.

To create a DCL command file, append the /SAVE (or /SA) option in response to the RES[TORE], LOA[DINDEX], or LIS[T] question. The following example shows the procedure and RESTOR's response:

```
RES[TORE], LOA[DINDEX], or LIS[T]? RES/SAVE
INDIRECT FILE NAME<SY:[  1,213]RESTOR.CMD> ?
```

In specifying a file other than the default, you can designate a disk or tape as the output device. After you answer the question, RESTOR continues to prompt normally, writes your responses into the DCL command file, and carries out the commands.

File Specification

Several questions in the dialogue require you to respond with a file specification. The file specification lists criteria that files must meet for RESTOR to process them. The criteria can include file name, account, and dates of creation, and last access. In addition, you can specify one or more files as exceptions from a process. The file specification has the format:

```
filename/keyword:comparison:date:time/exception (filename . . .)
```

where:

filename
: is a standard RSTS/E file specification of the format:

  [proj,prog]filename.type

  You can use the * and the ? wildcard characters in the PPN, file name, and file type fields.

keyword
: is either CR[EATION] or AC[CESS] and specifies that RESTOR is to interpret the date and time as either file creation or access dates and times.

comparison
: is either BEF[ORE] or AFT[ER] and specifies that RESTOR is to process files created or accessed before or after the date and time.

date
: specifies the date RESTOR uses to process files. The format is:

  DD-MMM-YY

  where:

  DD — specifies the day of the month.

  MMM — specifies the first three month letters.

  YY — specifies the last two digits of the year.

time
: optionally specifies the time RESTOR uses to process files only when you specify CREATION. Use the 24-hour format such as 14:49. The default is 00:01.

exception
: is EXC[EPT] and specifies that RESTOR does not process the specified files. You can include all of the previous elements in an EXCEPT phrase but do not include another EXCEPT phrase. Separate multiple file specifications with commas and enclose the list within parentheses.

Separate creation and access date comparisons from each other and from the file name (even if it is null) by a slash (/). For example, a file specification applied to the default file name is:

```
/CR:BEF:13-JUN-85
```

The previous specification designates any files named by the default that were created before 13-JUN-85 (that is, 12-JUN-85 or earlier).

A colon (:) separates the components of the specification. You can include only one CREATION comparison and one ACCESS comparison in each file specification. For example:

```
*.BAC/AC:BEF:13-JUN-85/CR:BEF:09-APR-85
```

This specification designates all files with the file type .BAC (in the default account) that were last accessed before 13-JUN-85 and created before 09-APR-85.

In addition to specifying a date, you can designate the time of day (in 24-hour format) in the CREATION comparison only. The following specification designates all files with name PLANTZ that have been created since 21:13 (9:13 PM) on 13-JUN-85:

```
PLANTZ.*/CR:AFT:13-JUN-85:21:13
```

You can exempt one or more files from a comparison or operation by including an /EXCEPT switch in the file specification. Follow these rules when using the /EXCEPT switch:

- o Any /EXCEPT: switch is associated with the file specification immediately preceding the slash (/).
- o You must enclose All /EXCEPT switch file specifications in a single set of parentheses (...).

Note the following incorrect command line:

```
[1,200]*.*,[1,210]*.*/EXCEPT:[1,200]*.SY
```

Do not associate the /EXCEPT switch with the [1,210] file specification, but rather with the file specification for account [1,200]. Instead of placing the /EXCEPT:[1,200]*.SYS specification after the specification for account [1,210], place it with the file specification, [1,200]*.* as follows:

```
[1,200]*.*/EXCEPT:[1,200]*.SYS,[1,210]*.*
```

This is the correct command line for the previous example. You now have the /EXCEPT switch immediately following the file specification you want it to define.

To include a CREATION or ACCESS comparison in an exception, you must enclose the files and any comparisons in parentheses as in the previous example:

```
*.*/CR:BEF:13-JUN-85/EXC:(*.RNO/CR:AFT:16-MAY-85,MT?.*/AC:AFT:09-APR-85)
```

This command line specifies all files on the default account that were created before 13-JUN-85. The exceptions are:

- o All files with file type .RNO that were created after 16-MAY-85
- o Any file whose name is three characters long, begins with MT, and has been accessed since 09-APR-85

A CREATION or ACCESS comparison applies to the files in an EXCEPT comparison as well. To exclude an EXCEPT comparison from CREATION and ACCESS comparisons, specify a CREATION or ACCESS date for the exception (and enclose the exception in parentheses). A file specification can include only one EXCEPT comparison. The EXCEPT comparison must follow any CREATION or ACCESS date comparisons.

A list of file specifications often uses several lines. Use a hyphen (-) as the last character before the line ends; this indicates that the next line is a continuation of the current line. In response to the hyphen and line terminator, RESTOR prompts with CONT> and a tab. The specification must include all standard punctuation in addition to the hyphen. RESTOR performs no syntax or semantic processing on a response until you have entered all continuation lines. The following example shows the use of a continuation line, this time without the use of the /EXCEPT switch. (Note that the default is [1,213], which you are currently logged in to.)

```
FROM FILES<[1,213]*.*>? BACK.RNO,CHAP3.RNO,TEST.BAS,-
CONT> TAPE.BAS,[2,213]PRINT.BAC
```

For example, the following command line designates all files on account [100,250] (except SOCIO.TMP) that were created after 13-JUN-85:

```
[100,250]*.*/CR:AFT:13-JUN-85/EXC:SOCIO.TMP
```

RESTOR ignores spaces as separators in specification lines but considers them significant as terminators. For example, the following line is illegal because the space terminates the scan after the characters EXC:

```
EXC EPT:BEE.BAS
```

On the other hand, in EXCEPT#:#BEE.BAS, RESTOR ignores the spaces and processes the text as:

```
EXCEPT#:#BEE.BAS
```

You can include a comment in a command line by beginning the comment with an exclamation mark (!), for example:

```
FROM FILES<[1,213]*.*>? W:MANCH8.RNO!THIS FILE CONTAINS CHAPTER 8
```

If you need to continue a command line and also include a comment, place the hyphen denoting a continuation line before the exclamation mark. For example:

```
FROM FILES<[1,213]*.*>? CHAP2.RNO,BACK.CMD,TAPE.BAS-!THIS IS A COMMENT
CONT>
```

Running RESTOR Under OPSER-Based Batch Processing

The OPSER-Based batch processor lets you perform a Restore operation without any dialogue interaction. This is possible if you create and execute a batch control file. In addition to BATCH control statements, the batch control file contains:

- The RESTOR run command: RUN RESTOR$:RESTOR.
- The responses to all dialogue questions, either as an explicit list within the file or as a list in the DCL command file specified in the first dialogue question response. See the section "Running RESTOR" for a discussion on the /SA[VE] option that creates the DCL command file.
- The responses to all the mount dialogue questions.

See Appendix A for a description of the OPSER-Based Batch Processor control statements.

You must know the number of devices necessary for a RESTOR operation before submitting the batch job. If you do not know the exact number, it is better to include too many devices rather than too few. Furthermore, you must allocate a different drive for each device you specify. Make sure there are as many drives of the device type you are using as there are volumes in the Backup Set.

You must include also a response for every mount question generated, such as the MOUNT IT ANYWAY? question that appears only if a RESTOR volume's expiration date has not passed, or if a magnetic tape cannot be read. The general rule is to anticipate all mount questions and include an appropriate response.

The RESTOR Dialogue

The RESTOR dialogue lets you specify which files and accounts to restore, compare to their original versions, and delete. During the dialogue, RESTOR asks the same questions of all users.

If you answer NO, RESTOR does not add any accounts to the destination disk.

If you specify an account in the FROM FILE(S) <[CUR ACT] *.*> question and that account does not exist on the destination disk, RESTOR transfers the account only if you answer YES to the ENTER ACCOUNTS question.

Tables 2-22 through 2-24 list the RESTOR questions and the responses the program requires. Table 2-22 list the RESTOR Dialogue Summary.

Table 2-22: RESTOR Dialogue Summary

Question	Response and Meaning
1	RES[TORE], LOA[DINDEX], or LIS[T]? RES[TORE] (/SA[VE]) Restores files. If you append /SA[VE], RESTOR creates an indirect command file and asks question 1a.
1a	INDIRECT FILE NAME<_SY:[CUR ACT]RESTOR.CMD>? File specification Creates an indirect command file with the specified name. Valid output devices are disk and tape. Asked only if the SAVE switch is appended in question 1.
2	WORK-FILE NAME<_SY:[CUR ACT] Bddmmm.Jnn>? File specification

Table 2-22 (cont.)

Question	Response and Meaning
	Uses the specified file as the work file. If the default is accepted, RESTOR uses either Bddmmm or Bymmdd as the file name format. If the system date format, selected during system installation, is alphabetic, the default name format is Bddmmm. where: dd specifies the day of the month mmm specifies the first three characters of the month If the system date format is numeric, Bymmdd is the default name format. where: y specifies the last digit of the current year mm specifies the number of the month dd specifies the day of the month In the .Jnn file type, nn represents the job number RESTOR is running under. You can substitute any mounted, write-enabled disk for SY:.
3	LISTING FILE<_KB:>?
	KBn: or LPn: Writes the listing file to the specified keyboard or line printer unit.
	File specification Writes the listing file to the specified file on disk or tape. The default file name is SY: [CUR ACT] RESTOR.LST.

Table 2-22 (cont.)

Question	Response and Meaning
4	INDEX FILE<PRIMARY>? File specification Uses the specified file as the index file. You can specify an auxiliary index file or an index file created during a Loadindex operation. If you accept the default, RESTOR uses the primary index file that is on the final volume of the Backup Set.
5	FROM DEVICE<_MT:>? MT:, MM:, MS:, or disk name Restores backed-up files from the specified medium. (Do not specify a unit number here, only a device type. RESTOR requests the unit number later.)
6	FROM FILE(S) <[CUR ACT]*.*>? BACKUP File specification(s) Restores the specified files.
7	TO DISK<_SY:[*,*]>? Disk name Restores the specified files to the designated disk.
8	BEGIN AT<[*,*]*.*>? BACKUP File specification Restores starting with the file specified here. The default starts with the first file matching the file specification in question 6. The answer must be a single file specification. No EXCEPT modifiers are allowed.
9	ENTER ACCOUNTS<YES>? YES Creates, on the destination disk, all accounts on the Backup Set that do not already exist on the destination disk NO Does not create any accounts. You cannot selectively create accounts.

Table 2-22 (cont.)

Question	Response and Meaning
10	SUPERSEDE<NONE>? BACKUP File specification(s) Overwrites the specified files on the destination disk with the backup versions. If you accept the default, RESTOR does not supersede any files. (To supersede files, you must include a file specification. RESTOR does not supersede all files on the destination volume if you type ALL or YES. Instead of superseding the entire volume, RESTOR replaces only files that have a file specification of ALL or YES.)
11	COMPARE FILE(S)<NONE>? BACKUP File specification(s) Compares the restored files to the backup versions after transfer. As in the SUPERSEDE question, you must provide a file specification here instead of an answer such as ALL or YES.

Table 2-23 lists the LOADINDEX Dialogue Summary.

Table 2-23: LOADINDEX Dialogue Summary

Question	Response and Meaning
1	RES[TORE], LOA[DINDEX], or LIS[T]? LOA[DINDEX]? Copies index file from the Backup Set to a RSTS/E formatted disk. If you attach /SA[VE] to the LOA[DINDEX] response, RESTOR creates an indirect command file and asks question 1a.
1a	INDIRECT FILE NAME <_SY:[CUR ACT]LOADIN.CMD>? File specification Creates an indirect command file with the specified name. Valid output devices are disk and tape. This question is asked only if you use the /SA[VE] switch in response to question 1.
2	WORK-FILE NAME<_SY:[CUR ACT]Bddmmm.Jnn>? File specification Uses the specified file as the work file. If you accept the default, LOADINDEX uses either Bddmmm or Bymmdd as the file name format. where: dd specifies the day of the month mmm specifies the first three characters of the month y specifies the last digit of the year mm specifies the number of the month In the file type .Jnn, represents the job number RESTOR is running under. You can substitute any mounted, write-enabled disk for SY:.
3	LISTING FILE<_KB:>? KB: or LP: Writes the listing file to the keyboard or the line printer. If you press RETURN or LINE FEED, RESTOR sends the listing file to the keyboard <KB:>.

Table 2-23 (cont.)

Question	Response and Meaning
	File specification Writes the listing file to the specified disk or tape. If you type SY: and then press RETURN, the RESTOR program creates the default listing file [CUR ACT]LOADIN.LST.
4	FROM DEVICE<_MT:>? MT:, MM:, MS:, or Disk Name Copies the index file from the specified Backup Set device.
5	TO FILE<_SY:[CUR ACT] [BINDdd.IND]? File specification Loads the Backup Set index file into the specified file. You can copy the index to a disk device only. The format for the default file name is BINDdd.IND, where dd is the numeric day of the month.

Table 2-24 lists the LIST Dialogue Summary.

Table 2-24: LIST Dialogue Summary

Question	Response and Meaning
1	RES[TORE], LOA[DINDEX], or LIS[T]? LIS[T]/SA[VE] Runs the LIST dialogue. When you attach the /SA[VE] switch to the LIST response, RESTOR creates a DCL command file after you respond to question 1a.
1a	INDIRECT FILE NAME<_SY:[CUR ACT][LIST .CMD]>? File specification Creates a DCL command file with the specified name. Question 1a appears only if you use the /SA[VE] switch in response to question 1. Valid output devices are disk and tape.
2	WORK-FILE NAME<_SY:[CUR ACT]Bddmmm.Jnn>? File specification Uses the specified file as the work file. If you accept the default by pressing RETURN or LINE FEED, LIST uses either Bddmmm or Bymmdd as the file name format. where: dd specifies the day of the month mmm specifies the first three letters of the month mm specifies the number of the month In the file type .Jnn, nn represents the job number RESTOR is running under. You can substitute any mounted, write-enabled disk for SY:.
3	LISTING FILE<_KB:>? KB: or LP: Writes the backup directory information to the specified device. If you press RETURN or LINE FEED, the backup directory is written to your keyboard. Type LP: and press RETURN for a line printer listing.

Table 2-24 (cont.)

Question	Response and Meaning
	File specification Writes the backup directory information to the specified file. Type SY:, and press RETURN if you want RESTOR to create the default listing file, [CUR ACT]LIST.LST.
4	INDEX FILE<PRIMARY>? File specification Extracts the index file from the specified file and writes it to the listing file. If you accept the default by pressing RETURN or LINE FEED, LIST requests the index volume. It then takes the directory information from the index volume and writes the information to the listing file. The program asks the next question, FROM DEVICE<_MT:>, if you accept the default in question 4.
5	FROM DEVICE<_MT:>? File specification Asked only if you accept the default in the INDEX FILE<PRIMARY> question. The file specification response must identify the device on which the index resides. You can accept the magnetic tape default <_MT:> by pressing RETURN or LINE FEED. The LIST program begins a MOUNT dialogue, which asks you to mount the Backup Set index volume and identify the target device.

Interruption Commands

The RESTOR package includes interruption commands for your convenience. These commands enable you to detach, terminate, suspend, and continue processing, and to check the status of the run.

RESTOR shows it is ready to accept interruption commands by printing an * character on the job's console terminal after the dialogue is complete. You can type an interruption command as a response to a request in the MOUNT dialogue or any time after the * character appears.

Table 2-25 lists the RESTOR interruption commands and their meanings.

Table 2-25: RESTOR Interruption Commands

Command and Meaning
ABO[RT] Terminates processing immediately and returns control to your keyboard monitor.
CON[TINUE] Continues processing after a PAUSE command.
DET[ACH] Detaches job from KB:. RESTOR detaches if three conditions are met: o OPSER is running, o The listing file is not on KB: o No messages are currently pending for this copy of RESTOR. RESTOR is given a receiver identification of BACKnn, where nn is its job number. All interaction with RESTOR is done through OPSER.
END Terminates after processing the current file.
LAS[T] Reprints, on KB:, the last text printed. If the program is awaiting a response, reprints the question or prompt. If not, reprints the last message.
LEG[AL] Prints, on KB:, a list of the commands and responses that are legal now. In the list, **OTHER** means that a request for information from you is pending, for instance, Backup Set name. The response to such a request is any appropriate string. You do not need to use quotation marks (")unless the response can be construed as an interruption command. For example, you can legally name the Backup Set PAUSE, but you must type the name in quotes ("PAUSE")

Table 2-25: RESTOR Interruption Commands (Cont.)

Command and Meaning
NOT[ICE] Writes the specified text in the listing file. The NOTICE command has the format: NOT[ICE] message Message represents one line of text. No line terminators are allowed in the message.
OFF[LINE] Same function as ABORT.
PAU[SE] Suspends execution until you type CONTINUE. Any legal command typed during a PAUSE is executed immediately.
STA[TUS] Prints, on KB:, RESTOR status information.
TER[MINATE] Closes the current volume at the end of the current file or at the end of the current output volume, whichever comes first.

Mounting and Dismounting Volumes

After RESTOR selects all files and accounts for transfer, it requests Backup Set labeling information and the device unit number for the first backup volume. It also prints a volume identification summary.

The labeling information is the name and expiration date of the Backup Set. The name identifies the Backup Set. RESTOR interprets the expiration date as the date after which it can automatically write over the data on the volume. The default expiration date is one year from the current date. If you mount a volume for backup before its expiration date, RESTOR asks for confirmation before writing on the volume. RESTOR requests density and parity for 800 bpi magnetic tapes, and density for 1600 bpi tapes.

After answering the labeling questions, mount the volume and write-lock it for a Restore operation. In response to the DEVICE question, type the full mounted volume device specification (device mnemonic and unit number followed by a colon); for example, MM1:.

Note

> You must keep the disks that RESTOR uses write-enabled because bad block files on disk must be updated as required.
>
> RESTOR executes the interruption commands and the RET[RY], SKI[P], and IGN[ORE] requests during the MOUNT dialogue if responses to any question has as its first three characters the first three characters of a command or request. To avoid this, enclose the response in quotation marks (" "), or make sure the response does not contain the same first three characters as the commands and requests.

The following example shows the mount procedure printout. Press LINE FEED to select a default response:

```
PLEASE ENTER BACKUP SET NAME<BACK28>  --  <LF>
PLEASE ENTER DENSITY IN BPI<800>  --  800 <RET>
PLEASE ENTER THE PARITY<ODD>  --  <LF>
MOUNT      DEVICE:         _MM:
                                 ID:           BACK28
                               SEQ#:           1
                            DENSITY:           800 BPI
                             PARITY:           ODD
DEVICE? MM1:
THIS VOLUME HAS NO BACKUP LABEL!
MOUNT IT ANYWAY<NO>? Y
```

RESTOR attempts to read the mounted tape at all valid density and parity settings to determine whether there is a backup label on the tape.

When RESTOR finishes with a volume, the program prints a dismount message. For example:

```
DISMOUNT DEVICE:        _MM1:
                                 ID:        BACK28
                               SEQ#:        1
                            DENSITY:        800 BPI
                             PARITY:        ODD
```

When the dismount message appears, physically dismount the volume and prepare the next volume for processing. RESTOR requests name, expiration date, parity, and density for only the first volume in each Backup Set. Use the information in the dismount message to label the tape. For later volumes, RESTOR requires only the device specification.

Note

> RESTOR logically mounts disks you use during any of its operations. Do not attempt to mount or dismount them logically using the DCL MOUNT or DISMOUNT commands. DCL expects a RSTS/E file-structured disk. A RESTOR volume has its own structure and causes the system to generate an error when it tries to logically mount or dismount a RESTOR volume.

RESTOR Error Handling

Four types of errors are possible with RESTOR:

- Dialogue command errors that occur during the Restore, Loadindex, or List dialogue. RESTOR diagnoses these errors immediately. Usually, you can respond to these errors by typing the correct dialogue answer.
- Interruption command errors that occur when you type an interruption command. RESTOR responds immediately to these errors, also. Type a valid command.
- Volume mount errors that occur when RESTOR mounts tape and disk backup volumes. Some volume mount errors are user errors (for example, specifying an illegal density setting for tape); others involve hardware (for example, tape errors that prevent labeling).
- Processing errors that occur any time during the BACKUP run except during the dialogue. The hardware on which BACKUP is running or logic errors can cause processing errors.

The following sections describe each of these four error types.

Dialogue Command Errors

When RESTOR finds a dialogue command error (usually a syntax error), the program prints the error message ?Command error on your terminal.

The error message is followed by either a RESTOR-specific error message or a RSTS/E error message generated during the syntax processing of the command. After printing the error message, RESTOR prints a question mark (?) and repeats its prompt. If you type a question mark (?) and press RETURN, RESTOR prints the command line up to the position of the error, and then prints the prompt again.

Table 2-26 lists the RESTOR dialogue error messages and their meanings.

Table 2-26: RESTOR Dialogue Error Messages

Message and Meaning
?BAD DIRECTORY FOR DEVICE The directory structure on the device you specified is corrupt. Try to place the file on a different device.
?CAN'T FIND FILE OR ACCOUNT RESTOR cannot find the file or account you specified. Make sure you typed the file name and account correctly and that they exist.
?DEVICE HUNG OR WRITE-LOCKED The device you specified is hung or write-locked or has generated a read or write error. Make sure the device is ready, write-enable the device (if necessary), or specify a different device.
?DEVICE NOT AVAILABLE The device you specified is disabled or assigned to another user. Specify a different device.
?DUPLICATE SWITCH The file specification contains multiple ACCESS or CREATION comparisons. RESTOR accepts only one of each comparison in a file specification.
?ILLEGAL EXCEPT NESTING The file specification includes more than one EXCEPT comparison. RESTOR accepts only one EXCEPT in a file specification.
?ILLEGAL FIELD The response contains a field that is not permitted. For example, you cannot specify a device name in the response to the FROM FILES question.

Table 2-26: RESTOR Dialogue Error Messages (Cont.)

Message and Meaning
?ILLEGAL FILENAME The file name you specified contains invalid characters or has an incorrect format.
?ILLEGAL KEYWORD The response contains a misspelled or incorrectly abbreviated keyword or has incorrect punctuation marks.
?ILLEGAL OPERAND The EXCEPT comparison in the file specification contains an unmatched parenthesis, or the day or year number in a date is out of range.
?INCOMPLETE COMMAND FILE You entered CTRL/Z or the DCL command file ended before the expected end-of-file (EOF). RESTOR returns to the RES[TORE], LOA[DINDEX], or LIS[T] question.
?NO DEFAULT The current dialogue question does not have a default answer. You must type an explicit response.
?NOT A VALID DEVICE The device you specified is not on this system.
?NOT A VALID DEVICE (PROHIBITED) The device you specified in response to this question is illegal.
?PROTECTION VIOLATION The file or account you specified is protected.
?TOO MANY FILES OPEN ON UNIT Only one file at a time can be open on a magnetic tape unit. Specify the second file on another device.
?TOO MUCH DATA The response contains more data than RESTOR accepts for this question.
?UNIT NUMBER NOT VALID RESTOR does not accept a device unit number in the response to this question.

Interruption Command Errors

An invalid interruption command generates one of three error messages.

Table 2-27 lists the RESTOR interruption command error messages and their meanings.

Table 2-27: RESTOR Interruption Command Error Messages

Message and Meaning
?CAN'T DETACH The DETACH command is invalid because one or more of the three DETACH conditions described in Table 2-25 is not true.
?UNRECOGNIZED COMMAND You typed a string that is not an interruption command.
?ILLEGAL COMMAND You typed an interruption command that is illegal (for example, typing CONT when no PAUSE is in effect). Type LEGAL for a list of interruption commands that are currently legal.

Volume Mount Errors

Several errors can occur during the volume mount procedure. RESTOR prints an error message and then repeats the current prompt.

Table 2-28 lists the RESTOR volume mount error messages and their meanings.

Table 2-28: RESTOR Volume Mount Error Messages

Message and Meaning
?DENSITY/PARITY CANNOT BE SET ON THIS DRIVE Specified parity or density is illegal on this type of hardware.
?INVALID BACKUP SET NAME The set name contains illegal characters.
?INVALID DATE

Table 2-28: RESTOR Volume Mount Error Messages (Cont.)

Message and Meaning
Date is not in dd-mmm-yy format or has passed.
?INVALID DENSITY SETTING Valid density settings are 800 and 1600 bpi.
?INVALID DEVICE OR NO UNIT NUMBER The input device specification refers to a device not configured on the current system, or the device unit number is not specified.
?INVALID PARITY SETTING Valid parity settings are ODD and EVEN.
?INVALID SWITCH SPECIFICATION The switch does not contain (as its first three characters) SCR.
?MAGTAPE NOT WRITE LOCKED During a RESTORE operation, the magnetic tape must be mounted write-locked.
?MAGTAPE SELECT ERROR The magnetic tape unit specified is not READY or is OFFLINE. Ready the unit or specify another unit.
?MAGTAPE WRITE LOCKED During a backup operation, the magnetic tape must be mounted write-enabled.
?THIS IS NOT THE RIGHT VOLUME Some volume data does not match with what RESTOR expected.
?UNABLE TO WRITE ON THIS TAPE RESTOR cannot write a label on the magnetic tape because of tape errors. Try another magnetic tape.
?YOU ARE NOT PRIVILEGED TO USE THIS VOLUME The volume's expiration date has not yet passed, and the user has insufficient privilege (and is not the volume's owner).

RESTOR Processing Errors

Errors can occur as the package selects, transfers, compares and deletes files and generates the listing file. An error can be a hardware error (for example, ?Device hung or write-locked), or a logic error action (for example, ?Protection violation).

If a logic error occurs during the selection process, RESTOR skips over the file or account causing the error. RESTOR prints an error message and then selects the next file or account according to the following rules:

- If an error occurs while RESTOR is looking up a file, RESTOR skips that file in that account and looks for the next file.
- If an error occurs during the search for an account (except for the final input volume), RESTOR terminates the search on that input volume and continues the search for that account on the next input volume.
- If an error occurs during the search for an account on the final input volume, RESTOR returns to the first input volume and searches for the next sequential account.
- If an error occurs in the final account on the final input volume, the selection process ends and RESTOR begins to transfer the selected files and accounts.

RESTOR contains error handling routines that allow you to correct or avoid problems that cause the ?Device hung or write-locked hardware error. When this error occurs, RESTOR prints the error message, followed by one or more valid request commands (RETRY or SKIP) in parentheses. After RESTOR issues an * prompt, you can respond with one of the suggested commands. A RETRY response causes RESTOR to attempt the failed operation again. For example, device hung errors may indicate:

- A transient hardware problem
- A device has gone off line
- A file or account contains bad blocks

You may be able to correct a transient hardware problem by using the RETRY command. When a device goes off line you can usually place the device back on line and respond to the * prompt by typing RETRY. Finally, if bad blocks in a file or account cause a device hung error, you can attempt to bypass the bad block area by using the SKIP command. In reply, RESTOR ends its search of the file or account containing the bad block, proceeds to look for the the next file or account, and then attempts to resume normal processing.

During a Restore operation, a bad block error at index load time usually means that the index file is corrupt. If you specified that RESTOR use the primary index file, RESTOR displays the following error message:

```
?Primary index load failed! Starting secondary load
```

RESTOR attempts to load the secondary index file (which may require you to mount a different volume).

If an error occurs during the secondary index load, RESTOR prints the following error message and aborts:

```
?Secondary index load failed.  Run aborted!!!
```

You must rerun RESTOR, using the auxiliary index file.

If you specified the auxiliary index file and a load error occurs, RESTOR prints the following error message and aborts:

```
?Auxiliary index load failed  --  filename.type due to <text> error.
Run aborted!!!
```

The <text> is the RSTS/E error that caused the load failure.

The bad block error message ?Data error on device that occurs during the Restore operation is fatal. RESTOR prints the following error message and aborts the run:

```
?Unexpected error -- data error on device
```

Logic errors during the transfer, deletion, and listing processes usually result from failure to open a file that RESTOR must transfer or delete. For example, the error message ?Protection violation means the file protection code prohibits you from deleting or transferring a certain file. Another error message , ?Supersede failure, indicates a failure to restore a file that already exists on the destination disk. This error can occur even if you did not request a supersede operation. The RESTOR package automatically skips over the file or account causing a logic error during transfer, deletion, or listing processes.

RESTOR error handling routines allow you to retry (RETRY) or skip (SKIP) over an error that occurs during the transfer, deletion, or listing process. The associated error message is ?Device hung or write-locked. RETRY and SKIP work in the same way here as they do during the selection process.

In addition to device hung errors, other errors can interrupt RESTOR, displaying error messages such as ?Can't find file or account or ?No room for user on device. If RESTOR displays one of these error messages while processing a listing file, type the IGNORE request command in response to the * prompt. This procedure forces RESTOR to disable future I/O to the listing file and to continue with the current operation.

The sequencing problem on magtape error can occur when restoring from magnetic tape. This error means that RESTOR read a magnetic tape that did not have the expected record number. When the sequencing error occurs, RESTOR usually prints several bad block error messages (as it tries and fails to space to the correct magnetic tape record) and then recovers automatically. You cannot restore the file in which the sequencing error occurred; however, you can restore the remainder of the files on the tape.

If RESTOR finds a bad block during a transfer operation that involves a RSTS/E (not a RESTOR) disk, RESTOR prints an error message. It copies the rest of the current file although the file is corrupt. If the bad block is in the work file, RESTOR aborts.

When RESTOR discovers a bad block on a backup disk, it allocates the bad block to the bad block file. This procedure prevents future use of the block for backup data. (You should be aware that RESTOR bad block files are not equivalent to the RSTS/E file BADB.SYS. You must reinitialize a backup disk before using it as a RSTS/E file-structured disk.)

During the transfer process, RESTOR often prints informational messages. These messages usually do not indicate errors, but inform you about files that are open (and subject to change) and files that have changed in length since selection. RESTOR transfers these files, but the copies may not be accurate. Other messages list files that RESTOR cannot transfer: those deleted since selection and those whose length is zero.

The RESTOR program displays a timed reminder after you type the PAUSE command and when it is waiting for a user response. Two minutes after you type the PAUSE command, which suspends the execution of a Restore operation, RESTOR prints the message IN PAUSE. The reminder appears only once. Similarly, RESTOR reminds you, after a two minute interval, that it is waiting for a user response. The error message ?Further response necessary does not appear during any of the initial dialogue questions. Like the PAUSE reminder, note that RESTOR does not repeat this message.

Restore File Example

RESTOR recreates a RSTS/E formatted disk from a magnetic tape or disk Backup Set. The example in the following section shows how to restore an RK06 disk from a magnetic tape Backup Set. The section " Restore

Listing File" reproduces and describes the listing file that results from the Restore run in the following section.

Restore Terminal Printout

To duplicate the following example, run the RESTOR program, and respond to the dialogue questions as shown. The letters in the printout are keyed to the description that follows:

```
   $ RUN RESTOR$:RESTOR
   RESTOR V9.0 RSTS V9.0

 a  RES[TORE], LOA[DINDEX] OR LIS[T] ? RES

 b  WORK FILE NAME <_SY:[  1,110]B08SEP.J16> ? RESTOR.WKF

 c  LISTING FILE <_KB:> ? RESTOR.LST

 d  INDEX FILE<PRIMARY> ? <LF>

 e  FROM DEVICE<_MM:> ? MM: <RET>

 f  FROM FILES<[  1,110]*.*> ? [2,110]BACKUP.BAS,BACKTO.BAS

 g  TO DISK<_SY:[*,*]> ? DM1:

 h  BEGIN AT<[*,*]*.*> ? <RET>

 i  ENTER ACCOUNTS<YES> ? NO

 j  SUPERSEDE FILES<NONE> ? <LF>

 k  COMPARE FILE<NONE> ? <RET>

 l  * <RET>

 m  PLEASE ENTER BACKUP SET NAME<RESTOR> - BACKUP
    PLEASE ENTER DENSITY IN BPI<800> - <LF>
    PLEASE ENTER THE PARITY<ODD> - <LF>
    MOUNT    DEVICE:       _MM    :
                 ID:       BACKUP
               SEQ#:       INDEX
            DENSITY:       800 BPI
             PARITY:       ODD

 n          PLEASE MOUNT VOLUME WRITE LOCKED!

 o  DEVICE? MM1:
```

```
 p  *STATUS

    PHASE    : SELECT
    VOLUME # : 2
    ERRORS   : 0

    CURRENT VOLUME  : _MM     :
    CURRENT ACCOUNT : _DM1    :[2,110]

    ELAPSED TIME    : 1 SECONDS
    CPU TIME        : .4 SECONDS
    KCTS            : 64

    *
 q  DISMOUNT DEVICE:        _MM1    :
    MOUNT    DEVICE:        _MM     :
                 ID:        BACKUP
               SEQ#:        1
            DENSITY:        800 BPI
             PARITY:        ODD

 r          PLEASE MOUNT VOLUME WRITE LOCKED!

 s  DEVICE? MM0:
    *

 t  DISMOUNT DEVICE:    _MM0    :
    *
    $
```

The explanation of the Restore example follows:

Line	**Description**
a	Type RES in response to the initial RESTOR prompt, and then press RETURN. This starts the RESTOR program, which prints a set of dialogue questions.
b	Specify a work file name that does not have a .TMP file type such as RESTOR.WKF. RESTOR deletes any work file created during a Restore operation that has a temporary file type of .TMP.
c	Create the RESTOR.LST file to store the listing file generated from this Restore operation. RESTOR prints the listing file at your terminal (KB:) if you accept the default.
d	Press LINE FEED to have RESTOR transfer the primary index file, located on the last volume of the Backup Set, to the destination disk.

Line	Description
e	Type the magnetic tape device type MM:, MS:, or MT:.
f	Specify the files for RESTOR to recreate on the output disk. In this case, RESTOR returns the two files [2,110]BACKUP.BAS and BACKTO.BAS to the RK06 disk.
g	Identify the disk to which you plan to copy the files contained on the magnetic tape Backup Set. Type DM1:, which is the device specification of the RK06 disk.
h	Have RESTOR begin processing the first file it finds on the Backup Set by pressing RETURN to accept the default.
i	Type NO to make sure the files that exist on the Backup Set but do not exist on the output disk are not reproduced.
j	Press LINE FEED to have files superseded. That is, you do not want any files on the Backup Set to overwrite the same files on the output disk.
k	Accept the default; you do not want any files compared.
l	RESTOR prints an * prompt after you complete the dialogue. This indicates the program is ready to accept interruption commands. Ignore the prompt at this point; you have no reason to interrupt processing.
m	Type the correct name of the Backup Set. (Whatever name you enter in response to the WORK FILE NAME question, RESTOR uses for the default response to PLEASE ENTER BACKUP SET NAME.) After you press LINE FEED in response to the next two questions, RESTOR prints a summary of the label information contained on the Backup Set.
n	The message asks you to mount the last volume of the Backup Set, the one containing the primary index file. Mount it without the write-enable ring. If you have not created an index file with Loadindex, created an auxiliary index file by copying the work file using PIP.SAV, or have not given the work file a file type other than .TMP during the Backup operation, you must repeatedly load the last backup volume for each Restore operation. Use Loadindex to create an easily accessible index and eliminate this inconvenience.
o	Enter the device specification of the last volume of the Backup Set. RESTOR needs the index that is on the last volume to begin the Restore operation. (Note that SEQ #: tells you which volume you should load; the index volume MM1: is the correct volume to load in this case.)

Line	Description
p	Respond to the * prompt by typing STATUS, which summarizes the operational status of the Restore run. The summary indicates that RESTOR:

- o Is in the select PHASE
- o Is processing the last volume of the Backup Set
- o Has found no errors
- o Is performed from a magnetic tape Backup Set
- o Is processing account [2,110]
- o Has consumed 1 second of elapsed time, 0.4 seconds of CPU time, and 64 kilo-core-ticks

After printing the summary, RESTOR returns to normal processing, as the * prompt indicates.

q The dismount summary that RESTOR prints when it finishes processing the Backup Set volume that contains the primary index file.

r RESTOR asks you to mount the remaining Backup volume.

s Type the device specification of the device on which you have mounted the second volume of the Backup Set. After you press RETURN, RESTOR selects the specified files from the mounted volume and transfers them to the output disk.

t After RESTOR prints the dismount message for the second volume, it prints the * prompt.

Restore Listing File

The listing file summarizes the phases of the Restore operation and provides directory information on the restored files. The letters in the example are keyed to the description that follows:

```
                            Restore Run Listing

  a  Restore from '_MM   :' to '_DM    :[*,*]

     Run started on 13-Jun-85 at 11:57 AM

     Work-File is _SY    :[1,110]RESTOR.WKF/M0:256
```

```
   Transfer    : [2,110]BACKUP.BAS,[2,110]BACKTO.BAS

   Begin at    : [*,*]*.*

   Superseded : <none>

   Compare     : <none>

b  PHASE     : LIST COMPLETE
   ERRORS     : 0

   ELAPSED TIME : 1 SECONDS
   CPU TIME      : .5 SECONDS
   KCTS          : 72

c  PHASE     : MOUNT / DISMOUNT COMPLETE
   ERRORS     :0

   ELAPSED TIME : 61 SECONDS
   CPU TIME      : .9 SECONDS
   KCTS          : 160

d  PHASE     : INDEX LOAD COMPLETE
   ERRORS     : 0

   CURRENT VOLUME : _MM1   :

   ELAPSED TIME : 1 SECONDS
   CPU TIME      : .7 SECONDS
   KCTS          : 107

e  PHASE     :  SELECT COMPLETE
   VOLUME # : 2
   ERRORS     : 0

   ELAPSED TIME : 2 SECONDS
   CPU TIME      : .8 SECONDS
   KCTS          : 128

f  PHASE     : MOUNT / DISMOUNT COMPLETE
   ERRORS     : 0

   ELAPSED TIME : 24 SECONDS
   CPU TIME      : .8 SECONDS
   KCTS          : 144

g  PHASE     : TRANSFER COMPLETE
   VOLUME # : 1
   ACCOUNTS : 1
   FILES      : 2
   BLOCKS     : 289
```

```
    ERRORS    : 0

    CURRENT VOLUME   : _MM0   :
    CURRENT FILE     : _MM0   :

    ELAPSED TIME : 50 SECONDS
    CPU TIME     : 1.5 SECONDS
    KCTS         : 254

h   PHASE    : MOUNT / DISMOUNT COMPLETE
    ERRORS    : 0

    ELAPSED TIME : 15 SECONDS
    CPU TIME     : .2 SECONDS
    KCTS         : 32

i   Backup Set Name : BACKUP             Backup Device : _MM
    Volume Sequence # : 1                Owner : [1,110]
    Creation Date : 13-Jun-85            Expiration Date : 13-Jun-85
    Density :  800 BPI                   Parity : ODD

j     Account : [2,110]                  Quota : 0
      Clustersize : 4

                                   Creation             Access
     Name  Ext  Size  Prot     Date        Time          Date     Clu  RTS    TSCD
    BACKUP.BAS  121   42   01-Jun-85 09:56 AM  01-Jun-85    8 BASIC  T
    BACKTO.BAS  168   42   01-Jun-85 04:36 PM  01-Jun-85    8 BASIC  T

k    Account total of 289 blocks in 2 files on account [2,110]
l    ***** Account continued on next Volume *****

m   Volume total of 289 blocks in 2 files in 1 accounts on volume _MM

n   Backup Set Name : BACKUP             Backup Device : _MM
    Volume Sequence # : 2(Index)         Owner : [1,110]
n   Creation Date : 13-Jun-85            Expiration Date : 13-Jun-85
    Density :  800 BPI                   Parity : ODD

o   <NO TRANSFER SELECTED>

p   Volume total of 0 blocks in 0 files in 0 accounts on volume _MM

q   RUN total of 289 blocks in 2 files on 0 accounts on 2 Volumes
    RUN total of 0 errors

r   PHASE    : LIST COMPLETE
    VOLUME # : 1
    ERRORS    : 0
```

```
ELAPSED TIME : 3 SECONDS
CPU TIME     : 1.1 SECONDS
KCTS         : 162
```

The explanation of the Restore listing file follows:

Line	Description
a	Summarizes the RESTOR dialogue that indicates you are restoring two files in account [2,110], BACKUP.BAS and BACKTO.BAS. The summary also shows that RESTOR starts processing at the beginning of the file, does not supersede any files, and does not compare any files.
b	Supplies accounting statistics about writing the dialogue summary into the listing file.
c	Summarizes the mount/dismount phase, during which you mounted the backup index volume.
d	Describes the loadindex phase.
e	Summarizes the selection phase statistics.
f	Indicates that RESTOR finished selecting data from the backup index volume and requested that it be dismounted. The summary also describes the various times required to complete the dismount operation and mount the other volume in the Backup Set.
g	Describes the transfer phase which shows that two files selected for transfer were on volume 1.
h	Displays the mount/dismount summary, which is caused by the dismount request that occurs at the end of the transfer.
i	Lists the label information for volume 1 after displaying each operational phase.
j	Prints the directory information for each file and account for that volume. This section includes the account, quota, and cluster size for each affected account and lists each restored file by name. The block size, protection code, date and time of creation, date of last access, cluster size, and associated run-time system are listed for each file. The TSCD column contains a T for each restored file because you specified only that they only be transferred, not superseded (S), compared (C), or deleted (D).
k	Indicates that a total of 289 blocks in 2 files were transferred for account [2,110].

Line	Description
l	States that account [2,110] is continued on the second volume.
m	Displays the block, file, and account summary for the entire volume.
n	Lists label information for volume 2.
o	Notifies you that no files or accounts were transferred. The account message, displayed at k, does not appear for volume 2 because no accounts were transferred from volume 2.
p	Displays the volume summary information for each volume of the Backup Set, even when the totals are zero.
q	Reports the total statistics for the run.
r	Supplies statistics that describe the listing phase that generated the listing file.

Loadindex File Example

Loadindex allows you to copy a primary index file from a Backup Set to a RSTS/E formatted disk. To perform the transfer, run the RESTOR program, type Loadindex in response to the initial RESTOR prompt, and answer the resulting questions. During the dialogue, you specify the name and location of the Backup Set and the file into which the program is to copy the index. RESTOR then prompts you to mount the Backup Set index file, copies the index when the device is ready, and requests the dismount of the index file volume. Finally, RESTOR prints a listing file that summarizes the Loadindex operation.

When you perform a series of restores, you must have access to the index file located on the last volume of the Backup Set. If you have not created an index file with Loadindex, created an auxiliary index file by copying the work file using PIP.SAV, or have given the work file a file type other than .TMP during the Backup operation, you must repeatedly load the last backup volume for each Restore operation. Use Loadindex to create an easily accessible index and eliminate this inconvenience.

A discussion follows both the Loadindex example in the following section and the Loadindex listing file example in the section "Loadindex Listing File."

Loadindex Terminal Printout

Log in to the system under account [1,110] and run RESTOR. The letters in the sample printout are keyed to the description that follows:

```
   $ RUN RESTOR$:RESTOR
   RESTOR V9.0 RSTS V9.0

a  RES[TORE], LOA[DINDEX] OR LIS[T] ? LOA

b  WORK FILE NAME<_SY:[  1,110]B08SEP.TMP> ? LOADIN.WKF/MO:256

c  LISTING FILE<_KB:> ? LOADIN.LST

d  FROM DEVICE<_MM:> ? MM1:<RET>

e  TO FILE<_SY:[  1,110]BIND08.IND> ? BACIND.IND

f  * <RET>

g  PLEASE ENTER BACKUP SET NAME<LOADIN> - BACKUP

h  PLEASE ENTER DENSITY IN BPI<800> - <RET>

i  PLEASE ENTER THE PARITY <ODD> - <RET>

j  MOUNT    DEVICE:         _MM    :
                ID:         BACKUP
              SEQ#:         INDEX
           DENSITY:         800 BPI
            PARITY:         ODD
           PLEASE MOUNT VOLUME WRITE LOCKED!

l  DEVICE? MM1:

m  * <RET>

n  DISMOUNT DEVICE: _MM1:

o  $
```

The explanation of the Loadindex example follows:

Line	Description
a	Run the Loadindex dialogue by typing LOA or LOADINDEX in response to the initial RESTOR question. RESTOR understands only the first three letters of the option.

Line	Description
b	Type LOADIN.WKF/MO:256 to store the work file on the public disk structure. Giving the specified file a file type other than .TMP allows you to use the work file as an auxiliary index file at some later date. However, if you accept the default or specify a work file with a .TMP file type, RESTOR deletes the file at the end of the Loadindex run. The MODE 256% switch enables random data caching that facilitates data transfers within the LOADIN.WKF work file. For an explanation of the MODE 256% switch, see the *RSTS/E Programming Manual*.
c	Specify LOADIN.LST as the file into which RESTOR prints the listing file. RESTOR displays the listing file on your terminal if you accept the default.
d	Enter MM:, MS:, or MT: and press RETURN.
e	Have RESTOR copy the backup index to the file BACIND.IND.
f	Ignore the * prompt, which indicates that RESTOR is ready to accept interruption commands.
g	Respond by typing the Backup Set Name, in this case, BACKUP.
h	Press RETURN to accept the default density of 800 bpi for the magnetic tape.
i	Press RETURN to accept the ODD parity default setting for the magnetic tape.
j	Check the magnetic tape label information summary.
k	Make sure the magnetic tape is mounted write-locked and is at its load point.
l	Type MM1:, which is the mnemonic name and unit number of device on which the magnetic tape is mounted.
m	If you want to suspend processing temporarily at this point, type a RESTOR interrupt command. Because the operation is running smoothly, ignore the * prompt. Before RESTOR issues the dismount message at n, it copies the index file to the specified area.
n	Dismount the Backup index volume by typing the mnemonic name and unit number (followed by a colon) of the device on which the magnetic tape is mounted.
o	After RESTOR dismounts the Backup index volume, it gives the * prompt and returns control to your keyboard monitor.

Loadindex Listing File

The Loadindex listing file summarizes the Loadindex dialogue, provides statistics on each phase, and lists the operations performed on each selected file. The letters in the file are keyed to the descriptions that follow:

```
                    Restore Run Listing

a  Loadindex from '_MM    :

   Run started on 13-Jun-85 at 12:03 PM

   Work-File is _SY    :[1,110]LOADIN.WKF/MO:256

   To file  : _SY    :[1,110]BACIND.IND

b  PHASE   : LIST COMPLETE
   ERRORS : 0

   ELAPSED TIME : 1 SECONDS
   CPU TIME     : .5 SECONDS
   KCTS         : 72

c  PHASE   : MOUNT / DISMOUNT COMPLETE
   ERRORS   : 0

   ELAPSED TIME : 60 SECONDS
   CPU TIME     : .9 SECONDS
   KCTS         : 160

d  PHASE   : INDEX LOAD COMPLETE
   ERRORS   : 0

   CURRENT VOLUME  : _MM1    :

   ELAPSED TIME : 1 SECONDS
   CPU TIME     : .3 SECONDS
   KCTS         : 46

e  PHASE   : MOUNT / DISMOUNT COMPLETE
   ERRORS   : 0

   ELAPSED TIME : 12 SECONDS
   CPU TIME     : .2 SECONDS
   KCTS         : 32
```

The explanation of the Loadindex listing file follows:

Line	Description
a	Summarizes the Loadindex dialogue described in the previous section. The summary shows the date and time of the Loadindex operation, the work file name, the name of the output file, and that the index resides on magnetic tape.
b	Provides time and error statistics about writing the dialogue summary into the listing file.
c	Lists the statistical details concerning the mount phase of the Backup Set index volume.
d	Describes which volume contains the index, the times required to complete the load, and the errors found during the index load phase.
e	Summarizes the dismount of the Backup Set volume.

Index File Listing Example

List initiates a set of dialogue questions like the other RESTOR modes. After you answer these questions, RESTOR creates a copy of the Backup Set index file, an index created by Loadindex, or, if specified during the Backup operation, an auxiliary index file. These files contain a directory listing of the Backup Set and a log of any errors found during the Backup operation. Before it can copy the index file, RESTOR asks for the index name and location and then asks where you want the index listed. As options, you can either list the index file at your terminal or place it in another file. Like the other RESTOR modes, a summary of the dialogue is written to the listing file.

The example in the following section describes how to generate a list of an auxiliary index file.

List Terminal Printout

Log in to your system, run RESTOR, and then respond to the dialogue questions as shown. The letters in the printout are keyed to the description that follows:

```
$ RUN RESTOR$:RESTOR
RESTOR V9.0 RSTS V9.0
```

```
a  RES[TORE], LOA[DINDEX] OR LIS[T] ? LIS

b  WORK FILE NAME_SY:[  1,110]B08SEP.TMP> ? LIST.WKF/MO:256

c  LISTING FILE<_KB:> ? LIST.LST

d  INDEX FILE<PRIMARY> ? BACKUP.WKF

e  * <RET>

f  $
```

The explanation of the List example follows:

Line	Description
a	Type LIS and press RETURN in response to the first RESTOR question.
b	Specify LIST.WRK as the work file and attach the MODE 256% switch to optimize file transfer. See the *RSTS/E Programming Manual* for a description of MODE 256%.
c	Have RESTOR copy the listing file to LIST.LST. If you accept the default (KB:), RESTOR prints the index file at your terminal.
d	Type BACK.WRK to indicate which index file you want RESTOR to copy. In this case, specify the BACK.WRK file. Because you used a file type other than .TMP, namely .WRK, when you specified the work file name, RESTOR created BACK.WRK as a permanent auxiliary index file. After you completed the dialogue, RESTOR copied the BACK.WRK file to the listing file, LIST.LST. If you accept the default, RESTOR prints the FROM DEVICE question. In response, specify the device type on which the primary index volume is mounted.
e	Respond to the * prompt by pressing RETURN. The * prompt appears after you complete the dialogue and indicates that RESTOR is ready to accept interruption commands. After your response, RESTOR copies BACK.WRK to the LIST.LST listing file.
f	After copying the files, RESTOR displays the $ prompt and returns you to your keyboard monitor.

List Listing File

The List listing file summarizes the List dialogue, provides statistics on the each phase of the operation, and prints a copy of the Backup index file. The following example is the listing file for the List operation described in the previous section. The letters in this listing file are keyed to the description that follows:

```
                 Restore Run Listing

     List

  a  Run started on 13-Jun-85 at 12:08 PM

     Work-File is _SY    :[1,110]LIST  .WKF/MO:256

  b  PHASE    : LIST COMPLETE
     ERRORS    : 0

     ELAPSED TIME : 1 SECONDS
     CPU TIME     : .5 SECONDS
     KCTS         : 72

  c  PHASE    : INDEX LOAD COMPLETE
     ERRORS    : 0

     ELAPSED TIME : 2 SECONDS
     CPU TIME     : .4 SECONDS
     KCTS         : 55

  d  Backup Set Name : BACKUP               Backup Device : _MM
     Volume Sequence # : 1                  Owner : [1,110]
     Creation Date : 13-Jun-85              Expiration Date : 13-Jun-86
     Density :  800 BPI                     Parity : ODD

       Account : [1,110]                    Quota : 0
       Clustersize : 4

                                   Creation           Access
      Name  Ext  Size Prot     Date       Time          Date    Clu  RTS   TSCD
     SAVE  .DOC    10  60  01-Jun-85 05:01 PM 01-Jun-85   8 ...RSX*T
     OPEN  .TMP     0  60  10-Jun-85 10:46 AM 10-Jun-85   4 BAS4F
     ADDR  .DAT  2000  60  09-Apr-85 10:46 AM 09-Apr-85   4 BAS4F    T

        Errors :
  e       DATA UNRELIABLE - FILE OPENED BY ANOTHER USER on FILE
                  Total of 1 error encountered on FILE
     RT11   .RTS    20C  60  08-May-85 04:39 PM  08-May-85   4 RSTS  T
     SAVE   .RNO     9   60  09-Apr-85 04:59 PM  09-Apr-85   8 TECO  T
     UTILTY.BAS     47   60  13-Jun-85 10:44 AM  13-Jun-85   8 BASIC T
     DIRECT.BAS     56   60  13-Jun-85 11:24 AM  13-Jun-85   8 BAS4F T
```

```
     * Attributes associated with this file

    Account total of 2142 blocks in 7 files on account [1,110]
        Total of 1 error encountered on ACCOUNT

     Account : [2,110]                  Quota : 0
     Clustersize : 4

f
                                 Creation          Access
    Name   Ext  Size  Prot     Date     Time         Date    Clu   RTS   TSCD
   BACKUP.BAS   121    42  21-Apr-85 09:56 AM 21-Apr-85    8 BASIC T
   BACFRM.BAS   158    42  09-Apr-85 04:49 PM 09-Apr-85    8 BASIC T
   BACKTO.BAS   168    42  13-Jun-85 04:36 PM 13-Jun-85    8 BASIC T
   MASTER.DAT  6904    60  08-May-85 10:55 AM 08-May-85    4 BAS4F T C
             (of 9000)

       ***** File continued on next Volume *****

     Account total of 7351 blocks in 4 files on account [2,110]
     ***** Account continued on next Volume *****

g  Volume total of 9493 blocks in 11 files in 2 accounts on volume _MM
     Total of 1 error encountered on VOLUME

h  Backup Set Name : BACKUP             Backup Device : _MM
   Volume Sequence # : 2(Index)         Owner : [1,110]
   Creation Date : 08-Sep-83            Expiration Date : 08-Sep-84
   Density :  800 BPI                   Parity : ODD

     Account : [2,110]                  Quota : 0
     Clustersize : 4

i
                                  Creation         Access
    Name   Ext  Size  Prot     Date      Time         Date   Clu   RTS   TSCD
   MASTER.DAT  2096   60  08-Sep-83 10:55 AM 08-Sep-83   4 BAS4F  T CD
                 of 9000)

     Account total of 2096 blocks in 1 files on account [2,110]

j  Volume total of 2096 blocks in 1 files in 1 accounts on volume _MM

k  RUN total of 11589 blocks in 11 files on 2 accounts on 2 Volumes
   RUN total of 1 errors
```

```
l  PHASE    : LIST COMPLETE
   VOLUME # : 2
   ACCOUNTS : 2
   FILES    : 11
   BLOCKS   : 11589
   ERRORS   : 0

m  ELAPSED TIME : 2 SECONDS
   CPU TIME     : 1.7 SECONDS
   KCTS         : 252
```

The explanation of the List listing file follows:

Line	Description
a	Summarizes the List dialogue. This includes the name of the operation, the run date and time, and the work file name.
b	Provides statistics for the listing phase on writing the List dialogue summary into the listing file. Note that RESTOR rounds off elapsed time to whole seconds and CPU time to tenths of seconds. The elapsed time can therefore read zero while the CPU time is not zero.
c	Summarizes the index load phase that prints the elapsed time, CPU time, and the kilo-core ticks needed to load the index into the listing file.
d	Prints the Backup directory information extracted from the index file after it completes the List dialogue summary and reports on the various phases of the List operation. The directory report begins by listing the label information for the first volume of the Backup Set. It then displays a list of the files for each account on the directory. At the end of each account report, RESTOR prints the total number of blocks used by the files listed and the number of errors found on the account.

In addition to supplying the file name, RESTOR includes in the account report the file block size, protection code, date and time of creation, date of last access, the cluster size, and the associated run-time system. The TSCD column contains a T for each restored file, an S for each superseded file, a C for a compared file, and a D for each deleted file. An asterisk beside a letter in the TSCD column indicates that the file has attributes associated with it. Zero-length files, such as OPEN.TMP in account [1,110], are not processed; thus, no letter is assigned to them in the TSCD column. Any errors in the dialogue sequence appear in the index.

Line	Description
e	Prints the DATA UNRELIABLE message immediately after the file that caused the error. After enumerating the error messages, RESTOR prints the total number of errors detected in the file.
f	Summarizes the files contained in account [2,110].
g	Prints the total number of blocks, files, and accounts on the entire volume at the end of the account report.
h	Prints label information found on the second volume of the Backup Set.
i	Prints the account summary for volume 2. If there is more than one volume in the Backup Set, as there is in this example, RESTOR prints account and file information in the same format for the other volumes in the set.
j	Summarizes the account that was transferred for the second volume of the set. Because there is only one account report, the volume and account totals are equal.
k	Displays a summary of the blocks consumed by all the files on both volumes of the Backup Set. This completes the account and file statistics.
l	Informs you that there are two volumes in the Backup Set, two accounts in the directory, and 11 files in the two accounts that require 11,589 blocks of disk space. In addition, an item in the list phase report indicates that there were no errors found during the printing of the list phase summary.
m	Prints the time required to complete the list phase.

Obtaining System Status Reports: The SYSTAT Program

The SYSTAT program provides current system information about jobs, devices, disks, buffers, run-time systems, resident libraries, message receivers, open files, and memory allocation. You can:

- o Display the report on a terminal
- o Send the report to a line printer
- o Write the report to a file

Privilege Required

To run the SYSTAT program, you need the following:

- o WREAD privilege to use the /O and /W switches to obtain information about open files
- o SYSIO privilege to obtain status of temporary privileges for jobs
- o TUNE privilege to obtain maximum size, priority, and run burst information about jobs

Running the SYSTAT Program

To run SYSTAT, type:

```
$ RUN $SYSTAT
```

The program responds with a header and a prompt:

```
$SYSTAT V9.0 RSTS V9.0
Output Status to?
```

At this point you can specify any RSTS/E device or a file specification for the status report output. Possible replies are:

Output Response	Meaning
<RET>	Displays report on your terminal. This is equivalent to KB: response.
LP:	Sends the report to the system line printer if only one line printer is on the system or to line printer 0 if multiple line printers are on the system.
LPn:	Sends the report to line printer n if that printer is not currently in use.
KB:	Displays the report on your terminal. This is equivalent to pressing RETURN for a response.
KBn:	Displays the report on terminal n in the system if that terminal is on line and not currently in use.
dev:filename.type	Writes the report to the file specified. The default device is the system device. No file type is appended unless specified.
?	Writes the report to a file on the public structure, and displays the file name on your terminal. SYSTAT names the file according to the current date and time of day; its type is .RPT.

An example of an output file created by the ? response is F04N59.RPT. The file name has four parts:

- A letter, from A to L, denoting the month according to its alphabetical position; here, F denotes June.
- Two digits, from 01 to 31, denoting the day of the month; here, 04 denotes the fourth day.
- A letter, from A to X, denoting the hour from 00 to 23; here, N denotes the 14th hour (2:00 p.m.).
- Two numbers, from 00 to 59, denoting the minute of the hour; here, 59 denotes the 59th minute.

To obtain a partial system status report, follow the device or file name specification with one or more of the switches listed in Table 2-29.

Table 2-29: SYSTAT Switches

Switch	Format	Meaning
Attached	/A	Report status of attached jobs only.
Busy devices	/B	Report status of busy devices only.
Memory allocation	/C	Report memory allocation on system.
Disk structure	/D	Report status of disk structure only.
Free buffers	/F	Report status of free buffers only.
Terminal	/Kn	Display report on terminal n in the system.
Resident libraries	/L	Report status of resident libraries only.
Message receivers	/M	Report status of message receivers only.
Job	/n	Report status of job n only.
Account	/n,m	Report status of account [n,m] only.
Project	/n,*	Report status of jobs having project number n only.
Programmer	_*,n	Report status of jobs having programmer number n only.
	null	Report complete system status to include jobs, run-time systems, resident libraries, busy devices, disk structure, free buffers, and message receivers.
Open files	/O	Report all nonsystem files that are open.
	/O:dev	Report all nonsystem files on a specific device that are open.

Table 2-29: SYSTAT Switches (Cont.)

Switch	Format	Meaning
	/W	Report all nonsystem files that are open, the jobs accessing them, and the block number being accessed.
	/W:dev	Report all nonsystem files that are open on a specific device, the jobs accessing them, and the block number being accessed.
Run-time systems	/R	Report status of run-time system only.
Jobs	/S	Report status of jobs only.
Detached	/U	Report status of detached jobs only.
Logged out	/0,0	Report status of jobs not logged into the system only.

You can specify switches separately or in any combination. If you specify multiple switches, use only one slash (/).

The following examples show some typical SYSTAT switches:

Example	Result

`$ RUN $SYSTAT`
`SYSTAT V9.0 RSTS V9.0`
`Output Status to? STAT`
	Writes complete SYSTAT report in the file STAT in your current account on the public structure.

`$ RUN $SYSTAT`
`SYSTAT V9.0 RSTS V9.0`
`Output Status to? LP:/3`
	Sends status report for job 3 to the line printer.

`$ RUN $SYSTAT`
`SYSTAT V9.0 RSTS V9.0`
`Output Status to? /D`
	Displays disk structure report on your terminal.

`$ RUN $SYSTAT`
`SYSTAT V9.0 RSTS V9.0`
`Output Status to? /SF`
	Displays job status and free buffer reports on your terminal.

Contents of the Full Status Report

The full SYSTAT report contains seven parts:

- Jobs
- Busy devices
- Disk structure
- Free buffers
- Run-time systems
- Resident libraries
- Message receivers

This section contains a complete SYSTAT report. For purposes of clarity, an explanation follows each part of the SYSTAT report.

```
$ RUN $SYSTAT
SYSTAT V9.0  RSTS  V9.0
Output Status to? <RET>

RSTS V9.0  status at 13-Jun-85, 09:38 AM Up: 67:52:51

Job   Who    Where   What     Size    State   Run-Time  Pri/RB  RTS
 1    1,2    Det     ERRCPY   11/32K  SR        2:58.2    0/6   ...RSX
 2    1,2    Det     OPSRUN   22/32K  SL          15.4   -8/6   ...RSX
 3    1,2    Det     QUMRUN   23/32K  SL           9.1    0/6   ...RSX
 4    1,2    Det     SPLIDL   22/32K  SL           3.5   -8/9   ...RSX
 5    1,2    Det     BATIDL   20/32K  SL           0.1   -8/6   ...RSX
 6    1,2    Det     BATIDL   20/32K  SL           0.1   -8/6   ...RSX
 7    1,2    Det     BATIDL   20/32K  SL           3.6   -8/6   ...RSX
 8    1,2    Det     NPKDVR    9/32K  SL          17.0   -8/6   ...RSX
 9    1,223  KB27    ...EDT    7/32K  KB        9:40.3   -4/6   ...RSX
10    1,2    Det     EVTLOG   26/32K  SL        3:26.2   -8/6   ...RSX
11    1,2    Det     MAILSN   27/32K  SR          50.2   -8/6   ...RSX
12    1,223  KB52    ...NET   16/32K  SL        1:02.4   -8/6   ...RSX
13    1,229  KB41    DCL       4/32K  ^C        2:16.8   -8/6   DCL
14    2,214  KB32    DCL       4/32K  ^C          11.0   -8/6   DCL
15    1,214  KB51    SYSTAT   20/32K  R Lck     1:16.4   -8/6   ...RSX
16    1,2    Det     PBS..2   14/32K  SL           6.4   -8/6   ...RSX
```

The jobs report provides information about currently active jobs. Items reported are:

Item	Description
Job	Active job number. If a job is running with temporary privilege, the job number includes a plus sign (+). If the job is running without temporary privilege, the job number includes a minus sign (-).
Who	Account number under which the job runs.
Where	Number of the keyboard from which the job is running. An * character follows the keyboard number if the job logged in by via a dial-up line. DET appears in place of the keyboard number for jobs running detached from a keyboard. The abbreviation PxJy appears for a job running on a pseudo keyboard. Px identifies pseudo keyboard number x and Jy denotes job number y, under which the controlling job is running.
What	Program name.
Size	Current size of the job and the size to which it can expand.
State	Current condition of the job. See Table 2-30 for the list of abbreviations used in this column.
Run-Time	Total amount of central processor run time used, in hours, minutes, seconds, and tenths of seconds.
Pri/RB	Current job priority and run burst units.
RTS	Name of the run-time system under which the job is running.

```
Busy Devices:
Device      Job    Why
PK0         17     Open
PK1         17     Open
DMP-0.0     TRN    AS+Open
DMP-1.0     TRN    AS+Open
LP0         23     Open
```

The busy devices report provides information about devices that are assigned or opened by a specific user. Items reported are:

Item	Description
Device	Device specification.
Job	Number of the job owning the device.
Why	Condition of the device. See Table 2-30 for the list of abbreviations used in this column.

```
Disk Structure:
Dsk Open    Size       Free    Clu Err Name    Level  Comments
DM1    0   53768    1891   3%   1   8  WRENCH   1.1   Pri, DLW
DR0    0  500352   69872  13%   8   0  H        1.2   Pri, R-O, DLW, DP
DR1   64  131648    8072   6%   4   0  ARK      1.2   Pub, DLW
DR3   25  500352   23672   4%   8   0  W        1.2   Pri, DLW
DR4    0  242572  163772  67%   4   0  M        1.2   Pri, DLW
DR5    0  131648  107500  81%   4   0  U        1.2   Pri, DLW
```

The disk structure report provides information about each disk in use on the system. Items reported are:

Item	Description
Dsk	Disk name (device specification).
Open	Number of open files on the disk.
Size	Total number of blocks accessible on the disk.
Free	Number of free 512-byte blocks and the percentage of free blocks.
Clu	Pack cluster size.
Err	Count of disk hardware errors.
Name	Pack identification or system logical name (if any) assigned to the device.
Level	RSTS System Device file structure.
Comments	Comments on its status. The program reports disks assigned as non-file-structured and private and the report can include the job number and the designation "Dirty", if appropriate. See Table 2-30 for the list of abbreviations used in this column.

```
  General   FIP                         Hung
  Buffers   Buffers   Jobs/Jobmax     TTY's    Errors
    307        75       26/50           0        294
```

The free buffer report provides information about the number of small and FIle Processor (FIP) buffers available. Items reported are:

Item	Description
General Buffers	Number of small 16-word buffers that are generally available.
FIP Buffers	Number of buffers available from the FIP buffer pool.
Jobs/Jobmax	Number of jobs currently running and the maximum number allowed to run (two numbers separated by a slash).
Hung TTY's	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.
Errors	Total number of errors logged on the system.

```
  Run-Time Systems:
   Name    Ext   Dev     Size     Users  Comments
  ...RSX                 0(32)K    17    Monitor
  RSX      TSK   DR1:    1(28)K     0    Primary Perm, Addr:71, KBM
  DCL      COM   DR1:   27(4)K      4    Temp, Addr:75, DF KBM
  RT11     SAV   DR1:    4(28)K     1    Temp, Addr:106, KBM, CSZ, EMT:255
  BAS4F    BAC   DR1:   15(16)K    10    Perm, Addr:62, KBM, CSZ
  TECO     TEC   DR1:   10(20)K     0    Temp, Addr:379:KBM
  BAS4AL   BAC   DR1:   17(12)K     0    Non-Res, KBM, CSZ
  BASIC    BAC   DR1:   13(16)K     2    Temp, Addr:206, KBM, CSZ
```

The run-time systems report provides information about run-time systems. Items reported are:

Item	Description
Name	Name of the run-time system.
Ext	Default file type for (executable) files created by that run-time system.
Dev	Device location.
Size	Size of the run-time system, in K words.

Item	Description
Users	Number of user jobs currently executing under control of the run-time system.
Comments	Information about the status of the run-time system. See Table 2-30 for the list of abbreviations used in this column.

```
Resident Libraries:
 Name   Prot          Acct        Size  Users   Comments
HISTOG < 42>        [  1,2   ]   34K     0    Perm, Addr:1230
EDT    < 42>   DR1:[  0,1   ]   21K     2    Temp, Addr:1209
EDTV3  < 42>   DR1:[  0,1   ]   38K     0    Temp, Addr:1171
RMSRES < 42>   DR1:[  0,1   ]    4K     2    Temp, Addr:1167
RMSLBB < 42>   DR1:[  0,1   ]    3K     1    Temp, Addr:1164
RMSLBA < 42>   DR1:[  0,1   ]    4K     1    Temp, Addr:1160
RMSLBD < 42>   DR1:[  0,1   ]    2K     0    Temp, Addr:1158
RMSLBC < 42>   DR1:[  0,1   ]    2K     0    Non-Res, Addr:1156
RMSLBE < 42>   DR1:[  0,1   ]    3K     0    Temp, Addr:1153
RMSLBF < 42>   DR1:[  0,1   ]    4K     0    Temp, Addr:1149
B2MRES < 42>   DR1:[  0,1   ]   19K     0    Non-Res, Addr:1130
DAPRES < 42>   DR1:[  0,1   ]   10K     0    Non-Res, Addr:1120
```

The resident libraries report provides information about resident libraries. Items reported are:

Item	Description
Name	Name of the resident library.
Prot	Protection code of the resident library.
Acct	Account number of the resident library.
Size	Size of the resident library.
Users	Number of users accessing the resident library.
Comments	Information about the status and the starting memory location for the resident library. See Table 2-30 for the list of abbreviations used in this column.

```
Message Receivers:
Rcvrid    Job    Rib   Obj    Msgs/Max   Links/InMax/OutMax   Access
ERRLOG    1       0     1       0/40          0/0/0             Prv
OPSER     2       0     0       0/30          0/0/255           Lcl
QUEMAN    3       0     6       0/20          0/0/255           Lcl
LP0SPL    4       0     0       0/5           0/0/255           Prv
BA0SPL    5       0     0       0/5           0/0/255           Prv
```

BA1SPL	6	0	0	0/5	0/0/255	Prv
BA2SPL	7	0	0	0/5	0/0/255	Prv
MAIL	8	0	27	0/5	0/1/255	Net
QM$CMD	9	1	3	0/20	0/0/255	Prv
QM$SRV	9	2	4	0/10	0/0/255	Prv
QM$URP	9	3	5	0/10	0/0/255	Lcl
EVTLOG	10	0	0	0/32	0/0/255	Evt, Prv
EVTLSN	10	1	26	0/16	0/8/255	Net, Prv
NWTT11	11	0	24	0/5	1/0/255	Net
NWPK17	17	0	23	0/16	2/8/255	Net
NWTT18	18	0	24	0/5	1/0/255	Net
PR$26A	26	17	65	0/5	0/0/255	Prv
BA$26A	26	41	66	0/5	0/0/255	Prv
NSP	NSP	0	0	0/0	0/32/0	None

The message receivers report provides information about message receivers. Items reported are:

Item	Description
Rcvrid	Receiving job name.
Job	Receiving job number.
Rib	Number of Receiver ID blocks (RIBs).
Obj	Object type code.
Msgs/Max	Number of messages queued for the job and the declared maximum number of queued messages allowed.
Links/InMax/OutMax	Number of logical links in use and the maximum number of links allowed.
Access	Indicates whether local and network senders are allowed and whether the job can handle only one link. See Table 2-30 for the list of abbreviations used in this column.

Table 2-30 lists the abbreviations for a full SYSTAT report.

Table 2-30: Abbreviations for Full SYSTAT Report

Abbreviation	Meaning
Job Status (State)	
??	Job's state cannot be determined.
,	Job is not logged in to the system.
^C	Job is in CTRL/C state, awaiting input from the keyboard monitor.
BF	Job is waiting for buffers (no space is available for I/O buffers).
CR	Job is waiting for card reader input.
DB,DF,DK,DL, DM,DP,DR,DS, or DU	Job is waiting to perform disk I/O.
DET	Job is detached from all terminals.
DT	Job is waiting for DECtape I/O.
DX	Job is waiting for flexible diskette I/O.
FP	Job is waiting for file processing action by the system (opening or closing a file, file search).
HB	Job is detached and waiting to perform I/O to or from a terminal.
KB	Job is waiting for input from a terminal.
LP	Job is waiting to perform line printer output.
MM,MS, or MT	Job is waiting for magnetic tape 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.
RJ	Job is waiting for RJ2780 I/O.

Table 2-30: Abbreviations for Full SYSTAT Report (Cont.)

Abbreviation	Meaning
Job Status (State)	
RN	Job is running or waiting to run.
RS	Job is waiting for residency.
SL	Job is sleeping (SLEEP statement).
SR	Job is sleeping and is a message receiver.
TT	Job is waiting to perform output to a terminal.
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 Devices (Why)	
ANSI	Magnetic tape is assigned with ANSI standard labeling format.
AS	Device is explicitly assigned to a job.
DOS	Magnetic tape is assigned with DOS labeling format.
OPEN	Device is open on a channel.

Table 2-30: Abbreviations for Full SYSTAT Report (Cont.)

Abbreviation	Meaning
Disk Structure (Comments)	
Dirty	Disk needs cleaning.
DLW	Date of last write (modify), rather than date of last access, is stored in file accounting entries.
Job n	Private disk is assigned to job n.
INI	Disk is open in mode 512 or is being initialized by DSKINT.
Lck	Disk is in a locked state.
NFF	New files on this disk are put at beginning of directory.
NFS	Disk is open as non-file-structured device.
PRI	Cartridge or pack is private.
PUB	Cartridge or pack is public.
R-O	Disk unit is read-only (write-locked).
Run-time Systems and Resident Libraries (Comments)	
Addr:xxx	Denotes the starting address of the 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.
DF KBM	Run-time system is the default keyboard monitor.
EMT:yyy	Denotes the EMT code for special EMT prefix.
KBM	Run-time system can serve as keyboard monitor.
Loading	Run-time system or library is being loaded into memory.
Monitor	Denotes monitor RSX emulation.

Table 2-30: Abbreviations for Full SYSTAT Report (Cont.)

Abbreviation	Meaning
Run-time Systems and Resident Libraries (Comments)	
NER	Errors occurring within run-time system or library will not be sent to system error log.
Non-Res	Run-time system or library is nonresident.
Perm	Run-time system or library will stay in memory when not being used.
Primary	Denotes the primary run-time system.
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.
R/W	Run-time system or library allows read/write access.
Temp	Run-time system or library will be removed from memory when not being used.
1US	Run-time system or library can serve only one user.
Message Receivers (Access)	
Evt	Indicates the DECnet event logger.
Lcl	Local senders are allowed for this receiver ID.
Net	Network senders are allowed for this receiver ID.
One	User has indicated that job should handle only one network link.
Prv	Local senders must have SEND privilege.

Contents of the Open File Reports

This section contains a description of the open file reports. Use the /O switch to obtain the report about open files only. Use the /W switch to also include information about the jobs accessing the open files. The following example includes both the open files and the jobs accessing them:

```
$RUN $SYSTAT
SYSTAT V9.0 RSTS V9.0
Output Status to? /W

Open files and jobs accessing them:

DM0: -- None

DR0: -- None

DR1: -- File                  Op/RR     Size    Clu    Status
DR1:[  1,204]TEMP27.TMP<60>    1/0        6      4    Tent
      Job 27  Block 3         Rd, Wr, Tent

DR1:[  1,2  ]BATC07.WRK<60>    1/0        9      4
      Job 7   Block 1         Rd, Wr

DR1:[ 50,50 ]MAIL  .TSK<232>  0/1      620    128
      Job 24  Block 76        Rd, RR
           .
           .
           .
```

Items reported are:

Item	Description
File	Device, account, file name, file type, and protection code of each file opened on the system or specified device.
Op/RR	Number of jobs that have the file opened (or opened in read regardless mode).
Size	File size.
Clu	Cluster size of the file.
Status	File status. See Table 2-31 for the list of abbreviations used in this column.

Item	Description
Job	Job(s) accessing the file.
Block	Block being accessed.
Access	File access allowed for users.

Table 2-31 lists the abbreviations used in the Access column.

Table 2-31: Abbreviations for Open File Reports

Abbreviation	Meaning
Open Files (Status)	
Ctg	File is contiguous.
MDL	File is marked for deletion.
NoK	File cannot be renamed or deleted.
None	No files are open on the disk.
Pla	File is placed.
System files only	Only swap files or run-time systems are open on the disk.
Tent	Tentative file.
UFD	File is a UFD-type entry.
Upd	File is open in UPDATE mode.
Jobs Accessing Open Files (Access)	
Ca	File is open for user data caching.
Rd	User has read access to the file.
RR	File is open read regardless.
SpUp	File is open in special UPDATE mode.
Sq	File is open for sequential user data caching.

Table 2-31: Abbreviations for Open File Reports (Cont.)

Abbreviation	Meaning
Jobs Accessing Open Files (Access)	
Tent	Tentative file.
Up	File is open for UPDATE.
Wr	User has write access to the file.

Contents of the Memory Allocation Report

This section contains a description of the memory allocation report. The memory allocation report lists the starting and ending addresses for currently installed run-time systems and resident libraries. You can use this information to determine available addresses when installing new run-time systems and resident libraries. Use the /C switch to obtain the memory allocation report.

The memory allocation report lists memory information about the following items:

- Monitor (the resident portion).
- XBUF (extended buffer pool, if present).
- Locked out memory (if any).
- Nonexistent memory (if any exists before the end of physical memory).
- Resident libraries.
- Run-time systems that were added at specific addresses or were loaded with the STAY attribute.

Note

The memory allocation report lists only those items assigned to specific addresses. The report does not reflect the dynamic state of memory.

The following example shows the memory allocation report:

```
$RUN $SYSTAT
SYSTAT V9.0 RSTS V9.0
Output Status to? /C

Memory allocation table:
 Start    End  Length  Permanent    Temporary
   0K -    70K (   71K) MONITOR
  71K -    71K (    1K) RSX    RTS
 724K -   850K ( 127K) ** XBUF **
              .
              .
              .
1120K - 1129K (   10K)   (User)   DAPRES LIB
1130K - 1148K (   19K)   (User)   B2MRES LIB
              .
              .
              .
1264k - 1390k ( 127k) Locked out
1664K -   *** END ***
```

Items reported are:

Item	Description
Start	Starting address of that portion of memory.
End	Ending address of that portion of memory.
Length	Length of that memory segment.
Permanent	Those items allocated by INIT.SYS, and run-time systems and resident libraries that are permanently loaded. The program labels memory that is not permanently allocated in one of these as "User."
Temporary	Run-time systems and resident libraries that were added at specific addresses but not permanently loaded.

SYSTAT as a CCL Command

You can run SYSTAT with a CCL command if your system manager has defined it on your system. When you use the CCL command, you can include a file specification on the command line. Assuming that SYS is the CCL command, the following examples show how to run SYSTAT using this CCL command:

```
$ SYS REP
```

SYSTAT creates file REP and writes a full status report to the file. The file resides in your current account on the public structure.

```
$ SYS /B
```

SYSTAT displays a busy devices report on your terminal.

To write a particular report to a file, type the file specification followed by the appropriate switch. For example:

```
$ SYS REP/B
```

In this example, SYSTAT writes a busy devices report to the file REP. The default file type is .RPT.

Communicating with Other Terminals: The TALK Program

The TALK program allows you to communicate with other terminals on your system. You can send a message to a user's terminal or to the system console.

Privilege Required

Usually your system manager assigns a protection code of <232>, which permits all users to run TALK. However, your system manager can restrict the use of TALK by assigning a protection code of <124>. Check with your system manager to find out if you have access to TALK.

Running the TALK Program

TALK enables you to send line-by-line messages to another terminal. If the terminal user receiving your message wants to communicate with you, that user must also run TALK. Both the sending and receiving terminals must be on line, but a user does not need to be logged in to receive a message.

To run TALK, type:

```
$ RUN AUXLIB$:TALK
TALK V9.0 RSTS V9.0
```

After TALK prints its header, the program prompts you for the keyboard number of the terminal to which you want to send a message:

```
To which keyboard?
```

Press RETURN after you enter the keyboard number, 12, for example. TALK accepts a response such as KB12 but the KB prefix is optional. To make sure that you do not disrupt an elaborate printout on a user's terminal, it is good policy not to send a message to a terminal unless you know the type of activity the terminal user is performing. An unscheduled message can sometimes destroy hours of work. Use SYSTAT or DISPLY to determine the status of another user's job. If the job state is ^C, then the job is at a keyboard monitor prompt and most likely your message will not interrupt the job.

If you type the number of an offline terminal:

- o TALK does not issue an error message

- o TALK sends Messages normally, but they are never received

- o TALK does not terminate when you press RETURN

If you type the number of a nonexistent terminal:

- o TALK does not issue an error message
- o TALK terminates when you press RETURN

Once you enter a keyboard number and press RETURN, TALK displays the following instructions:

```
Enter message below (CTRL/Z to end):

Message:
```

To send a message, type a line and press RETURN. As long as you terminate a line by pressing RETURN, TALK displays the Message: prompt on the next line.

To terminate the message and TALK, enter CTRL/Z at the Message: prompt. If you enter CTRL/Z at the end of a message line, TALK terminates and does not send that message line. For example:

```
Enter message below (CTRL/Z to end):

Message:Sending a test message
Message:<CTRL/Z>
$
```

On the receiving terminal, TALK displays the message line and positions the cursor at the beginning of the next line. To return to your keyboard monitor prompt, press RETURN. For example:

```
$ ** KB51 ** Sending a test message
(cursor here) RET>
$
```

A Sample TALK Terminal Session

This sample session represents a typical TALK dialogue. Maryann originates the TALK session from keyboard number 9 and communicates with Robert at terminal number 12. Notice that blank lines placed between lines of text, make the received message more readable. To send a blank line, press the RETURN key at the message: prompt. After determining that the terminal with which you want to communicate is not performing a critical operation, proceed as follows:

```
$ RUN AUXLIB$:TALK
TALK V9.0 RSTS V9.0
```

```
To which keyboard? 12
Enter message below (CTRL/Z to end):

Message: Robert, <RET>
Message: <RET>
Message: Would you please take the RP06 disk, marked SYSGEN, from <RET>
Message: <RET>
Message: my cabinet and place it on line? I will be needing it <RET>
Message: <RET>
Message: later on. Thanks! Maryann <RET>
Message: <CTRL/Z>
$
```

Robert's terminal automatically identifies the sending device by enclosing the terminal number within * characters at the beginning of each message line.

```
$ ** KB9 ** Robert,
 ** KB9 **
 ** KB9 ** Would you please take the RP06 disk, marked SYSGEN, from
 ** KB9 **
 ** KB9 ** my cabinet and place it on line? I will be needing it
 ** KB9 **
 ** KB9 ** later. Thanks! Maryann
```

Robert replies as follows:

```
$ RUN AUXLIB$:TALK
TALK V9.0 RSTS V9.0

To which keyboard? 9
Enter message below (CTRL/Z to end):

Message: Maryann, <RET>
Message: <RET>
Message: Found it! And, you probably thought it would be easy. <RET>
Message: <RET>
Message: Such a cabinet! (Just kidding.) Your RP06, labeled SYSGEN, <RET>
Message: <RET>
Message: is now on line. <RET>
Message: <RET>
Message: Robert
Message: <CTRL/Z>
$
```

The message is displayed on Maryann's terminal as follows:

```
$ ** KB12 ** Maryann,
** KB12 **
** KB12 ** Found it! And, you probably thought it be would be easy.
```

```
** KB12 **
** KB12 ** Such a cabinet! (Just kidding.) Your RP06, labeled SYSGEN,
** KB12 **
** KB12 ** is now on line.
** KB12 **
** KB12 ** Robert
$
```

Appendix A

The Batch Processor Control Language

The OPSER-Based Batch Processor can execute a stream of commands from a control file that you create. This lets you submit jobs for processing without requiring terminal dialogue. Because the OPSER-Based Batch Processor runs from a pseudo keyboard, your terminal is available for other use during processing. Batch processing is particularly useful in data processing operations that do not require interaction. For simplicity, the remainder of this appendix refers to the OPSER-Based Batch Processor as the Batch Processor.

You can submit batch input from control files that you create using PIP or an editor on a random access device. For purposes of this description, input is dealt with as though it were on cards, where each card represents one record. Such input consists of batch processor control language statements and is called a batch stream. It is possible to execute multiple streams simultaneously by submitting multiple control files to the Batch Processor. The system manager controls whether you can run more than one batch stream. This section describes the Batch Processor language control statements and how to include them in Batch Processor control files.

Privilege Required

To submit jobs to the Batch Processor you need the following:

- o Read access to the files you are submitting.
- o Write access if you use the /DELETE switch.

Control Language Statements

Batch Processor control statements consist of a command field, one or more specification fields, and a comment field, in the following format:

```
$[command-field] [specification-field(s)] [!comment]
```

You must separate fields by one or more spaces and/or tabs.

Control statements always require a command field, which can contain switches to control or limit the command. When appropriate, one or more specification fields follow the command field. The exclamation mark character is a comment prefix signifying that the information between the ! character and the line terminating character is a comment. The Batch Processor takes no action on comment information. Note that the comment character (!) only applies to command lines. If you include a comment in a data statement, the Batch Processor interprets the comment as data.

Command Field

The command field directs the Batch Processor to perform the operation you specify in the field.

The command field has the format:

```
$command[switch(es)]
```

The elements of the command field are:

Element	Meaning
$	The dollar sign ($) character is the control statement recognition character. It must always occupy the first character position.
command	This specifies the function you want the Batch Processor to perform. A command begins in the second character position immediately after the $ character.
switch(es)	This controls or limits the command. A switch, denoted by a slash (/)character, follows the command. You cannot include a blank between the command name and the switch.

You can shorten command names and switch names to their first three characters. For example, you can use either $BAS or $BASIC.

Because you must use blanks to delimit fields within a control statement, you cannot include blanks within the command field.

Specification Fields

The command field supplies the objects on which the Batch Processor performs the operation you specify in the command field. Depending on the command, a specification field can consist of a device specification, a file specification, or an arbitrary ASCII string, any of which can be followed by switches.

The specification field has the format:

```
[filespec][/sw] [filespec][/sw]...
```

Use blanks to separate multiple specification fields, and terminate the line with any line terminating character, for example RETURN, or an exclamation mark character if you include a comment.

The / character signals the start of a switch. For example:

```
XYZ.BAS/SOURCE
```

This switch indicates that file XYZ.BAS is a source file.

If you optionally omit all or part of the file specification, the Batch Processor assigns default values. See the section "File Specification Syntax" for a description of these defaults.

Comments

Use an exclamation mark (!) character to denote the start of a comment in a control statement. The Batch Processor interprets any text included between the ! character and the end-of-line terminator as a nonexecutable comment.

You can use empty comment lines that include no text to force line spacing on the job log and thereby make the log more readable. An empty comment consists of $! followed immediately by pressing RETURN.

Control Statement Syntax

Syntax rules for control statements follow:

- Except for empty comment lines, you must include a command name in all control statements. If you omit the command name, the Batch Processor does not execute the control statement. If you include a command name that is not recognized by the Batch Processor, it displays an error message.

- Subject to restrictions on individual commands, you can substitute an * wildcard character in the file name or the file type field of a file specification. See the Section "File Specification Syntax" for the description of file specifications.

  An * can refer only to files that already exist. If you include an * in the specification of a file that does not exist, the file specification is invalid.

- If you include file names that are longer than six characters, the Batch Processor uses only the leftmost six characters. If you include file types that are longer than three characters, the Batch Processor uses only the leftmost three characters.

- You can use switches in the command and specification fields of a control statement. Switches appearing in the command field are command qualifiers, and their function applies to the entire command. Switches appearing in specification fields apply only to the field in which they appear. If a switch in the specification field conflicts with a command field switch, the Batch Processor returns an error message. An unrecognizable switch invalidates the control statement in which it appears.

- A physical command line can have a maximum of 120 characters.

- The hyphen (-) character can indicate the continuation of a command. If a hyphen is the last character on a line, the Batch Processor interprets the next line as a continuation of the previous line. The continuation line, however, must begin with a $ character followed by a blank. If you use the hyphen, you must place it before any comment.

- You can use quotation marks (") in control statements to denote a literal text string or to override special character interpretation by the Batch Processor. For example, in command environments other than DCL, the exclamation mark (!) designates auxiliary library account [1,3] or an installation-defined account. In control statements however, the ! character denotes a comment. To prevent the Batch Processor from misinterpreting the ! character given as an account designator, include quotation marks. For example:

  ```
  $RUN "!UPDAT"
  ```

  As a result, the Batch Processor executes program UPDAT from the auxiliary library account. Without the quotation marks, the Batch Processor interprets the characters following the ! character as a comment. See Table A-1 for a summary of special characters.

The following example shows the syntax of control statements:

```
$JOB/NAME=SMYTHE !FIRST JOB
$!
$!COMPILATION OF NEW SOURCE FILES
$!
$MESSAGE STARTING COMPILATIONS
$BASIC XYZ/SOURCE XYZ.LIS/LIST XYZ/EXECUTE
$BASIC ABC/SOURCE ABC.LIS/LIST ABC/EXECUTE
$!
$MESSAGE STARTING LISTING OUTPUT
$!
$PRINT *.LIS !ALL LIST FILES
$!
$EOJ
```

Table A-1 lists and describes the control statement special characters.

Table A-1: Control Statement Special Characters

Character	Meaning
Space	Separates fields in a control statement. Otherwise ignored unless embedded in a string delimited by quotation marks.
Horizontal tab	Separates fields in a control statement. (Equivalent to one space (blank) character; otherwise ignored.)

Table A-1: Control Statement Special Characters (Cont.)

Character	Meaning
Hyphen(-)	As the last nonblank character in a control statement, indicates that a continuation line follows. If the statement contains a comment, the hyphen must be the last nonblank character before the exclamation mark.
Exclamation mark (!)	Indicates a comment unless embedded in a string delimited by quotation marks.
Dollar sign ($)	Used as the first character in the first position of a control statement; causes control statement recognition.
Slash (/)	Denotes a switch (separates specification field from switch name).
Asterisk (*)	Indicates a wildcard in place of the file name or file type.
Colon (:) Equal (=)	Separates the switch name from the argument.
Quotation marks, (")	Used to open and close a string to preserve embedded spaces or to pass a special character (for example, !) without interpretation by the Batch Processor.
plus (+)	Indicates file concatenation in $COPY commands.
comma (,)	Separates file, device, and/or account specifications within a specification field, which allows multiple elements.

File Specification Syntax

A file name is a string of one- to six-alphanumeric characters. You can include an asterisk (*) wildcard in place of a file name to denote all files of the specified type in the specified account. If necessary, the Batch Processor generates a default file name related to the time of day.

A file type consists of a period immediately followed by a string containing three alphanumeric characters.

The file type reflects the nature of the file. For example, a BASIC-PLUS source file has a .BAS file type. An asterisk (*) wildcard in place of a file type denotes all file types, including null types.

Some standard file types are:

File Type	Meaning
.CTL	Batch control file
.DAT	Data file
.DIR	Directory file
.BAS	BASIC-PLUS source file
.LIS	List file
.BAC	BASIC-PLUS compiled output file
.OBJ	FORTRAN compiled output file
.SRT	PDP-11 SORT-11 input, output, or listing file
.MAP	Task Builder map file
.TMP	Temporary file
.B2S	BASIC-PLUS-2 source file
.TSK	Task built executable file (from TKB)
.FOR	FORTRAN source file
.SAV	RT11 run-time system executable file (from LINK)

These file types are the defaults when you do not specify a file type. The Batch Processor determines the default file type based on the associated command and the file type expected.

Table A-2 summarizes the default file types associated with particular commands.

Table A-2: Command-Related File Specification Defaults

Command	Default File Type	Meaning
$BASIC	.BAS .B2S	Input source file default (BASIC-PLUS) (BASIC-PLUS-2)
	.BAC .TSK	Output executable file default (BASIC-PLUS) (BASIC-PLUS-2)
	.LST	Listing file default
$CREATE	.DAT	The file generated as output by CREATE has a file type of .DAT
$DIRECTORY	.DIR	The directory listing file has a file type of .DIR
$FORTRAN	.FOR	Input source file default
	.OBJ	Output object file default
	.LST	Listing file default
	.SAV	Executable linked file
$JOB	.CTL	Batch control file default; assumed when the batch job is on a file-structured device
	.LOG	Batch output log file default
$PRINT	.LIS	Default file type for files that you type
$RUN	--	Checks internal default runnable file type list for installed run-time system
$SORT	.SRT	Input or output file default

When you omit file name and file type elements, the Batch Processor assigns default file names and file types.

Table A-3 describes the file specification defaults.

Table A-3: File Specification Defaults

Condition	Default	Example
File name, but no file type	Default assigned as appropriate to the current operation. For example, with the $BASIC command, the default is .BAS.	ABC=ABC.typ
File name, followed by period, but no file type	Default file type is null. No file type is assigned.	ABC.=ABC
File type, but no file name	Default file name is related to time of day.	.LIS=B2347P.LIS (created at 01:23:47 PM)
No file specification	Default file name (related to time of day) with default type as appropriate to current operation.	Null=B2347P.typ

Switch Syntax

Switches consist of a slash (/) character followed immediately by a name. If the switch takes an argument, use a colon (:) or equal (=) symbol to separate the argument from the switch name. If the switch takes an argument and subarguments, use a colon to separate the argument and each subargument. For example:

```
/NAME=JOB3
/FOR:F:512
```

You can negate switch values by prefixing the switch name with NO. For example:

```
/NOOBJ
```

This switch tells the Batch Processor not to produce an object file.

The NO prefix is not considered part of the switch name. Thus, a negated switch must contain at least five characters. For example, the switch /NOOBJ or /NOOBJECT is valid; /NOO is invalid.

Commands

Table A-4 lists the Batch Processor command set and their meanings. The sections that follow explain each command in detail.

Table A-4: Batch Processor Command Set

Command	Function
$JOB	Begins a job
$EOJ	Ends a job
$BASIC	Executes the BASIC-PLUS interpreter or BASIC-PLUS-2 compiler
$DELETE	Deletes files
$COPY	Copies files
$PRINT	Queues a file for the default line printer
$DIRECTORY	Lists an account directory
$CREATE	Creates a file from data in the input stream
$RUN	Executes a program
$DATA	Begins data images
$EOD	Ends data images
$MESSAGE	Logs a message on the operator services console
$MOUNT	Assigns a device
$DISMOUNT	Deassigns a device
$SORT	Executes the PDP-11 SORT-11 program
$FORTRAN	Executes the FORTRAN compiler

Table A-4: Batch Processor Command Set (Cont.)

Command	Function
CCL system command or $ system command	Executes a system utility function

Note

The $BASIC, $FORTRAN, and $SORT commands are subsets of BASIC, FORTRAN, and SORT capabilities. To use the full capabilities of BASIC, FORTRAN, or SORT, use the $RUN command.

The $JOB Command

This command marks the beginning of a job. The $JOB control statement has the format:

```
$JOB[/sw] [proj,prog]
```

You can use the following switches with the $JOB command:

Switch	Meaning
/NAME=jobname	Assigns a name to the job. Jobnames can be up to six characters long. This name overrides the control file name as the identifier of the job.
/NONAME	Indicates that no jobname is defined. The Batch Processor assigns a default jobname. The default jobname is the control file name. This name appears in all messages to the system operator.
/LIMIT=nnn	Assigns an elapsed time limit to the job. The Batch Processor interprets the value of nnn, a decimal number, as minutes. That the elapsed time required to execute a job is heavily dependent on overall system load.

Switch	Meaning
/NOLIMIT	Gives the job an unlimited amount of elapsed time to complete. If you do not specify either /LIMIT:nn or /NOLIMIT, the Batch Processor allows ten minutes for the job to complete. After ten minutes, the Batch Processor terminates the job.
/CPU:nnn	Assigns a CPU time limit to the job. The Batch Processor interprets the value of nnn, a decimal number, as seconds. If you specify /CPU but not /LIMIT, the Batch Processor does not enforce any elapsed time limit, but does enforce the CPU time limit. If you specify both switches, the Batch Processor enforces both limits. If you omit a CPU time limit specification, the Batch Processor allows unlimited CPU time.
/NOCPU	Gives the job an unlimited amount of CPU time to complete. This is the default.
/QUE	Queues the batch log file to LP0: (default) or the device specified at system start-up.
/NOQUE	Inhibits printing the batch log file.
/PRIORITY:n	Sets the RSTS/E job priority to n (or the next lowest multiple of 8) for the batch stream. The value of n can be between -120 and +127. Unless otherwise specified by the /PRIORITY:n switch, all jobs run at -8 priority.
/DCL	Disables all current commands that conflict with DCL commands and allows DCL to process them.
/CCL	Allows the use of the system's interactive Concise Command Language (CCL). When you specify this switch, you can include any CCL command that does not conflict with control statement commands after the $ character. The Batch Processor makes sure that the job is waiting for input for the keyboard monitor (the ^C state reported by SYSTAT) before executing the command.

Switch	Meaning
/ERROR:n	Specifies the level of error that the Batch Processor allows without terminating the job. Specify the error level you want with n, which can be: FAT[AL] BATCH tolerates all errors WAR[NING] BATCH tolerates warning errors, but a fatal error terminates the job NON[E] Any error terminates the job If the Batch Processor terminates a job because of an error, termination occurs at the next request for input. The Batch Processor writes a message explaining the termination in the log file. The system manager determines the default error level for the batch stream at start-up time.

Note

See the *RSTS/E System User's Guide* for the severity standards in error messages. Some programs (for example the FORTRAN IV compiler) which operate under a run-time system other than BASIC-PLUS do not use the standard severity characters in error messages. Therefore, the Batch Processor does not detect compilation errors. User programs coded to run under batch control must use standard severity characters.

You can include the following within the $JOB command:

[n,m] You can execute a job from an account other than the account under which you queued the job. The specification field indicates the account you want. Use spaces to separate specification fields from other parts of the control statement. For example:

```
$JOB/NAME=TEST/NOLIMIT/CCL [1,2]
```

The following $JOB command error conditions are possible:

- o Unrecognized switch
- o Illegal switch value
- o Multiple conflicting specifications (switches)
- o Different account specified
- o Higher priority desired

The $EOJ Command

This command marks the end of a job. The Batch Processor deassigns all devices previously assigned by the job when $EOJ executes. The Batch Processor performs a logical deassignment, then prints an message indicating that the operator should dismount the physical device.

The $EOJ command is implied when the Batch Processor finds a physical end-of-file (EOF) condition or another $JOB control statement while processing a control file.

No switches or file specifications are legal in the $EOJ command.

The $BASIC Command

The $BASIC command calls a BASIC compiler to compile a source program. The $BASIC command has the format:

```
$BAS[IC][/sw] specification-field[/sw] [spec-field[/sw]]...
```

When $BASIC appears in the control file, the Batch Processor assumes that BASIC-PLUS is the default keyboard monitor. If this is not the case and you want to use the BASIC-PLUS compiler, use the DCL command SET JOB/KEYBOARD_MONITOR=BASIC to switch to the BASIC-PLUS keyboard monitor before issuing the $BASIC command in the control file (see the *RSTS/E Quick Reference Guide*).

The $BASIC Command Field

This section describes the command field part of the $BASIC command. It has the format:

```
$BAS[IC][/sw] [specification-field[/sw]] [spec-field[/sw]]...
```

The switches in the $BASIC command field are:

Switch	Meaning
/BP1	Tells the Batch Processor to use the BASIC-PLUS compiler. If you do not specify either /BP1 or /BP2, the Batch Processor assumes /BP1. The default file type is .BAS.
/BP2	Tells the Batch Processor to use the BASIC-PLUS-2 compiler. If you do not specify either /BP2 or /BP1, the the Batch Processor assumes /BP1. The default file type is .B2S.
/RUN	Tells the Batch Processor to execute only a previously compiled/task built program. If you use this switch, the entire command line must include only one file specification. The only other switch you can use is /EXECUTE.
/NORUN	Tells the Batch Processor to perform the compile/task-build procedure, but not to execute the final file.
/LIST	Tells the Batch Processor to produce the listing file filename.LST, where the file name is the source file name. The default is /NOLIST.
/NOLIST	Tells the Batch Processor to not produce a listing file. This is the default.
/EXECUTE	Tells the Batch Processor to create an executable file, whose file name is the source file name. To select a default file type, the Batch Processor uses these rules: o The file type is .BAC for BASIC-PLUS. o The file type is .TSK for BASIC-PLUS-2.
/NOEXECUTE	Tells the Batch Processor that if an executable file is needed, to create a temporary file and delete the file after executing the $BASIC command.

The following switches imply /BP2 causing BATCH to run BASIC-PLUS-2:

Switch	Meaning
/OBJECT	Create the object file filename.OBJ, where the file name is that of the source file. This switch causes a task build operation. Thus it is legal only with BASIC-PLUS-2.
/NOOBJECT	Create the object file filename.TMP, where the file name is that of the source file. Delete this .TMP (temporary) file upon completing the command.
/MAP	Create the task builder map file filename.MAP, where the file name is that of the source file. This switch causes a task build operation. Thus, it is legal only in a BASIC-PLUS-2 run. The default is /NOMAP.
/NOMAP	Do not create a map file. This is the default.

The $BASIC Specification Field

This section discusses the specification field part of the $BASIC control statement. It has the format:

```
$BAS[IC][/sw] specification-field[/sw] [...]
```

You can include more than one specification field in the command. The legal switches can vary depending on the position of the field on the command line.

The $BASIC command directs BATCH to search for a source file. The command looks for a specification field containing the /SOURCE switch or a field that has no switch. $BASIC then assumes this field to be the input file for the command. For this reason, only one file specification can appear without switches. In addition, this specification can not contain * or ? wildcards.

If you do not specify a source file, you must follow the $BASIC command with a set of BASIC source statements, terminated by either $EOD (see the section "The $EOD Command") or some other recognizable batch control statement. For example:

```
$BAS LSTING/LIS
  BASIC
  Source
  Deck
$EOD
```

If you explicitly specify a source file, BATCH appends any source statements following this command. If any source statements that follow this command have line numbers equal to those in the source program, BATCH replaces those lines in the source program. You must provide source input either through either a file specification or source statements, or both.

Because you can use switches in both the command field and specification field, you must avoid specifying conflicting switches. Moreover, you cannot use each switch more than once in the entire $BASIC command line, nor can you use a switch and its negation anywhere in the same command line. For example, /LIST and /NOLIST cannot appear together in a command line.

You can use the following switches in the first specification field of the control statement:

Switch	Meaning
/SOURCE	Both switches mean that this is the BASIC-PLUS or
/BASIC	BASIC-PLUS-2 source file on which to operate.
/EXECUTE	This switch is legal only if /RUN appears in the command field, and means that this is an executable file.

When you specify a file without a switch, BATCH applies the default /SOURCE. If /NORUN appears in the command field, the absence of a switch here implies /BASIC. If /RUN appears in the command field, the absence of a switch here implies /EXECUTE. If either /RUN or /NORUN does not appear in the command field, the absence of a switch here implies /BASIC.

You can include additional specification fields in the command line to define other files that you may need in the operation. You can use any of the following switches:

Switch	Meaning
/LIST	Define the listing file as specified. If you specify the /LIST switch but do not include a listing file, the Batch Processor creates the default listing file. If you specify a file specification with the /LIST switch, the Batch Processor uses that specification for the file. To print the specified file as part of the Batch job, supply a $PRINT control statement (see the section "The $PRINT Command").
/OBJECT	Define the object file as specified. This switch implies /BP2 and causes BATCH to run BASIC-PLUS-2, with task build. Therefore, it is legal only with BASIC-PLUS-2.

Switch	Meaning
/MAP	Define the map file as specified. This switch implies /BP2 and causes BATCH to run BASIC-PLUS-2, with task build. Therefore, it is legal only with BASIC-PLUS-2.
/EXECUTE	Define the executable file as specified. If you do not specify an executable file with the /EXECUTE switch, BATCH creates a default executable file and deletes it after job completion. If you explicitly specify an executable file, BATCH preserves it after job execution.

Errors result from conflicting switch specifications such as placing both /BASIC and /SOURCE on different specification fields.

The following error conditions are possible:

- Unrecognized switch
- Multiple conflicting specifications (switches)
- File specification syntax error

The $DELETE Command

The $DELETE command deletes files you specify. It has the format:

```
$DEL[ETE] file-specification [ ...]
```

You must include the file name and file type. You cannot use the * wildcard character in either the file name or the file type field.

You cannot use switches with the $DELETE command.

The following error conditions are possible:

- No file specification
- Syntax error in file specification

The $COPY Command

The $COPY command copies files. The control statement has the format:

```
$COP[Y] file1[/sw] [file2...filen]
```

Follow the command field with a list of file specifications, separated by spaces. You cannot use the * wildcard character in the file specification. The following are the valid switches:

Switch	Meaning
/OUTPUT	To create new files
/INPUT	To copy existing files

If you do not specify a switch, the Batch Processor assumes the default, /INPUT. For example, in the following command, the Batch Processor assumes TERRY.LIS is the input file:

```
$COPY TER.LIS/OUTPUT TERRY.LIS
```

You can use the plus sign (+) character to direct COPY to concatenate files. Place the + character in the file specification field between the specifications of files you want to concatenate. The Batch Processor concatenates the files to create a single file. For example:

```
$COPY MERGE.LIS/OUTPUT DATA1.LIS+DATA2.LIS+DATA3.LIS
```

This command directs the Batch Processor to create a file named MERGE.LIS, copy the contents of DATA1.LIS to that file, and append the contents of DATA2.LIS and DATA3.LIS to that file. DATA3.LIS follows DATA2.LIS.

The following error conditions are possible:

- o No output specification
- o No input specification
- o Multiple conflicting specifications
- o Syntax error in file description

The $PRINT Command

The $PRINT command prints the contents of files on the system line printer by invoking spooling program SPOOL. File specifications accept all switches available in the Q command. You can use the * wildcard in file specifications. The $PRINT command has the format:

```
$PRI[NT] file1[/sw] [file2[/sw] ...  filen]
```

The specification field contains the file or files you want to print.

The following error conditions are possible:

- o No file specification
- o Syntax error in file specification

The $DIRECTORY Command

The $DIRECTORY command produces a directory listing of the file(s) in the account you specify, and has the following format:

```
$DIR[ECTORY] [specification field]
```

The specification field can contain full RSTS/E file specifications. If no file specification appears, the $DIR command lists the contents of the current account in the Batch log file. A file specification indicates the directory of a file or set of files and can contain an asterisk (*) wildcard in either the file name field or file type field. For example:

```
$DIR *.BAS
```

This command creates a directory listing of all files in the current account with the .BAS file type.

You can use the /DIRECTORY and /INPUT switches in the control statement. The following text shows their use.

To create a directory in a disk file rather than on the batch log device, specify a file and the /DIRECTORY switch. For example:

```
$DIR BAJOB.DIR/DIR
```

This control statement creates the directory listing in a file BAJOB.DIR on the system disk under the current account.

To create a directory in a disk file and to designate the files you want to list, specify both the /DIRECTORY and /INPUT switches in the related file specification. For example:

```
$DIR BA.DIR/DIR *.BAC/INPUT
```

The $DIR command in this example creates a directory listing of all compiled BASIC-PLUS (.BAC) files and stores the listing in the file BA.DIR on the system disk under the current account.

The following error conditions are possible:

- o Syntax error in file specification
- o Multiple conflicting specifications

The $CREATE Command

The $CREATE command creates the ASCII file you specify in the file field. The $CREATE command has the format:

```
$CRE[ATE] file
   .
<data>
   .
$EOD
```

The file contains the data following the $CREATE command in the input stream. If you do not terminate the data (which must follow the $CREATE command) with $EOD, the Batch Processor returns an error message. No other command can precede the data because the $CREATE function terminates on finding a $ in the first column of a line.

At batch execution time, the Batch Processor deletes any existing file having the same name, and replaces it with the file that $CREATE creates.

The following error conditions are possible:

- o Syntax error in file specification
- o No file name specified
- o Characters other than a ! character following file specification

The $RUN Command

The $RUN command executes system programs. The command line has the format:

```
$RUN [ppn] program [command line]
```

The $RUN command allows no switches. If you omit the program name, the Batch Processor uses the current program as the default. For example, to run PIP, type:

```
$RUN $PIP
```

Follow the PIP command with the appropriate command line. The PIP program reads the commands as data images in the input stream. The Batch Processor terminates PIP when it reads the next control statement.

The following error conditions are possible:

- o Syntax error in file specification
- o Noncomment characters (any character that is not a ! character) following file specification

The $DATA Command

The $DATA command lets you supply data to a program that one of the language commands ($BASIC, $FORTRAN) compiles and runs. $DATA makes sure that the Batch Processor runs the program, unless you include the /NORUN switch. The Batch Processor discards any remaining data that the program does not use.

Issue the $DATA command without specification fields or switches, in the format:

```
$DATA
  .
<data>
  .
$EOD
```

The $EOD Command

The $EOD command marks the end of data records you include in the input stream following commands such as $BASIC, $CREATE, $DATA, and $RUN. For example:

```
$DATA
  .
<data>
  .
$EOD
```

The $MESSAGE Command

The $MESSAGE command logs a message on the operator services console. It provides a way for the job to communicate with the operator. The $MESSAGE command has the format:

```
$MES[SAGE][/WAIT] message-string
```

The /WAIT switch directs the system to pause and wait for operator action. The system pauses until the operator gives the appropriate command. For example, the following command halts the program until the operator takes action:

```
$MESSAGE/WAIT MOUNT SCRATCH TAPE ON DT0:
```

The WAIT condition remains in effect until the operator responds to the message on the operator services console. See the *RSTS/E System Manager's Guide* for information on operator response procedures.

The $MOUNT Command

The $MOUNT command directs the Batch Processor to print a mount message on the operator service console, and to create a logical to physical device assignment. After the message appears, an automatic /WAIT occurs. The operator responds with the device and unit number in the standard format (for example, MT1:). The $MOUNT command has the format:

```
$MOU[NT] devn:[/sw] devm:[/sw]
```

You must specify both the logical device and the physical device. Use a colon to terminate each device specification. You can use the following switches for the physical device:

Switch	Format	Description
Physical	/PHY	Identifies the device specification to be the physical device (default).
Write	/WR	Tells the operator to write-enable the device (or volume).
No write	/NOWR	Tells the operator to write-protect the volume.
Visual identification	/VID=[name]	Identifies the volume for the operator with [name], which is a visual identification.
Density	/DEN:nnn	Specifies density for magnetic tape.
Parity	/PAR:ODD	Specifies odd parity for magnetic tape.
	/PAR:EVEN	Specifies even parity for magnetic tape.

You can use the following switch with the logical device:

Logical	/LOG	Identifies the device specification to be the logical device name. BATCH accepts logical device names up to six characters. For RSTS/E disks, this specification must correspond to the PACK ID.

The /VID switch on the physical device field specifies the volume identification. The value associated with /VID is the name physically attached to the volume. Include this switch to help the operator locate the volume. The volume identification cannot contain slashes (/), commas (,), or embedded blanks. If you must include blanks in the /VID string, enclose the name in quotation marks (").

The following example shows the use of the $MOUNT command:

```
$MOUNT MT:/PHY/VID="MY TAPE" TAPE:/LOG
```

In this example, the Batch Processor assigns the logical device name TAPE to a magnetic tape unit. The logical device name notifies the operator that the reel of tape you want physically mounted is labeled MY TAPE. The operator uses a PLEASE command to respond with the device and unit number on which the tape is mounted (see the *RSTS/E System Manager's Guide*). Thereafter, in the control file, reference to device TAPE: accesses the physical device on which the operator mounted the reel labeled MY TAPE. If the physical device is a removable disk pack or cartridge, the logical device name must be the pack identification. The Batch Processor logically mounts and unlocks private disks that the operator mounts as a result of $MOUNT.

The valid physical devices that you can request for mounting are:

Dev	Physical Device
CR:	Card Reader
DK:	RK11/RK05 Disk Cartridge
DP:	RP11/RP03/RP02 disk pack
DB:	RH11/RP04/RP05/RP06 disk pack
DM:	RK611/RK06/RK07 disk pack
DR:	RM02/RM03/RM05/RM80 disk pack
DS:	RH11 disk
DL:	RL01/RL02 disk
DX:	RX01/RX02 flexible diskette
DT:	TU56 DECtape
DU:	RA60/RA80/RA81/RC25/RD51 disks
LP:	Line printer
MT:	TE10/TU10/TS03 magnetic tape
MM:	TE16/TU16/TU77/TU45 magnetic tape
MS:	TS11/TSV05/TU80 magnetic tape
MU:	TU81 magnetic tape
PP:	Paper tape punch
PR:	Paper tape reader
SY:	System device
KB:	Terminal

The following error conditions are possible:

- o Syntax error in device specification fields
- o Invalid device name/unit
- o Invalid logical device name specifications
- o Unit number already assigned
- o Both physical and logical names have not been specified

The $DISMOUNT Command

The $DISMOUNT command cancels the logical to physical device assignment you created using the $MOUNT command. It also prints an operator message, requesting that the volume be physically dismounted. If you include a /WAIT switch in the command field, the job does not resume until the operator types a response, as with the $MESSAGE command. The $DISMOUNT command has the following format:

```
$DIS[MOUNT][/WAI[T]] logical device name:
```

Using the example from the $MOUNT description, the following command dismounts the device:

```
$DIS/WAI TAPE:
```

This command sends a message to the operator to dismount the magnetic tape that was mounted in previous example. The /WAI switch instructs the Batch Processor to pause until the operator responds.

The Batch Processor automatically dismounts all devices at end-of-job (EOJ).

The following error conditions are possible:

- o Syntax error in specification field
- o Illegal switches
- o Logical device not assigned

The $SORT Command

The $SORT command directs the Batch Processor to execute the SORT-11 program, which RSTS/E systems support. For additional information on the SORT-11 program, see the *PDP-11 SORT/MERGE User's Guide*. The $SORT control statement has the format:

```
$SORT [job switches] [output[/OUTPUT]] [input[/INPUT]] [spec/SPEC]
```

Job switches, which you can abbreviate to the first three letters, define the sort process. When you specify more than one switch, separate each switch with a slash (/). The following are valid job switches:

BATCH Switch	Meaning
/ALL[OCATION]:n	For output, specifies the initial space allocation for the file. The value n is in the range 0 to 65535 blocks. If you do not specify an allocation, the default depends on the sort process (see /PROCESS). If /PROCESS is Record or Tag, the default is the input file size. If /PROCESS is Index or Address Routing, the default is the number of records sorted.
/BLO[CKSIZE]:n	For magnetic tape input or output, specifies the tape block size. The default is a 512-byte block.
/BUC[KETSIZE]:n	For output, specifies the file's bucket size. The default is the bucket size of the input file.
/CON[TIGUOUS]	For output, specifies a contiguous file. The default is noncontiguous.
/DEV[ICE]:x	For input, specifies the device to be used for scratch files, where x is a one- to four-character device name.
/FIL[ES]:n	For input, specifies the number of scratch files; you can specify from three to ten scratch files.
/FOR[MAT]:x:n	For input, specifies the file's record format (x) and maximum record size (n). This switch is required. The value x can be: F (Fixed) S (Stream) V (Variable) U (Unknown) The default format is VARIABLE; you must specify a value n for the record size.
/IND[EXEDSEQUENTIAL]:x	For input, specifies indexed file organization. The value x specifies the number of keys.

BATCH Switch	Meaning
/KEY[S]:abm.n	For input, specifies the sorting key field. If you do not specify a file in the command line, you must include this switch. You can specify a maximum of ten keys, separated by colons. For example: /KEY:abm.n:abm.n:...abm.n
/PRO[CESS]:x	For input, specifies the sorting process. The value x can be: R (Record sort, the default) T (Tag sort) A (Address Routing) I (Index sort)
/REL[ATIVE]	For output, specifies relative file organization.
/SEQ[UENTIAL]	For output, specifies sequential file organization.
/SIZ[E]:n	For output, specifies the file's cluster size.

In the $SORT control statement, you can enter a maximum of three file specifications, separated by spaces: an input file, an output file, and a specification file. To distinguish these files, use the following switches, which you can abbreviate to the first three letters:

SORT Switch	Meaning
INP[UT]	The file to be sorted.
OUT[PUT]	The file to contain the sorted data.
SPE[CIFICATION]	The file that contains the control information for the sorting process.

The Batch Processor uses a file specification without a switch as the file you want to sort. If you use the /SPECIFICATION switch, you cannot use the /KEYS and /PROCESS switches in the command line. If you do not give a specification file in the control statement, you must include the /KEYS switch in the command field to control the sorting process. Always include the /FORMAT switch in the command field. If you omit a type from the file specification, the Batch Processor uses .SRT as the type.

The $FORTRAN Command

The $FORTRAN command calls the FORTRAN compiler. The $FORTRAN command has the format:

```
$FOR[TRAN][/sw] [specification field[/sw]] [spec fields[/sw]]
```

The following switches are valid in the command field:

Switch	Meaning
/RUN	Execute the previously compiled file. You can specify only an object file.
/NORUN	Compile the source program but do not execute the object file.
/OBJECT	Create the compiled file filename.OBJ, where the file name is that of the source file. The default is /NOOBJECT.
/NOOBJECT	Do not create an object file. This is the default.
/LIST	Produce the listing file filename.LST, where the file name is that of the source file. The default is /NOLIST.
/NOLIST	Do not produce a listing file. This is the default.
/MAP	Create the map file filename.MAP, where the file name is that of the source file. The default is /NOMAP.
/NOMAP	Do not create a map file. This is the default.

You can use one of the following switches in the first specification field:

/FORTRAN /SOURCE	Both switches have the same meaning. They indicate the source file on which to operate. If a file specification lacks a switch, the Batch Processor assumes it is a source file.

The optional specifications ending the format description define other files that you may need in the operation. You can use any of the following switches in the formats indicated. However, you can use each switch only once in the entire $FORTRAN command line. For example, you cannot use a switch and its negation anywhere in the same command line. For example, /LIST and /NOLIST cannot appear together in a command line.

File Specification	Meaning
file specification/LIST	Define the listing file as specified.
file specification/OBJECT	Define the object file as specified.
file specification/MAP	Define the map file as specified.

You can specify more than one library file in the command line by using the following switch:

```
file specification/LIBRARY
```

Each such specification is a library file that you want linked with the FORTRAN program. Library file specifications, delimited by spaces, can appear anywhere in the command line.

Batch Operating Procedures

This section describes how you request batch processing and how the Batch Processor generates output.

Requesting a Batch Job Run

To submit a job to the Batch Processor, run the QUE program and specify the control file or files as follows:

```
$RUN $QUE
QUE     V9 RSTS V9
#Q BA:BATJOB=FILE1,FILE2,FILE3.DAT
#
```

You normally queue a batch job to device BA:. The job and log files in this example are named BATJOB, and the files FILE1.CTL, FILE2.CTL, and FILE3.DAT are to be concatenated to form the control file. Log file BATJOB.LOG is to be printed after job completion, providing you did not specify the /NOQUE switch in the $JOB command or during start-up of the Batch Processor. Note that QUE schedules jobs on a strict priority basis, thus a high priority compute-bound job can slow processing.

Batch Job Processing

As the Batch Processor reads the control file, it checks for command sequence and syntax validity. If the Batch Processor detects an error, it writes an error message to the log file, does not run the job, and continues to check the syntax in the remainder of the control file.

A $MESSAGE/WAIT, a $MOUNT, or a $DISMOUNT/WAIT command causes the job to pause for an operator response. Until the operator takes action, the Batch Processor sends no further commands to the pseudo keyboard.

If the Batch Processor detects no errors, it processes the job and creates a log file showing the sequence of commands processed during the course of the job. If you direct program output to KB:, the output appears after the command that runs the program.

In the example that follows, the Batch Processor runs a job named JOB1. The control file contains the following sequence of commands:

```
$JOB/NAME=JOB1/LIMIT=4
$CREATE SUB1.BAS

      source statements

$EOD
$BASIC/BP2 LISTING/LIS MAIN/OBJ

      source statements

$DATA

       data

$PRINT SUB1.BAS
$EOJ
```

The following section describes each of the commands in the previous sample control file:

```
$JOB/NAME=JOB1/LIMIT=4
```

The Batch Processor assigns the name JOB1 to the job. This name, along with the date and time of execution, appears in the job log. The Batch Processor sets an elapsed time limit of four minutes. If the job is not finished in four minutes elapsed time, the Batch Processor terminates the job and writes an error message in the log file.

$CREATE SUB1.BAS

The Batch Processor creates the BASIC source file SUB1 from data records that must follow the $CREATE command.

$EOD

The $EOD command signals the end of SUB1.BAS.

$BASIC/BP2 LISTING/LIS MAIN/OBJ

The BASIC-PLUS-2 compiler compiles the source statements that follow this command. The compiler writes the source statements into the newly created LISTIN.B2S file, and places the object data in the MAIN.OBJ file. The temporary task built file contains the executable image.

$DATA

The data to be read during execution of MAIN.EXE follows this command.

$PRINT SUB1.BAS

The Batch Processor prints the source file created by $CREATE SUB1.BAS. This command also terminates data input to MAIN.EXE.

$EOJ

This command signals the end of job JOB1.

Error Procedures

When the Batch Processor detects a syntax error, it does not execute the job. Instead, the Batch Processor continues to scan the control file and writes an error log listing all scanned commands and data, along with the appropriate error messages. The log file always indicates all command lines scanned. If the Batch Processor finds an error on a command line, it writes the error message following the command, marked with question marks (????).

If no syntax errors occur, the Batch Processor writes the time of output of lines in the left margin of the log. All normal terminal interaction corresponding to the commands appear in the log.

Table A-5 lists the Batch Processor error messages and their meanings.

Table A-5: Batch Processor Error Messages

Message and Meaning
?BATCH BEING SHUT DOWN The Batch Processor is going off line and the job must be terminated.
?CANNOT INCREASE PRIORITY A /PRIORITY:n switch appeared in the $JOB command. The user was privileged but specified a value for n greater than 127. Or, the user was nonprivileged and specified a value greater than -8.
?CANNOT USE THAT ACCOUNT An account specification appeared in the $JOB command but the request did not come from a privileged user.
?CONTINUATION MISSING The hyphen (-) character was the last nonblank character in a control statement to continue the statement on the next line, but the following line did not begin with a dollar sign ($) and a blank.
?DEVICE NOT MOUNTED A $DISMOUNT command was present but the device indicated had not been mounted.
?DISK MOUNT FAILURE The volume to be mounted was not correct (pack IDs did not match) or the device was in use by another job.
?INVALID COMMAND An undefined command name followed the $ character in a statement but the /CCL command switch had not been specified in the $JOB command.
?INVALID SPECIFICATION FIELD The specification given in a control statement is in the wrong format.
?INVALID SWITCH The switch used in the command field or in the specification field is undefined, in the wrong format, or is privileged.
?NO BATCH JOBS POSSIBLE AT THIS TIME The Batch Processor requires a pseudo keyboard to execute a job but one is not available. Requeue your request.

Table A-5: Batch Processor Error Messages (Cont.)

Message and Meaning
?NO SUCH ACCOUNT The account specified in the $JOB command or in a specification field could not be found on the device.
?SEQUENCE NOT SUPPORTED YET The $SEQUENCE command is not available with this version of the Batch Processor.
?TIME LIMIT EXCEEDED Time specified in $JOB command is insufficient to execute the job. Specify a larger limit by using /LIMIT=nnn or /NOLIMIT switch.
?TOO MANY MOUNTED DEVICES The job has requested mounting of more devices than the maximum (12) allowed by the Batch Processor.
?UNABLE TO LOG IN BATCH JOB To execute a request, the Batch Processor logs a job into the system using the account under which the job was queued or the account specified in the $JOB command. For some reason, the login procedure failed. For example, this can occur, if logins had been disabled. The Batch Processor requeues the job for later execution.
?UNMATCHED PARENTHESES An opening left parenthesis appears in a specification field but an accompanying closing right parenthesis is not found.
?UNMATCHED QUOTATION MARKS Quotation marks (both single and double) must be paired in a control statement.

-N-

-O-

-P-

-Z-

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA and Puerto Rico call **800–258–1710**

In Canada call **800–267–6146**

In New Hampshire, Alaska or Hawaii call **603–884–6660**

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
A&SG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

*Any prepaid order from Puerto Rico must be placed with the Local Digital Subsidiary:
809–754–7575

RSTS/E
Utilities Reference Manual
AA–EZ11A–TC

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. ______________________________

____
____
____
____
____
____

Did you find errors in this manual? If so, specify the error and the page number. ______________

____
____
____
____
____
____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ______________________________

Name ______________________ Date ______________

Organization ______________________________

Street ______________________________

City ______________________ State ____________ Zip Code or Country ____________

Do Not Tear - Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: Office Systems Publications MK01-2/E02
RSTS/E Documentation
DIGITAL EQUIPMENT CORPORATION
CONTINENTAL BOULEVARD
MERRIMACK, N.H. 03054

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line