DEC-10-PPEO-D

PDP-10
PIP
(PERIPHERAL INTERCHANGE PROGRAM)
PROGRAMMER'S REFERENCE MANUAL

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Edition, October 1967
2nd Edition (Rev) May, 1968
3rd Edition (Rev) November, 1968
4th Edition (Rev) November, 1969
5th Edition (Rev) June, 1970
6th Edition (Rev) March, 1972

PREFACE

The functions provided the user by the DECsystem-10 Peripheral Interchange Program (PIP) and their use are described in this manual.

NOTE

Monitor commands are available which perform the common PIP functions of copying, renaming, protecting and deleting files.

It was assumed in the preparation of this manual that the reader is familiar with or has access to the DECsystem-10 Monitor Calls manual and the DECsystem-10 Monitor Commands manual. These manuals as well as the PIP manual are available in the DECsystem-10 Software Notebook and in the following handbooks:

a) DECsystem-10 User's Handbook (contains both PIP and the Monitor commands manuals).

b) DECsystem-10 Assembly Language Handbook (contains Monitor calls manual).

CONTENTS

CONTENTS

CONTENTS

SECTION 1

INTRODUCTION

1.1 INTRODUCTION

PIP (Peripheral Interchange Program) transfers files between standard I/O devices and can be used to perform simple editing and magnetic tape control operations during those transfer operations.

To call PIP into core (1) from the Monitor level, the user types the command

```
.R PIP     <CR>
```

When PIP is loaded and ready for input it prints the character * at the console. The user may then enter the command string needed to perform the desired operations followed by a carriage return input. On completion of the operation or operations requested in a command string, PIP again prints the character * to indicate that it is ready for the next command string input. To exit from PIP, the user types a Control C (↑C) command.

1.1.1 Controlling PIP Indirectly

PIP is normally controlled by commands entered via the console keyboard. PIP, however, is also capable of reading commands from a prepared file and executing these commands as if they had been just entered via the input console. PIP command files which are to be processed indirectly are identified by the addition of the symbol @ to their identifying file specification (see paragraph 2.1.2 for a description of file specifications). For example, the file specification FOO.CCL@ identifies the file FOO.CCL as an indirect command file. Any filename extension may be used in specifying an indirect command file, however, if none is given, the default extension .CCL is assumed.

An indirect PIP command file consists of one or more PIP commands structured as described in Section 2.

---

(1) The PIP program operates in 4K pure core plus a minimum of 1K of impure core in all DECsystem-10 systems.

Once PIP is in core, the user passes control of PIP to an indirect command file by entering the file's filename. For example, the input command sequence

```
.R PIP     <CR>
*FOO.CCL@     <CR>
```

loads PIP and initiates the execution of the indirect PIP command file FOO.CCL.

1.2 WRITING CONVENTIONS

The following symbols and abbreviations are used throughout this manual:

Symbol or Abbreviation	Meaning
dev:	Any logical or physical device name, the colon must be included when it is used as part of a PIP command.
file.ext	Any filename and filename extension.
[directory]	Identifies the directory of a specific file storage area within the system; it may also specify the location of specific file within the identified storage area. (See paragraph 2.4 for a detailed description of [directory].) When the input terminal used is either a Model 33 or 35 Teletype unit, the right and left brackets are input in the following manner: To Obtain a: — Type: a) left bracket — SHIFT K b) right bracket — SHIFT M
↑ch	A control character obtained by depressing the CTRL key and then the selected character key (e.g. ↑Z).
=	An equals character is used in the PIP command to separate the destination and source command sections. NOTE PIP will also accept the back arrow (SHIFT-O) entry. A SHIFT-O entry is echoed on the terminal printer as the symbol ←.
*	PIP's response to a command string to indicate that it is ready for the next input string.

Symbol or Abbreviation	Meaning
	The Monitor's response to a command string to indicate that it is ready for the next command.
<CR>	This symbol represents a carriage return, line-feed operation. It is initiated by the entry of a RETURN keyboard input. A RETURN input is normally used to terminate each PIP input command.
-----	Underscoring indicates computer typeout.
n	A number, either octal or decimal.
↑	This up-arrow symbol indicates the use of a CTRL key entry. The up-arrow is used with other character key inputs to produce special control entries such as ↑C which requests that control be returned to the Monitor. Up-arrows are also used to enclose identifiers which may be assigned to DECtapes using the facilities provided by PIP (see 3.2.1.2.).

SECTION 2

PIP COMMAND STRING AND ITS BASIC ELEMENTS

2.1 COMMAND STRING

PIP command strings may be of any length; both upper and lower case characters may be used. PIP commands are normally terminated and the requested operation is initiated by a RETURN keyboard entry (i.e., <CR>). However, an ALT MODE, line feed, vertical TAB or form feed keyboard entry can also be used as a command terminator.

2.1.1 Command Format

All PIP commands which involve the interchange (transfer) or data must have the following format:

DESTINATION=SOURCE <Terminator>

where:

a. The DESTINATION portion of a PIP command describes the device and file(s) which are to receive the transferred data. This portion of a command consists of either one file specification or a subset of a file specification.

b. The equals sign is a required delimiter in all PIP commands to separate the DESTINATION and SOURCE portions of the command.

c. The SOURCE side of the command describes the device from which the transferred data is to be taken. This portion of a command may contain one or more file specifications or subsets of file specifications.

d. A Terminator is required to end each PIP command. A RETURN entry (symbolized as <CR>) is normally used, however, any other paper-motion command may be used as a terminator.

PIP commands which do not require the transfer of information may be written using the form

DESTINATION=Terminator

The equals delimiter and a terminator are still required in commands

formatted in this manner despite the fact that only the destination portion of the command is used.

2.1.2 File Specification

A file specification contains all of the information needed to identify a file involved in a PIP function. It may consist of:

1. a device name;
2. a filename;
3. a directory identifier;
4. a protection code which is to be assigned to either a specified file, a User File Directory (UFD), or a SubFile Directory (SFD);
5. and an identifier to be assigned to the tape mounted on a specified DECtape unit.

The format of a PIP command containing all possible items of a file specification is:

dev:name.ext[directory]<nnn>↑ident↑=dev:name.ext[directory] <CR>

where:

1. DEV is either a physical device name (e.g., DSK, DTA1, etc.) or a logical device name (refer to paragraph 2.2).

2. NAME is a 1 to 6 alphameric character identification which is either to be assigned to a new file (NAME is on the destination side of the command) or which identifies an existing file (NAME is on the source side of the command). (Refer to paragraph 2.3 for a description of filenames.)

3. EXT is a 1 to 3- character extension assigned to the name of a file either by the user or by the system. (Refer to paragraph 2.3 for a description of filename extensions.)

4. [DIRECTORY] is the identifier of a specific directory (i.e., UFD or MFD) within the system. This identifier may consist of a project,programmer number pair and Sub File Directory (SFD) names. (See paragraph 2.4 for details.

5. <nnn> is a 3-digit protection code which is to be assigned to either one or more destination files or to a specified User File Directory[1]. (Refer to paragraph 2.5 for a description of protection codes.)

6. ↑IDENT↑ is a 1 to 6 character name which is to be given to the contents of a DECtape reel mounted on a specified DECtape unit. (Refer to paragraph 3.2.1.2 for details.)

---

[1]A User File Directory (UFD) is contained by the system for each user permitted access to it. A user's UFD is identified by his project,programmer number; it contains the names of all files belonging to the user together with pointers to the actual location of each file.

The manner in which each of the possible elements of a file specification may be used in either the destination or source portions of a PIP command is described in the following table:

Element	Destination	Source
dev.	Name of device onto which the specified file is to be written.	Name of device on which the specified file resides.
name	Name to be assigned to the copied file.	Name of the file to be copied.
.ext	User-specified file-name extension.	Current filename extension.
[directory]	Identification of the disk storage area which is to receive the file to be transferred.	Identification of the disk storage area which contains the file to be copied.

NOTE

The [directory] identifier must include a full directory path specification whenever sub-file directories are involved. For example [proj,prog,SFDA...SFDn]. (See paragraph 2.4 for more details.)

<nnn>	Protection code to be assigned to either a copied file or a specified UFD.	NOT PERMITTED IN SOURCE PORTION OF PIP COMMANDS.
↑ident↑	Name to be assigned to the tape mounted on a specified DECtape unit.	NOT PERMITTED IN SOURCE PORTION OF PIP COMMANDS.

File specifications may be delimited by:

1. an equals character (=) if the specification is on the destination side of the command string (e.g. dev:name.ext=...<CR>).

   NOTE

   PIP will accept a back-arrow entry (←) in place of the equals character (=).

2. a comma (,) if the specification is on the source side of the command string and is one of a series of file specifications. For example

   dev=dev1:name.ext,dev2:name.ext,name.ext,..name.ext<CR>

3. a RETURN <CR> entry if it is the last item on the source side of a command. For example

   dev=dev1:name.ext,dev2:name.ext,..devn:name.ext<CR>

2.1.3 Command String Delimiters

The delimiters which may be used to separate the elements of a PIP command string are described in the following table.

PIP COMMAND STRING DELIMITERS

Delimiter	Use and Description
:	The colon delimiter follows and identifies a device name. For example, the device DTA1 is specified as DTA1: in PIP commands.
[ ]	Square brackets are used to enclose the user DIRECTORY numbers and SFD names (if SFDs are used). For example [40,633] or [40,633,SFD1,SFD2,...SFDn] represent the manner in which DIRECTORY numbers can be written.
< >	Angle brackets must be used to enclose a protection code (e.g. <Ø57> which is to be assigned to either a file or a user file directory (UFD).
,	Commas are used to separate user project and programmer numbers, and file specification groups. For example dev:[4Ø,633]=dev:name.ext,name.ext<CR>
↑↑	A name to be assigned as an identifier to a DECtape is enclosed within a set of up-arrows (e.g. ↑MACFLS↑).
.	A period delimiter must be the first character of a filename extension. The form on an extension is .ext.
#	A number symbol is used as a flag to indicate the presence of an octal constant in a filename or a filename extension.
!	An exclamation symbol may be used to delimit a file specification. When used, the ! symbol causes control to be returned to the Monitor from PIP and the specified file (or program) to be loaded and run. This function is provided as a user convenience to eliminate the need for several control entries.
=	The equals character must be used to separate the destination and source portions of a PIP command.
( )	Parentheses are used to enclose magnetic tape options, PIP control switches, and one or more PIP function switches. The form of a command employing parentheses to enclose a series of switches is: dev:name.ext(sw1sw2..swn)=...<CR>

2.2 DEVICE NAMES

Both physical or logical device names may be used in PIP commands. The user must remember that a logical name takes precedence over a physical name when both are used in the same command.

2.2.1 Physical Device Names

Each standard DECsystem-10 peripheral device is assigned a specific device name consisting of a 3-character generic name plus either a unit number (0 to 777) or:

1) 3 characters,
2) 3 characters and a station number,
3) an abbreviated disk name or,
4) the name of a disk file structure.

A list of the generic physical device names is given below:

PERIPHERAL DEVICES

Device	Generic Physical Device Name
Card Punch	CDP
Card Reader	CDR
Console TTY	CTY
DECtape	DTA
Disk	DSK
Packs	DPx
Fixed-Head	FHx
Display	DIS
Line Printer	LPT
Magnetic Tape	MTA
Operator Terminal	OPR
Paper-tape Punch	PTP
Paper-tape Reader	PTR
Plotter	PLT
Pseudo-TTY	PTY
System Library	SYS
Terminal	TTY
Pseudo-device TMPCOR	TMP

2.2.2 Logical Device Names

A logical device name is a user-assigned designation which is employed in the preparation of a program in place of a specific physical device name. The use of logical device names permits the programmer to write programs which do not specify one particular device but may use, at run time, any available device which can perform the required function.

Logical device names may consist of from one to six alphanumeric characters of the user's choice.

2.3 FILENAMES

Filenames are file identifiers assigned either by the system (for system programs) or by the user. A filename may consist of a name field and an extension field but only a name field is required. Whenever both fields are used in a filename, it has the form name.ext. A period delimiter is required as the first character of the extension. Filename fields are defined as:

1. Name Field. Names of files may consist of from one to six alphanumeric characters or octal constants; in user-assigned names the characters may be arbitrarily selected by the user. Names generated by the user must be unique at least within the file structure in which the file is located.

2. Extension Field. Filename extensions may consist of up to three alphanumeric characters. Extensions are normally used to specify the type of data contained by the file identified by the filename field. Filename extensions which are recognized by the system and the type of data each specifies are given in Appendix A. In filenames, users may specify a standard extension (one recognized by the system), one which he has devised, or none at all. If no extension is given in a filename, the system may add one to the filename during PIP operations.

   PIP utilizes the filename extension given in a file specification to determine whether the file is to be transferred in a binary or ASCII mode. If it is all possible, PIP will transfer files in a binary mode since it is faster.

   In dealing with filename extensions PIP performs a specific series of tests in order to determine the mode which should be used during a requested transfer operation. The following mode determination tests are performed in succession until PIP obtains a firm indication as to the type of mode required:

   a) PIP tests for the presence of a data mode switch (see paragraph 3.4.). If no switch is found, PIP goes to the next test.

   b) PIP tests for the presence of a known (standard) filename extension which specifies a binary mode of transfer (see Appendix A). If no binary extensions are found, PIP goes to the next test.

c) PIP tests both the input and output devices specified to determine if they are both capable of handling binary data. If either or both of the devices cannot handle binary, the transfer is made in the ASCII mode. If both devices can handle binary data, PIP goes to the next test.

d) PIP tests for the presence of an X option switch (/X) in the command string; if it is found, the transfer is made in the binary mode. If an X option is not found, PIP goes to the next test.

e) PIP tests for the presence of commas (non-delimiters) in the command string; if commas are found an ASCII mode is indicated. If no commas are found, the transfer is made in the binary mode.

2.3.1 Naming Files with Octal Constants

Octal constants may be used as either a part of or all of a filename. In either of the foregoing cases, the first constant of each group of octal constants which appear in a filename must be preceded by the symbol #, and each group is delimited by a non-octal digit or a character. For example, the filenames:

1. #124ABC.ext (constants are used as part of a filename)
2. #12AB#34.ext (constants are intermixed with other characters)
3. #12467Ø.#123 (constants form the whole filename)

are all acceptable to PIP.

The symbol # is not regarded by PIP as part of the filename but is used only as a flag to PIP to indicate an octal constant.

The number of octal digits used in a filename or an extension should be even since two octal constants may be stored in a SIXBIT character. If an odd number of octal constants is given, PIP will add an extra Ø to the filename or extension. For example, the constant #123 would be expanded to #123Ø by PIP.

Names comprised of octal constants are left-justified by PIP. The following are examples of the use of octal filenames:

```
DTAØ1:#12467Ø.BIN=DSK:#1ØØØØØ.BIN<CR>
```

2.3.2 Wildcard Characters

The two symbols * and ? may be used in PIP to represent, respectively, complete fields and single characters. These symbols are referred to as wildcard characters; their use is described in the following paragraphs.

2.3.2.1 THE ASTERISK SYMBOL - The asterisk symbol * may be used to replace a filename or extension:

1. name field (e.g. *.ext),
2. extension field (e.g. name,*),
3. both filename fields (e.g., *.*).

For example, the filename FILEA.MAC, which specifies the MACRO source language file named FILEA, may be altered by the use of the asterisk in the following manner:

1. *.MAC specifies all files with the extension .MAC.
2. FILEA.* specifies all files with the name FILEA, and,
3. *.* specifies all files.

2.3.2.2 THE QUESTION MARK SYMBOL - The character ? may be used to indicate a wild character in file names and extensions. The symbol ? replaces characters of a filename to mask out any or all of the characters of a name, extension or both the name and extension fields of a file. When PIP processes a filename which includes ? characters, it ignores the wildcard characters. This masking capability enables the user to specify, with one command, groups of files whose filenames have common characters identically positioned within their filenames. For example, assume that the device DTA1 contains the files TEST1.BIN, TEST2.BIN, TEST3.BIN and TEST4.BIN; the user can specify all of these files with one file specification:

```
DTA1:TEST?.BIN
```

2.3.2.3 COMBINING * AND ? WILDCARD SYMBOLS - The symbols * and ? can be combined in filenames to specify specific groups of files which have common characteristics in either or both of their name or extension files.

For example, the file specification

```
ABC???.*
```

specifies all files having the character group ABC as the first three characters of its filename. Again, the file specification

```
*.??A
```

specifies all files having an extension which has the character A as its third character.

In combining the * and ? symbols, the user should remember that for:

a. filenames, * is equivalent to ??????, and
b. extensions, * is equivalent to ???.

For example, the filenames *.* and ??????.??? are equivalent.

2.4 DIRECTORY IDENTIFIER

The [directory] identifier is used in PIP commands to identify a specific:

a) User File Directory (UFD),
b. Sub File Directory (SFD), or
c) a specific UFD-SFD directory path.

The item identified by a given [directory] identifier can be a directory or an item located within a directory which belongs to either the current user or, when the protection code scheme permits, to another user. (Refer to paragraph 2.5 for a description of protection codes.)

A [directory] identifier can consist of a project,programmer number pair (abbreviated as proj,prog) and the names of SFDs. The most expanded form of the [directory) identifier is:

```
[proj,prog,SFD1,SFD2,...SFDn]
```

As shown, a [directory] identifier is always enclosed within square brackets and its elements are delimited by commas.

2.4.1 UFD-Only Identifiers

Each UFD is identified in the system by the project,programmer number pair assigned to the user for whom the UFD was created. A [directory] identifier for a UFD has the form

[proj,prog]

UFD [directory] identifiers may be written without either one or both of the project,programmer numbers. In such cases, PIP assumes either a previously specified default number or the number assigned to the current user. For example, assume that the current user is logged in under the number pair [57,124] and that no default identifier has been specified. The current user can use [directory] identifiers having any of the following formats:

	The Format:	Which is Interpreted by PIP as:
1)	[,]	[57,124]
2)	[57,]	[57,124]
3)	[,124]	[57,124]

2.4.2 SFD (Full Directory Path) Identifiers

A Sub File Directory (SFD) is identified by its user-assigned name plus the project,programmer number pair which identifies the UFD in which it is located. A [directory] identifier for an SFD then has the form

[proj,prog,SFDname]

Whenever an SFD is located in a UFD which has a multi-level directory arrangement, the UFD containing the desired SFD must be included in the [directory] identifier for the desired SFD. A [directory] identifier for an SFD in a multi-directory level UFD has the form

[proj,prog,SFD1,SFD2,...SFDn]

and is referred to as a full directory path identifier. For example, assuming that the current UFD is identified by the proj,prog number pair 57,124 and has the following directory organization:

Level 1		UFD	
Level 2		SFDA	
Level 3	SFD1		SFDB
Level 4	SFD2		SFDC

the [directory] identifier for SFD2 is written as

[57,124,SFDA,SFD1,SFD2]

The proj,prog number pairs in full directory path identifiers may be written using the format variations described in paragraph 2.4.2. However, when no proj,prog numbers are specified by the user, two commas must be used in the identifier in the following manner

[,,SFD1,...SFDn]

The first comma represents the delimiter between the proj,prog numbers; the second represents the delimiter between the last number (prog) and the first SFD name.

2.4.3 Specifying Default and Current [Directory] Identifiers

The position in which a [directory] identifier is given in a PIP command determines if it is viewed as a default identifier for all subsequent file specifications given in that command or is the current identifier for an individual file specification.

If a [Directory] identifier is given before one or more file specifications of a command it regarded as the DEFAULT identifier for those specifications. For example, in a command segment having the form:

[directory A] File Specification 1,File Specification 2

the identifier [directory A] is the default for both File Specifications 1 and 2.

If a [Directory] identifier is given after the filename within a File Specification it is viewed as the current identifier for that file specification and will override any given default [directory]. The form of a file specification with the current identifier specified is:

dev:filename.ext[directory]

Both default and current [directory] identifiers can be specified in the same PIP command. For example, the PIP command source segment:

```
=dev:[directory A]filename.ext,dev:filename.ext[directory B]<CR>
```

is valid. In the foregoing example, the identifier [directory A] is the default identifier for the first file specification; and will act as the default identifier for the second file specification if [directory B] is not given. When [directory B] is given, it overrides the default identifier and is accepted as the identifier for the second file specification.

2.5 FILE ACCESS PROTECTION CODES

Three-digit (octal) protection codes which specify the degree of access that each of three possible types of users may gain to a file can be specified in the destination side of a PIP command string. File access protection codes are written within angle brackets and must contain three digit positions (e.g., <nnn>). Each digit within a protection code specifies the type of access a specific type of user may have to the file or files involved. Considering the protection code <n1n2n3> the digits give the file access code for the following types of users:

a. n1 = File OWNER
b. n2 = project MEMBER, and
c. n3 = OTHER system users.

The user types are defined as follows:

1. FILE OWNERS. Users who are logged in under either:

   a. the same programmer number as that of the UFD which contains the file; or

   b. the same project and programmer number as associated with the UFD which contains the file..

   The decision as to which of the above items defines an OWNER is made at Monitor Generation time.

2. PROJECT MEMBER. Users who are logged in under the same project number as that which identifiers the UFD containing the file.

3. OTHER USERS, any user of the system whose project and programmer number do not match those of the UFD containing the file in question.

File access protection codes are placed in PIP commands after the destination filename of the file involved. For example, the command

DPA3:FILEA.BIN<nnn>=DSK:SOURCE.BIN<CR>

copies the contents of file SOURCE.BIN onto disk pack device DPA3 under the name FILEA.BIN with an assigned file protection code of nnn.

2.5.1 Digit Numeric Protection Code Values

Each of the digits in a 3-digit file protection code may be assigned an encoded numeric value ranging from Ø to 7. The meaning of each octal value is:

Code Value	Permitted Operations
7	No access privileges. File may be looked up if the UFD permits.
6	Execute only.
5	Read, execute.
4	Append, read, execute.
3	Update, append, read, execute.
2	Write, update, append, read, execute.
1	Rename, write, update, append, read, execute.
Ø	Change protection, rename, write, update, append, read, execute.

Files are afforded the greatest protection by the code value 7; the least protection by Ø. It is always possible for the owner of a file to change the access protection associated with that file even if the owner-protection field is not set to Ø; thus, the values Ø and 1 are equivalent for the owner. Files with their owner-protection field set to 1 are preserved (i.e., saved by .KJOB/K).

It is recommended that important files such as source files be assigned an owner-protection code of 2. This level of protection will prevent the file from being accidentally deleted by permitting them to be edited.

2.6 UFD AND SFD PROTECTION CODES

When a user directory (UFD or SFD) is created, it is assigned a 3-digit octal access protection code by either the owner of the file or, by default, the system. The 3-digit code specifies the type of access permitted to the directory by each of the three possible classes of users (i.e., OWNER, MEMBER, or OTHER). (Refer to paragraph 2.5 for a description of user classes.)

Once assigned, a directory access protection code may be changed by the owner and, if the protection code permits (i.e. CREATES allowed), by users other than the owner. (Refer to the description of the PIP rename option given in paragraph 3.5.3.1 for the procedure required to change directory protection codes.)

The access protection code assigned each user class may range from 0 through 7; the following table lists the codes and the operations which each permits.

CODE	PERMITTED OPERATION(S)
0	Access not permitted.
1	The directory may be read as a file.
2	CREATEs are permitted.
3	The directory may be read as a file and CREATEs are permitted.
4	LOOKUPs are permitted.
5	The directory may be read as a file and LOOKUPs are permitted.
6	CREATEs and LOOKUPs are both permitted.
7	The directory may be read as a file and both CREATEs and LOOKUPs are permitted.

SECTION 3

STANDARD PIP SWITCHES

3.1 OPTIONAL PIP FUNCTIONS

PIP provides the user with a group of optional functions which can be executed during the performance of the primary PIP transfer function.

Each optional function is assigned an identifier which, when added as a "switch" to a PIP command, initiates the execution of the identified function.

For the purposes of this manual, the PIP optional functions are divided into standard and special groups. The standard group of options described in this section consist of switches which:

1. determine which files are transferred;
2. edit all the data contained by each source file;
3. define the mode of transfer;
4. manipulate the directory of a directory-type device.

All optional functions which deal with non-directory devices and which perform functions other than those listed above are considered special and are described in Section 4.

3.1.1 Adding Switches to PIP Commands

All switches in PIP commands must be preceded by a slash (i.e., /sw); for example, the optional function identified by the letter w is added to a PIP command:

```
*DTA1:DESTFL.BIN/w=DSK:FILEA.BIN,FILEB.BIN<CR>
```

When more than one switch is to be added to a command, they may be listed either separated by slashes (e.g., /B/X....) or enclosed in parentheses (e.g., (BX)).

3.2 BASIC TRANSFER FUNCTION

The basic function performed by PIP is the interchange (i.e., read/write transfer) of files or data blocks between devices. There are two types of transfer operations:

1. An optional X-switch transfer in which the source files or blocks are transferred as separate files to the destination device.

2. A non-X type in which all files or blocks transferred from the source device are combined (i.e., concatenated) into a single file on the destination device.

3.2.1 X-Switch Copy Files Without Combining

The use of the X-switch enables the user to move (copy) a group of source files onto the destination device as individual files without changing their creation dates, time dates, filenames and filename extensions. The following are examples of how the X-switch is used in PIP:

1. To transfer all the user's disk files to a DECtape, type:

   ```
   DTA1:/X=DSK:*.*<CR>
   ```

   Assuming that there are three files on the user's disk area named FILEA, FILEB, FILEC.REL, these files will be transferred to DTA1 and can be referenced on DTA1 by those names.

   One significant difference between the disk and all other devices is file protection. If the disk is the source device, PIP will by-pass those protected files to which the current user is not permitted access. A suitable message is then issued by PIP if the rest of the command string is successfully executed. Similar processing is described later for the L, Z and D switches. If none of these switches is given, a requested DSK file which is protected will cause termination of the request.

2. To transfer all the files from card reader to disk, type:

   ```
   DSK:/X←CDR:*<CR>
   ```

   When transferring files from the card reader with the * command, the input files must either be wholly ASCII or wholly binary.

3. To transfer two specific files from user [11,7]'s disk area to a DECtape, type:

```
DTA2:/X=DSK:[11,7]FILEA,REL.FILEA.MAC<CR>
```

4. To copy files from a paper tape onto a directory-type device, the user may employ either:

   a. A copy command in which the number of files to be read are specified by adding a series of commas to the command after the source device name (i.e., PTR,,,,,,,). The number of commas required is always one less than the total number of files to be transferred. For example, the command:

```
DSK:/X=PTR:,,,,<CR>
```

   specifies that five (5) files are to be copied from paper tape and written, individually, into the current user's disk area.

   b. A copy command in which all the files contained by a paper tape are to be copied onto a specified device. For example, the command

```
DSK:/X=PTR:*<CR>
```

   specifies that all files contained on the paper tape loaded as PTR are to be copied into the current user's disk area. Whenever a command of this type is used, the last file on the paper tape must be followed by two consecutive end-of-file codes.

NOTE

In both the foregoing examples, PIP will generate any needed destination filenames. This function is described in paragraph 3.2.1.1.

Whenever the X-switch is used and is not combined with an editing option, PIP transfers any file involved as it appeared on the source device. X-switch operations are copy operations and are referred to as such.

3.2.1.1 NON-DIRECTORY TO DIRECTORY COPY OPERATION - In copying files from a non-directory device onto a directory-type device, PIP must perform special operations in naming the destination files. For example, a special case of source and destination filenames arises in the command:

```
DTA2:FNME.EXT/X=MTAØ:*<CR>
```

Here, every file is to be copied from a non-directory device (MTAØ) to a directory device (DTA2) without combining files (/X). Only one destination filename is given (i.e., FNME.EXT) but the source device (MTAØ) may contain more than one file. If more than one file is transferred, it is necessary for PIP to generate a unique filename for each copied file. PIP generates filenames by developing a 6-character name field in which the first three characters are either:

1. the first three characters of a given destination filename, or
2. the characters "XXX" if no destination filename is given in the command.

The second portion of the PIP-generated name field consists of the decimal numbers ØØ1 through 999 which are added, in sequence, to each filename developed during the /X copy operation.

For filename extensions, PIP uses either the extension of a given destination filename or a null field if no filename is given in the command.

For example, assuming that three files are present on MTAØ, the command:

```
DTA2:FNME.EXT/X=MTAØ:*<CR>
```

transfers the files to DTA2 and establishes the following names in the DECtape directory for the files copied:

1. FNMØØ1.EXT,
2. FNMØØ2.EXT,
3. FNMØØ3.EXT.

If, in the above example, the command given did not include a destination filename (i.e., DTA2:/X=MTAØ:*<CR>) the copied files would have been named:

1. XXXØØ1
2. XXXØØ2
3. XXXØØ3

The use of the 3-digit decimal number for the last three characters of the filename name gives the user 999 possible input files from non-directory devices. If PIP finds more than 999 files on the source device it will terminate the transfer operation after the 999th file is copied and will issue the error message

?TERMINATE/X,MAX OF 999 FILES PROCESSED.

Any error messages referring to individual files named by PIP (either input or output) will use the generated filename.

3.2.1.2 ASSIGNING NAMES TO DECTAPE TAPES - A tape mounted on a specified DECtape unit can be assigned an identifier during copy operations. Identifiers are from 1 to 6 character names (any SIXBIT character - except ↑ - within the code range 40-137 can be used) which are added to the DECtape's directory (128th word). DECtape identifiers can be read by PIP, FILEX and DIRECT programs; the Monitor does not read identifiers. A DECtape identifier is assigned by adding the selected name to a PIP command when the DECtape to be named is mounted on the specified destination device.

The format required for a DECtape identifier is

↑name↑

A DECtape identifier is inserted into a PIP command following the given destination device name:

dev:↑name↑=source file specification(s)

For example, the command

DTA3:↑MYFILE↑/X=DTA1:.*

specifies that the DECtape on device DTA3 be given the identifier "MYFILE" and receive copies of all the files contained by the tape on device DTA1.

3.2.2 DX-Switch, Copy All But Specified Files

When the DX-switch is added to a PIP command it causes all the files to be copied from the source device to the destination device except

those files which are named in the command string. If the source device is DSK, a maximum of 10 source-file specifications are allowed. Only directory-type devices are allowed as source devices; no check is made on the existence of the files which are not to be copied. Only one source device is permitted; for example, the command

```
DTA1:(ZDX)=DSK:*.LST,*.SAV,CREF.CRF<CR>
```

zeroes out the directory of DTA1 and transfers to DTA1, from the disk, all files except CREF.CRF and all files with either the extension .LST or .SAV.

3.2.3 Transfer Without X-Switch (Combine Files)

When the X-switch is not included in a PIP command all files or blocks transferred from the source device are combined into a single file on the destination device. For example:

1. To combine three paper tape files into one, type

   ```
   PTP:=PTR:,,<CR>
   ```

2. To combine two files on DECtape into one on another DECtape, type

   ```
   DTA3:FILCOM=DTA2:FILA,FILB<CR>
   ```

3. To combine files from two DECtapes into one on the user's disk area, type

   ```
   DSK:DSKFIL=DTA2:ONE,DTA4:TWO.MAC<CR>
   ```

4. To combine all the files on MTAØ into one file on the user's disk area, type

   ```
   DSK:TAPE.MAC=MTAØ:*<CR>
   ```

   (This assumes that MTAØ is positioned at the Load Point).

3.2.4 U-Switch, Copy DECtape Blocks Ø, 1 and 2

The U-switch is used during DECtape-to-DECtape copy operation to specify that Blocks Ø, 1 and 2 of the source tape are to be copied onto the destination tape.

This switch is commonly used to transfer DTBOOT from one tape to another. For example, the command:

```
DTA1:/U=DTA5:<CR>
```

transfers blocks Ø through 2 of DTA5 to DTA1.

3.3.1 A-Switch, Integral Output Lines (Line Blocking)

The use of the A-switch (/A) in a PIP command specifies that each output buffer is to contain an integral number of lines, no lines are to be split between physical output buffers. Line blocking is required for FORTRAN ASCII input. Each line starts with a new word.

3.3.2 C-Switch, Delete Trailing Spaces and Convert Multiple Spaces to Tabs

The addition of a C-switch (/C) to a PIP command causes groups of multiple spaces in the material being copied to be replaced by one or more TAB codes; trailing spaces are deleted.

The conversion of the spaces to TAB codes is performed in relation to the standard line TAB "stop" positions located at 8-character intervals throughout the line. Only those groups of multiple spaces which precede a TAB "stop" will produce a TAB code. For example:

1. [space][stop]--will not produce a TAB code.
2. [space][space][stop]--will produce [TAB].
3. [space][space][stop][space][space]--will produce [TAB] [space][space]

A totally blank input line is replaced by one space when this switch is used. The C-switch is used to save space when storing card images in DSK file structures. The conversion of spaces to tabs must be done with care since it could alter Hollerith text.

3.3.3 E-Switch, Ignore Card Sequence Numbers

This switch, normally used when a card reader is the source device, causes characters (i.e., columns) 73 through 8Ø of each input line to be replaced by spaces.

3.3.4 N-Switch, Delete Sequence Number

This switch causes line sequence numbers to be deleted from any ASCII file being transferred. Line sequence numbers are recognized

as any word in the file in which bit 35 is a binary 1 and follows a carriage return, vertical TAB, form feed for start-of-file identification. Nulls used to fill the last word(s) of a line are ignored. If a line sequence number is followed by a TAB, the TAB is also deleted.

3.3.5 S-Switch, Insert Sequence Numbers

This switch causes a line sequence number to be computed and inserted as the output buffer at the start of each line. Sequence numbers are indicated by a 1 in bit 35 of a word following a carriage return, a vertical TAB or start-of-file indicator.

Sequence numbers assigned by PIP take the form nnnnn, starting at ØØØ1Ø and ranging through 999Ø in increments of 1Ø. Approximately one-third of each output buffer is left blank to facilitate editing operations on the file (DTA only).

3.3.6 O-Switch, Insert Sequence Numbers and Increment By 1

This switch causes the same operations to be performed as those for switch S, (see 3.3.5) except that the assigned sequence numbers are incremented by 1 instead of 1Ø.

3.3.7 P-Switch, Prepare FORTRAN Output for Line Printer Listing

This switch causes PIP to take output generated by a FORTRAN program, which was output on a device other than the line printer (LPT), for which it was intended, and performs the carriage control character interpretations needed when the data is sent to the LPT. The first character in each input line is interpreted by PIP according to the following table.

FORTRAN CARRIAGE CONTROL CHARACTER INTERPRETATION

Carriage Control Character Produced by FORTRAN Program	ASCII Character(s) Substituted	Line Printer Action
space		Skips to next line (single space) with a FORM FEED after every 6Ø lines.
*	Ø23	Skips to next line with no FORM FEED.
+	Ø15	Precede line with a carriage return only (i.e., over-print previous line).
, (comma)	Ø21	Skips to next 1/3Øth of page.
=	Ø15,Ø12,Ø12	Skips two lines.
.	Ø22	Skips to next 1/2Øth of page.
/	Ø24	Skips to next 1/6th of page.
Ø	Ø15,Ø12	Skips 1 line (double space).
1	Ø14	Skips to top of next page (page eject).
2	Ø2Ø	Skips to next 1/2 page.
3	Ø13	Skips to next 1/3 page (also vertical tab).

3.3.7.1 COPY FORTRAN BINARY FILES - The binary mode switch (/B) can be combined with /P in a PIP command to enable the user to obtain a copy of a FORTRAN binary file. The /B/P switch combination is needed when copying FORTRAN binary file(s) from a DECtape source onto a Disk in order to insert a needed control word into each physical buffer. The /B/P switch combination is not needed if both the source and destination devices have the same buffer size. The format for a FORTRAN binary file copy command is

dev:name.ext/B/P=dev:name.ext...<CR>

3.3.8 T-Switch, Delete Trailing Spaces

This switch causes all trailing spaces to be deleted from the file being transferred. If a transfer line consists of nothing but spaces, then a single space and a line terminator will be retained in its place in the copied file.

3.3.9 W-Switch, Converts Tabs to Spaces

The addition of a W-switch (/W) to a PIP command causes each TAB code contained by the material being copied to be converted to one or more sequential spaces.

The number of spaces produced when a TAB code is converted is determined by the position of the TAB in relation to the standard line TAB "stops". Each line has TAB stops positioned at 8-character intervals throughout the length of the line. When a TAB is converted in a /W switch operation, only enough spaces are produced to reach the next sequential line TAB stop position. For example, the series

[stop]ABCD[TAB]

is converted to

[stop]ABCDspspspsp[stop]

where:

sp = space.

The use of the W-switch causes files previously edited by the use of a C-switch to be restored to their original form (less the deleted trailing spaces).

3.3.10 V-Switch, Match Angle Brackets

This switch is not a true edit switch, because the input file is not edited. The use of this switch generates an output file which contains the results of cumulative matching of angle brackets located in the input file. If a line in the input file contains brackets which are not needed to match earlier brackets and which match each other, no output occurs. In all other cases where brackets occur,

a cumulative total and the line currently considered are printed. The symbol > scores a negative count; the symbol < scores a positive count. A typical use for this switch is to check source input to the MACRO-1Ø Assembler; for example, assuming that the file A contains:

```
ONE<<>
TWO<
THREE>
FOUR<>>
FIVE<>
SIX>
```

The request

```
LPT:=DTA2:A/V<CR>
```

results in the Line Printer output:

```
 1 ONE<<>
 2 TWO<
 1 THREE>
 Ø FOUR<>>
-1 SIX>
```

From this general example, the most likely conclusion is that there is either a < missing or an extra > in this file. Line five (i.e., FIVE <>) was not printed because the brackets which it contained were matched.

3.3.11 Y-Switch, DECtape to Paper Tape

The Y-switch enables the user to transfer DECtape files having the filename extension .RMT, .RTB or .SAV onto SAVE-formatted RIM1Ø or RIM1ØØ paper tapes. The type and contents of the paper tape produced in a Y-transfer is determined by the source file filename extension. If the extension is:

1. .RMT, - A RIM1Ø paper tape (with terminating transfer word) is produced;
2. .RTB, - A RIM1ØØ paper tape (with RIM loader and terminating transfer word) is produced;
3. .SAV, - A RIM1ØB paper tape is produced (with neither RIM loader nor terminating transfer word).

For example, the command

```
PTP:/Y=DTA2:TESTI.RTB<CR>
```

will punch a RIM10B tape as described in item 1 of the foregoing description from DECtape file TESTI.RTB.

Switches D and X may be used in conjunction with the Y-switch.

It is assumed that .RTB, .RMT and .SAV files are all in the standard "save" file format. In particular, it is assumed that no block of an .RMT saved file overlaps a preceding one.

NOTE

Optional switch Y is obtained by setting RIMSW=1 at assembly time (see source file PIP.CTL.).

The functions performed by PIP during /Y transfers in response to each possible type of source file filename extension are:

1. An .RTB file causes PIP to:

   a. Punch a RIM loader.

   b. Punch an I/O word (-n,x) at the start of each data block. The variable n is the number of data words punched in each block and has the octal value 17, or less. The variable x is the starting address-1 for loading the following data. Successive values of x are derived from the pointer words in the DECtape blocks. The first value of x is the value of the right side of the first pointer word in the DECtape file.

   c. The complete DECtape file is punched as described in item b.

   d. The final block punched is followed by a block containing a transfer word. If the right half of .JBSA contains 0 then a halt is punched. If the right half of .JBSA contains a non-zero value, a jump to that address is punched.

2. A .SAV file is treated in the same way as one having .RTB extension except that no RIM loader and no transfer word are punched.

3. An .RTM file initiates PIP functions which are similar to those described for .RTB files but which have the following differences:

   a. Only one IOWD is produced, (-n,x) where (n-1) data words and a transfer instruction follow.

   b. The first of the (n-1) data words punched from the saved file is the first word of the logical block which contains location .JBDA (i.e., the first location after the end of the JOBDATA area).

c. The variable x is then set to the starting address (address-1) of the first data word found. The effective program length is determined by the relationship n=(.JBFF)-x. Data is now transferred from (x+1) until (n-1) words have been punched.

d. Zero fill is used if a pointer word in a source block indicates noncontinuous data. The transfer word, calculated as described for .RTB files terminates the output file.

3.4 SET DATA MODE, SWITCHES B, H AND I

The addition of optional data mode switches to a PIP command specifies the mode in which the file(s) involved must be transferred.

Data modes are device dependent; complete descriptions of their use and effect on different devices are given in the DECsystem-1Ø Monitor Calls manual.

If both input and output devices can do binary I/O, no editing switches are in force and no concatenation is required. All files are transferred in binary mode (36-bit bytes). If an editing switch that requires PIP to do character processing is used, ASCII mode is used. The data mode switches are:

1. /B - initializes the input and output devices in binary mode.

NOTE

Since PIP recognizes the following as binary extension, /B is not required when these extensions are used in the PIP command.

Binary Extensions Recognized by PIP

.BIN	.HGH	.RES
.CHN	.INI	.SAV
.CKP	.LOW	.SFD
.DAF	.QUC	.SHR
.DAT	.QUD	.SYS
.DCR	.QUE	.UFD
.DMP	.QUF	

2. /H - initializes the input and output devices in image binary mode.

3. /I - initializes the input and output devices in image mode.

3.5 FILE DIRECTORY SWITCHES

Optional PIP switches whose functions affect user file directories are described in paragraphs 3.5.1 through 3.5.6.

3.5.1 L-Switch, List Source Device Directory

NOTE

The Monitor command DIRECT provides the user with more facilities for obtaining directory-type information than the PIP L-switch option (refer to the DECsystem-1Ø Monitor Command Manual for details).

This switch enables the user to obtain a listing of the source device directory. The type of output device used affects the directory listing as follows:

1. If the output device is TTY, the directory listing formats for directory-type devices are:

   a. For DTA source (e.g., TTY:=DTA4:/L<CR>)

      n FREE BLOCKS LEFT
      filename.ext no. of blocks creation date
      .
      .
      .
      .

   b. For DSK source (e.g., TTY:=DSK:/L<CR>)

      DIRECTORY [directory] (CURRENT TIME) (TODAY'S DATE)
      where [directory] is the project-programmer number of the requested directory.

      filename.ext<protection>no.of blocks creation date
      .
      .
      .
      .
      Total Blks n

Asterisk or question mark wildcard symbols (refer to paragraph 2.3.2.2) can be used in either the specified filename or extension fields to cause only those files in the disk directory of a particular filename or extension to be listed. Thus, the command TTY:/L=DSK:*.REL<CR> causes only those files with extension .REL to be printed in the directory listing.

2. If the output is not TTY, the directory listing is printed in one of the following formats:

   a. For DTA, source format is as in paragraph 1.(a)

   b. For DSK, source format is as in paragraph 1.(b) but includes access date and mode as well as the creation time and access date. If any disk file is protected, as much information as possible is given about it.

3.5.2 F-Switch, List Limited Source Directory

This switch performs, essentially, the same function as the L-switch; however, only the filenames and extensions of the files in the specified disk or DECtape directory are listed.

NOTE

The Monitor command DIRECT provides the user with more facilities for obtaining directory-type information than the PIP F-switch option (refer to the DECsystem-1Ø Monitor Command Manual for details).

Only DSK: and DTAn: are permitted as source device; if no source device is given, DSK: is assumed.

For example, the command

```
TTY:/F=<CR>
```

lists the directory of the user's disk area as described. The /F switch may work in cases where /L will not because of file access protection.

3.5.3 R-Switch, Rename Source Files

The use of this switch causes PIP to rename the source file to the name given as the destination file name. Only one source file specification can be given. If more than one is given, the error message PIP COMMAND ERROR is printed and no action is taken. The destination file specification can take the following forms (protection can always be specified):

1. Filename.extension
2. Filename.*
3. *,Extension
4. *.*<protection>

5. Filename
6. ??????.ext
7. ??????.???
8. *.???
9. ??????.*

In fact, <protection> can always be specified but the request *.* (4) has no effect without it. If no protection is specified, the current file protection is not altered.

During a rename operation on device DSK, if PIP finds that the filename to be changed exists on more than one file structure, PIP will output the following message to the user's terminal:

?AMBIGUOUS[file structure list][filename.ext]

The following are examples of the proper use of the /R switch:

1. DSK:MONI.F4/R=MONI.MAC<CR>

   Rename the file MONI.MAC as MONI.F4.

2. DSK:MON2,*/R=MONA.*<CR>

   Rename all files of name MONA and any extension to retain the extensions but take the new name MON2.

3. DSK:*.EXT/R=*.MAC<CR>

   Rename all files of extension MAC to retain their own names but take the extension EXT.

4. DSK:*.*<Ø77>/R=*.SAV<CR>

   Give all files of extension SAV the protection Ø77

5. DTA1:MON2/R=MONA.REL<CR>

   Rename the file MONA.REL to have the name MON2 and the null extension.

3.5.3.1 CHANGING SOURCE UFD OR SFD PROTECTION CODE USING THE RENAME (R) FUNCTION

- The access protection codes assigned to UFDs or SFDs can be changed using the PIP rename switch (/R) if the privileges assigned the current user permits the operation. (Refer to the DECsystem-10 Monitor Calls manual for a detailed description of user UFD and SFD access privileges.) The owner of a directory is always permitted the use of the PIP rename function.

The command format required to change a directory access protection code is

*dev:[directory].UFD<nnn>/R=[directory].UFD<CR>

where:

1. <nnn> represents the desired (new) protection code.
2. [directory] must be the same on both sides of the command.
3. The user indicates to PIP that the protection code of the identified directory (UFD or SFD) is to be changed by specifying the extension .UFD without a filename. Note that the same extension, .UFD, is used when changing the access protection of an SFD as well as for changing the protection of a UFD.

The following examples illustrate the use of the /R switch in changing the access protection codes of directories.

1. The command:

   DSKA:[57,123].UFD<222>/R=[57,123].UFD<CR>

   changes the access code of the UFD identified by the number pair 57,123 to /222.

2. The command

   DSKA:[57,123,AAA,BBB,111].UFD<222>/R=[57,123,AAA,BBB,111].UFD<CR>

   changes the access code of the SFD named 111 to the value 222. Note that the last name given in the [directory] identifier is the SFD which is affected by the /R operation.

3.5.4 D-Switch, Delete Files

This switch causes PIP to delete one or more specified files from the device given in the destination side of the PIP command. Only one device can be specified in a delete command; it is assumed that the source and destination devices are the same device.

For example, the following command

DSK:/D=FILEA,FILEB,FILEC.MAC,*.REL<CR>

causes PIP to delete from the user's disk area files FILEA, FILEB, FILEC.MAC and all files having the extension .REL.

If a nonexistent file is specified in a delete command, PIP prints the error message

```
%filename.ext FILE WAS NOT FOUND
```

and continues to process deletions of the existing specified files. If an existing file is found to be protected it will be skipped and the message

```
?filename.ext (2) PROTECTION FAILURE
```

is printed. If a user has the correct privileges he can delete files from other users' areas.

NOTE

An attempt to delete files from a DECtape that is write-locked results in the error message

```
DEVICE dev.name OPR operator station no.
ACTION REQUESTED
```

being printed at the user's terminal. When a system operator has write-enabled the DECtape unit involved, he will start the requested action and cause the message

```
CONT BY OPER
```

to be printed at the user's terminal.

On completion of a disk delete operation, PIP lists the names of the files deleted and the total number of blocks freed by the deletion.

For example, assume that a file three blocks in length and named FILEA.MAC exists in the current UFD: the command for its deletion and the subsequent messages printed by PIP would appear as:

```
*DSK:/D=FILEA.MAC <CR>      (user command)
FILES DELETED:              (PIP response)
FILEA.MAC                   (PIP response)
3 BLOCKS FREED              (PIP response)
*
```

3.5.5 Z Switch, Zero Directory

The use of this switch causes PIP to zero out the directory of the destination's device; a source device does not have to be specified in the command. A Z-switch request is implemented before any other operation specified in the command string in which it occurs. Thus,

```
DTA2:CARDS/Z=CDR:<CR>
```

zeroes out the directory of DTA2 before transferring one file from CDR onto DTA2. The command

```
DTA2:/Z=<CR>
```

zeroes out the directory of DTA2.

If the destination device is the disk, an attempt is made to delete all the files whose names are found in the directory specified. If protection codes prohibit the deletion of some of the files, the request will terminate after as many files as possible have been deleted, and the message

```
?filename.ext(2)PROTECTION FAILURE
```

is printed. The user should then change the protection of the protected files and repeat his request if he wants all files deleted. For example, the command

```
DSK:FLOUT/Z=DTA2:CARY<CR>
```

zeroes out the directory of the user's disk area, transfers file CARY from DTA2 to the disk, and names the disk file FLOUT.

3.5.6 Q-Switch, Print Summary of PIP Functions

This switch causes PIP to print on a specified device the system device file SYS:PIP.HLP. This file contains an alphabetical list of all PIP switches and functions. For example, the command

```
LPT:/Q=<CR>
```

causes the following summary to be listed on the line printer:

PIP	Switches (Alphabetic order) Summary
A	Line Blocking
B	Binary Processing (Mode)
C	Suppress Trailing Spaces, Convert Multiple Spaces to TABs
D	Delete File
E	Treat (Card) Columns 73-80 as Spaces
F	List Disk or DTA Directory *Filenames and Ext. only)
G	Ignore I/O Errors
H	Image Binary Processing (Mode)
I	Image Processing (Mode)
J	Punch Cards in 029 (Output Device must be CDP)
L	List Directory
M	See MTA Switches Below
N	Delete Sequence Numbers
O	Same as /S switch, except Increment is by 1
P	FORTRAN output Conversion assumed. Convert format control character for LPT listing. /B/P FORTRAN Binary
Q	Print (this) List of Switches and Meanings
R	Rename File
S	Resequence, or Add Sequence Number to File; increment is by 1Ø
T	Suppress Trailing Spaces Only
U	Copy Block Ø (DTA)
V	Match parentheses (<>)
W	Convert TABs to Multiple Spaces
X	Copy Specified Files
*Y	RIM, DTA to PIP if source extension is RTB Destination format is RIM Loader, RIM 1ØØ file transfer. If source extension is SAV destination format is as RTB - RIM 1ØB file only. If source extension is RMT destination format is RIM1Ø.
Z	Zero Out Directory

MTA switches:

Enclose in parentheses ( ).

M followed by	8	means select 8ØØ B.P.I. Density
	5	556 B.P.I. Density
	2	2ØØ B.P.I. Density
	E	Even Parity
	A	Advance MTAl File
	D	Advance MTAl Record
	B	Backspace MTAl File
	P	Backspace MTAl Record
	W	Rewind MTA or DTA
	T	Skip to Logical EOT
	U	Rewind and Unload MTA or DTA
	F	Mark EOF

(M#NA), (M#NB), (M#ND), (M#NP) mean advance or backspace MTAn files, or records.

*This is an optional switch obtained by setting RIMSW=1 at assembly time.

3.6 PERMITTED SWITCH COMBINATIONS

The combinations of PIP's standard and special option switches which are permitted in PIP commands are illustrated in the following matrix.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	Notes
A		#	✓	#	✓	#	✓	#	#	?		#	✓	✓	✓	?	✓	#	✓	✓		?	✓	✓	#	✓	ASCII mode
B	#		#	#	#	#	✓	#	#	#		#	✓	#	#	*	✓	#	#	#		#	#	✓	#	✓	Binary mode
C	✓	#		#	✓	#	✓	#	#	✓			✓	✓	✓	✓	✓	#	✓	✓		?	✓	✓	#	✓	ASCII mode
D						#																		*		#	Delete only
E	✓	#	✓	#		#	✓	#	#	✓			✓	✓	✓	✓	✓	#	✓	✓			✓	✓	#	✓	ASCII mode
F																											List Directory only
G	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	Always legal
H	#	#	#	#	#	#	✓		#				✓	#	#	#		#	#	#			#	✓		✓	Binary mode
I	#	#	#	#	#	#	✓	#					✓	#	#	#		#	#	#			#	✓		✓	Binary mode
J	✓	#	✓	#	✓	#	✓	#	#				✓	?	?	?		#	?	✓	#		✓	✓	#	?	ASCII mode
K																											Unused
L						#								✓	✓	✓	✓	✓	✓	✓	#	✓	✓	✓		✓	List Directory only
M	✓	✓	✓	#	✓	#	✓	✓	✓	?		✓		✓	✓	✓	✓	✓	✓	✓	#	✓	✓	✓		?	Magnetic Tape Only
N	✓	#	✓	#	✓	#	✓	#	#	?			✓		?	?	✓	#	?	✓			✓	✓		✓	ASCII mode
O	✓	#	✓	#	✓	#	✓	#	#	?			✓	#		?	✓	#	#	✓			✓	✓		✓	ASCII
P		*				#																				✓	FORTRAN
Q	✓	✓	✓	#	✓	#	✓	✓	✓			?	✓	✓	✓			#	✓	✓			✓	✓		✓	Prints file PIP.HLP
R						#																		✓		#	Rename only
S	✓	#	✓			#														✓						✓	ASCII mode
T		#	✓			#													✓							✓	ASCII mode
U						#																				✓	DTA only
V	✓	?	✓	#	✓	#	✓	?	?	✓			✓	✓	✓		#	#	✓	✓			✓			✓	ASCII mode
W		#	✓			#																				✓	ASCII mode
X	✓	✓	✓	*	✓	#	✓	✓	✓	✓		#	✓	✓	✓	✓	✓	✓	✓	✓				✓		✓	ASCII or Binary mode
Y						#																				✓	Binary mode
Z	✓	✓	✓	#	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	#	✓	✓	✓	✓	✓	✓	✓		Output only

LEGEND:

Symbol	Meaning
✓	A permitted combination
?	A permitted but unlikely combination
#	Not permitted
*	Special purpose combination
Blank	Untested or unused combination

SECTION 4

SPECIAL PIP SWITCHES

4.1 SPECIAL PIP FUNCTIONS

This section contains descriptions of optional PIP functions used in magnetic tape, error recovery and card punch operations.

4.2 MAGNETIC TAPE SWITCHES

When magnetic tape is used in a file transfer, PIP can set the tape parity and density parameters and position the tape reels. In PIP commands, magnetic tape switches apply to only one particular magnetic tape unit or file specification.

The optional PIP magnetic tape (MTA) switches are written enclosed in parentheses; the letter M is used as the first character of all optional switches or series of switches (e.g. (Msw) or (Mswlsw2..).

MTA switches must appear within the command file specifications of the particular file to which they refer. Thus, MTA switches refer to a particular device and, except for density and parity selections, to a particular file specification of that device.

4.2.1 Switches for Setting Density and Parity Parameters

The default Monitor density of 8ØØ bits-per-word (bpi) and odd parity are assumed unless either the Monitor SET DENSITY command was given or one of the following switches is included in the PIP command file specifications:

Switch	Meaning
(M8)	8ØØ bpi density (default value)
(M5)	556 bpi density
(M2)	2ØØ bpi density
(ME)	Even parity (odd parity is default)

The following command string causes PIP to transfer a file from MTA1 to MTA2 at 2ØØ bpi, with even parity (and in ASCII line mode)

```
MTA2:(M2E)=MTA1(ME2)<CR>
```

4.2.2 Switches for Positioning Magnetic Tape

The following switches are used in PIP command strings for magnetic tape handling:

Switch	Function Performed
(MA)	Advance tape reel one file.
(MB)	Backspace tape reel one file.
(MD)	Advance tape reel one record.
(MP)	Backspace tape reel one record.
(MW)	Rewind tape reel.
(MT)	Skip to logical End-of-Tape.
(MU)	Rewind and unload.
(MF)	Mark End-of-File.

In PIP MTA commands, the source device need not be given. For example, to rewind MTA1:, type

```
MTA1:(MW)=<CR>
```

If a source device is specified in the command string, information transfer will occur, except when PIP is requested to rewind and unload a magnetic tape.

Several magnetic tape functions may be specified in a single command string. Density or parity, when changed, will appear in the file specification. In the following example, density is set to 2ØØ bpi, parity is even, the tape is to be rewound and the first, third, fourth and fifth files on that reel are to be printed on the line printer.

```
LPT:=MTA1:(M2EW),(MA),,<CR>
```

If multiple backspace, advance file or record movements are needed, the number of movements required is specified by #n (interpreted as decimal). All positioning switches are implemented before any related file transfers are made; thus MTA1:(M#3A)-PTR: will advance MTA1 by three files before transferring a paper tape file to it.

1. If a backspace file (M#nB) request is given, after completion of "n+1" backspace files one advance file request is made unless the tape is at Load Point. In this way the tape is always initially positioned at the beginning of a file. Thus, the command:

   ```
   MTAØ:(MB)=<CR>
   ```

will backspace MTAØ to the start of the previous file.

2. If the Load Point is reached before a backspace file or record request is completed, an error diagnostic will terminate the run and the following error message is printed

   ?LOAD POINT BEFORE END OF BACKSPACE REQUEST?

3. Only one MTA movement per file specification is allowed in a command string. Thus:

   MTAØ:(MT#2B)=...<CR>

   is illegal since it requests two distinct types of MTA movement.

4.2.2.1 BACKSPACE TO START OF CURRENT FILE - The specification of Ø as the value of n in a multiple backspace command (e.g., M#ØB) causes the tape to be backspaced to the start of the current file. The use of M#ØB is not the same as MB, switch MB is equivalent to M#1B.

4.2.2.2 ADVANCE TO END OF CURRENT FILE - The specification of Ø as the value of n in a multiple advance command (e.g., M#ØA) causes the tape to be moved to a point just before the EOF marker of the current file. The use of M#ØA is not the same as MA, switch MA is equivalent to M#1A.

NOTE

The advance and backspace record requests are available as a convenience for the knowledgeable user, and should be approached with caution. Always remember that PIP typically has multiple input and output buffers and the physical position of the tape need not correspond to the physical position of the record currently being processed.

4.3 G-SWITCH, ERROR RECOVERY

If the error recovery switch /G is present in a command string, a specific set of I/O errors will be acknowledged by error messages. The I/O errors affected by the presence or absence of /G are listed in Section 5, paragraph 5.2, item 3 of the error messages, and are flagged by an asterisk (*). Processing will continue after the error message is printed as though no error had occurred. Thus, must I/O errors occurring within a file may be overridden. However, if the same error condition occurs in each buffer of the file, the error

message is repeated for each buffer until either the end of file occurs or the error condition disappears. A disk directory is used as an input file if it is read to be either listed or searched and is obtained as a core image from the Monitor; therefore, it is not subject to the input errors which may be diagnosed by PIP. However, I/O errors can occur for DECtape directories and are diagnosed at the Monitor level when a directory is read or written. This is, typically, on a LOOKUP or RELEAS request. If the G-switch is not used, any I/O error will close the current output file and, after printing a suitable message, terminate the current request to PIP.

4.4 J-SWITCH, CARD PUNCH

The J-switch causes cards to be punched in Ø29 mode. The output device specified by the command string must be the card punch (CDP).

SECTION 5

PIP ERROR REPORTING AND ERROR MESSAGES

5.1 ERROR MESSAGES

This section describes the various types of error conditions and error messages that can occur during PIP operations.

The special treatment of recoverable error messages which prevent the current job being prematurely terminated when running under the Batch Processor is also described.

When an error message terminates a PIP run, both the input and output devices are released. This means that all files, fully or partly created, are available on the destination device.

NOTE

All error messages preceded by a question mark (?) indicate a fatal (non-recoverable) error.

5.2 I/O ERROR MESSAGES

I/O error messages are opened with a description of the relevant device and file; for example,

1. INPUT DEVICE DTA3:FILE FILNAM.EXT...
2. OUTPUT DEVICE DTA3:FILE FLNAM.EXT...
3. DISK DIRECTORY READ...

Device	Message
DTA,DKS,MTA	WRITE (LOCK) ERROR
*CDR	7-9 PUNCH MISSING
*OTHER	BINARY DATA INCOMPLETE
*ALL DEVICES	DEVICE ERROR
*ALL DEVICES	CHECKSUM OR PARITY ERROR
DTA	BLOCK OR BLOCK NUMBER TOO LARGE
*OTHER	INPUT BUFFER OVERFLOW
*MTA	PHYSICAL EOT

*Recoverable error if a G-switch is used, read paragraph 4.3 for a description of /G.

Thus, for the command DTA4:CON.REL=DTA3:CON.REL, if DTA4 is WRITE LOCKed, PIP prints the error message:

?OUTPUT DEVICE DTA4:FILE CON,REL WRITE(LOCK)ERROR

Other messages for devices are:

1. ?DEVICE dev DOES NOT EXIST (DEVCHR request)
2. ?DEVICE dev NOT AVAILABLE (INIT request)

5.3 FILE REFERENCE ERRORS

The following error messages can occur during a LOOKUP, RENAME or ENTER request on disk.

message:? (filename.ext) then one of the following:

(Ø)	FILE WAS NOT FOUND or (Ø) ILLEGAL FILE NAME (used for enter errors only)
(1)	NO DIRECTORY FOR PROJECT-PROGRAMMER NUMBER
(2)	PROTECTION FAILURE
(3)	FILE WAS BEING MODIFIED
(4)	RENAME FILE NAME ALREADY EXISTS
(5)	ILLEGAL SEQUENCE OF UUOS
(6)	BAD UFD OR BAD RIB
(7)	NOT A SAV FILE
(1Ø)	NOT ENOUGH CORE
(11)	DEVICE NOT AVAILABLE
(12)	NO SUCH DEVICE
(13)	NOT TWO RELOC REG. CAPABILITY
(14)	NO ROOM OR QUOTA EXCEEDED
(15)	WRITE LOCK ERROR
(16)	NOT ENOUGH MONITOR TABLE SPACE
(17)	PARTIAL ALLOCATION ONLY
(2Ø)	BLOCK NOT FREE ON ALLOCATION
(21)	CAN'T SUPERSEDE (ENTER) AN EXISTING DIRECTORY
(22)	CAN'T DELETE (RENAME) A NON-EMPTY DIRECTORY
(23)	SFD NOT FOUND
(24)	SEARCH LIST EMPTY
(25)	SFD NESTED TOO DEEPLY
(26)	NO-CREATE ON FOR SPECIFIED SFD PATH

If the error code (V) is greater than 26_8, the error message:

?(V) LOOKUP,ENTER, OR RENAME ERROR

is printed.

Error values are used by the UUO's LOOKUP, ENTER and RENAME. Refer to the DECsystem-10 Monitor Calls manual for complete descriptions of these UUO's.

The following error messages may be given on a reject to an ENTER request on DECtape:

1. The error message printed if there is no room for an entry in a DECtape directory is

   ?DIRECTORY FULL:

2. The error message printed if a zero filename is given for a DECtape output file is

   ?ILLEGAL FILE NAME:

The following message is given if a filename is not found in a directory search of disk or DECtape

?NO FILE NAMED filename.ext

5.4 PIP COMMAND ERRORS

The following error messages are output by PIP on the detection of errors in the user command string:

1. ?PIP COMMAND ERROR

   Some of the possible causes of this type of error are:

   a. an illegal format for a command string,

   b. a nonexistent switch was requested,

   c. a filename other than * or *.* was given for a non-directory (source) device.

2. ?INCORRECT PROJECT-PROGRAMMER NUMBER:

   The project-programmer number must be in the form

   [number,number]

   where $\emptyset$<number<777777_8, a full path specification must be made if SFD's are involved.

3. ?SFD LIST TOO LONG:

   Too many SFD's were listed in the full directory path. A maximum of five levels (not including the UFD) is permitted in a directory path specification.

4. ?ILLEGAL PROTECTION:

   The protection number must be in the form <number>, where: $\emptyset$<=number<=777_8.

5. ?NO BLOCK Ø COPY

   The /U switch was specified, but PIP was not assembled to allow this.

6. ?TOO MANY REQUESTS FOR...(magnetic tape)

   Conflicting density and/or parity requests were given.

5.5 Y-SWITCH ERRORS

The following error messages occur only when the Y-switch is included in the PIP command string:

1. ?DTA to PTP ONLY:

   Only DECtape input and paper tape output are permitted.

2. ?/Y SWITCH NOT AVAILABLE THIS ASSEMBLY:

   The /Y switch was specified, but PIP was not assembled to allow this.

3. FILE filename.ext ILLEGAL EXTENSION:

   The extensions of the filenames given must be .RMT, .RTB or .SAV.

4. Filename.ext ILLEGAL FORMAT:

   The reasons for getting the diagnostic ILLEGAL FORMAT are:

   a. a zero length file was found,

   b. the required job data information was not available,

   c. a block overlapped a previous block (RIM 1Ø),

   d. an EOF was found when data was expected,

   e. a pointer word expected but not found in the source file.

5.6 GENERAL ERROR MESSAGES

The following is a list of the PIP error messages which are not included in any of the preceding categories:

1. ?DISK OR DECTAPE INPUT REQUIRED:

   This message is printed when a non-directory source device is specified for a PIP function which requires a directory-type source device.

2. ?filename.ext ILLEGAL FILE NAME:

   This message is output if an attempt is made to ENTER without giving a filename.

3. Errors found during /X, /Z, /D, and /R operations result in error messages which pertain to the specific error found. Error messages for these operations are printed only if no other fatal error occurs before the command string is processed. If another error does occur, its diagnostic takes precedence over the diagnostics for the above switch functions.

4. ?4K NEEDED:

   4K not currently available but is needed (for non-reentrant disk system).

5. ?DECTAPE I/O ONLY:

   The I/O device for a block Ø copy (/U switch) must be a DECtape.

6. ?TERMINATE /X.MAX. OF 999 FILES PROCESSED:

   PIP, during a /X copy function from a non-directory device, has processed 999 files. This is the maximum number of files which such a /X request can handle.

7. ?TOO MANY INPUT DEVICES:

   This error is for the /D and /DX functions; only one input device is allowed when these switches are used. If more than one device is specified in a /D command and the first device given is DSK, the disk files are deleted when this diagnostic is given.

8. ?NO FILE NAMED PIP.HLP:

   The data file requested by a PIP Q-switch is not available on the system device.

9. ?LINE TOO LONG:

   During an ASCII mode file transfer a line containing more than 18Ø characters was detected. This occurs only when switches entailing line processing are given (i.e., /A or /S).

10. ?LOAD POINT BEFORE END OF BACKSPACE REQUEST:

    This diagnostic occurs only if either the MTA (M#nB) or (M#nP) switch is used. If the Load Point is sensed before the "n" backspace files or records function is completed, an error is assumed to have been made by the user.

5.7 TMPCOR (DEVICE TMP) ERROR MESSAGES

If the temporary storage facilities provided by the UUO TMPCOR are used or are attempted to be used during PIP operations, the following error messages can occur:

1. ?TMPCOR NOT AVAILABLE:
2. ?NOT ENOUGH ROOM IN TMPCOR:
3. ?COMMAND NOT YET SUPPORTED FOR TMPCOR:
4. nn TMPCOR WORDS FREE

   Number of word locations free in the TMPCOR storage area.

Refer to the DECsystem-1Ø Monitor Calls manual for a description of the UUO TMPCOR.

APPENDIX A

STANDARD FILENAME EXTENSIONS

Table A-1
Filename Extensions

Filename Extension	Type of File	Meaning
AID	Source	Source file in AID language.
ALG	Source	Source file in ALGOL language.
ALP	ASCII	Printer forms alignment.
BAC	Object	Output from the BASIC Compiler.
BAK	Source	Backup file from TECO or LINED.
BAS	Source	Source file in BASIC language.
BIN	Object	Binary file.
BLB	ASCII	Blurb file.
BLI	Source	Source file in BLISS language.
BNC	ASCII	BINCOM output.
BUG	Object	Saved to show a program error.
CAL	Object	CAL data and program files.
CBL	Source	Source file in COBOL language.
CCL	ASCII	Alternate convention for command file (@ construction for programs other than COMPIL).
CCO	ASCII	Listing of modifications to non-resident software.
CKP	Binary	Checkpoint core image file created by COBOL operating system.
CHN	Object	CHAIN file.
CMD	ASCII	Command file for indirect commands (@ construction for COMPIL).
CMP	ASCII	Complaint file by GRIPE.
COR	ASCII	Correction file for SOUP.
CRF	ASCII	CREF (cross-reference) input file.

Table A-1

Filename Extensions (Cont'd)

Filename Extension	Type of File	Meaning
CTL	ASCII	MP batch control file.
DAE	Binary	Default output for DAEMON-taken core dump.
DAT	ASCII, Binary	Data (FORTRAN) file.
DCR	Binary	Core image save (DCORE).
DDT	ASCII	Input file to FILDDT.
DIR	ASCII	Directory from FILE command or DIRECT program.
DMP	PDP-6	PDP-6 format for a file created by a SAVE command.
DOC	ASCII	Listing of modifications to the most recent version of the software.
ERR	ASCII	Error message file.
F4	Source	Source file in FORTRAN language.
FLO	ASCII	English language flowchart.
FRM	ASCII	Form.
FUD	ASCII	FUDGE2 listing output.
HGH	Object	Nonsharable high segment of a two-segment program.
HLP	ASCII	Help files containing switch explanations, etc.
INI	ASCII, Binary	Initialization file.
LOG	ASCII	MP batch log file.
LOW	Object	Low segment of a two-segment program.
LSD	ASCII	Default output for DUMP program.
LSQ	ASCII	Queue listing.
LST	ASCII	Listing data.
MAC	Source	Source file in MACRO language.
MAN	ASCII	Manual (documentation) file.

Table A-1

Filename Extensions (Cont'd)

Filename Extensions	Type of File	Meaning
MAP	ASCII	Loader map file.
MEM	ASCII	Memorandum file.
MSB	Object	Music compiler binary output.
MUS	Source	Music compiler input.
OLD	Source	Backup source program.
OPR	ASCII	Installation and assembly instructions.
PAL	Source	Source file in PAL 1Ø (PDP-8 assembler).
PBT	ASCII	P-batch control file.
PLG	ASCII	P-batch log file.
QUC	Binary	Queue change request file.
QUD	ASCII, Binary	Queued data file.
QUE	Binary	Queue request file.
QUF	Binary	Master queue and request file.
REL	Object	Relocatable binary file.
RIM	Object	RIM loader file.
RMT	Object	Read-In mode (RIM) format file (PIP).
RNC	ASCII	RUNOFF input for producing a .CCO file.
RND	ASCII	RUNOFF input for producing a .DOC file.
RNO	ASCII	Programming specifications in RUNOFF input.
RNP	ASCII	RUNOFF input for producing a .OPR file.
RSP	ASCII	Script response time log file.
RTB	Object	Read-In mode (RIM1ØB) format file (PIP).
SAV	Object	Low segment from a one-segment program.
SCP	ASCII	SCRIPT control file.
SFD	Binary	Sub-file directory (restricted usage).
SHR	Object	Sharable high segment file of a two-segment program.

Table A-1

Filename Extensions (Cont'd)

Filename Extension	Type of File	Meaning
SNO	Source	Source file in SNOBOL language.
SNP	ASCII	Snapshot of disk by DSKLST.
SRC	ASCII	SRCCOM output.
SVE	Object	.SAVed file from a single user Monitor.
SYS	Binary	Special System files.
TEC	ASCII	TECO macro.
TMP	ASCII, Binary	Temporary files.
TXT	ASCII	Text file.
UFD	Binary	User file directory (restricted usage).
UPD	ASCII	Updates flagged in margin (SRCCOM).
WCH	ASCII	SCRIPT Monitor (WATCH) file.
XPN	Object	Expanded save file (FILEX).