In the example below, the command 30C moves the CP from the beginning of one line to the middle of the next.

**Initial location of CP**
**^Emily Dickinson said, <cr><lf>**
**"I find ecstasy in living - <cr><lf>**

**After execution**
**Emily Dickinson said, <cr><lf>**
**"I fin^d ecstasy in living - <cr><lf>**

## The L (Line) Command

The L command moves the CP the specified number of lines. After executing an L command, the CP is always located at the beginning of a line. The L command takes the following forms:

*n*L, −*n*L

where n is the number of lines the CP is to be moved. A positive number moves the CP towards the end of the edit buffer, and a negative number moves it back toward the beginning of the buffer. The command 2L moves the CP two lines forward through the memory buffer and positions it at the beginning of the line.

**After execution**
**"I find ecstasy in living - <cr><lf>**
**the mere sense of living <cr><lf>**
**^is joy enough."<cr><lf>**

The command -L moves the CP to the beginning of the previous line, even if it is originally located at a character in the middle of the line. However, the CP can be moved to the beginning of the current line with the 0 command.

## The n (Number) Command

The *n* command moves the CP and displays the line to which the CP is moved. This command takes the following forms:

*n*, −*n*

where n is the number of lines the CP is to be moved. In response to this command, ED moves the CP forward or backward the number of lines specified, then displays only the line containing the CP after movement has been completed. For example, the command -2 moves the CP back two lines.

```
After execution
Emily Dickinson said, <cr><lf>
^"I find ecstasy in living - <cr><lf>
the mere sense of living <cr><lf>
is joy enough."<cr><lf>
```

A further abbreviation of this command is to enter no number at all; in other words, to press RETURN without typing any command. In this case, ED assumes an n command of 1, moves the CP down to the next line, and displays that line as follows.

```
After execution
Emily Dickinson said, <cr><lf>
"^I find ecstasy in living - <cr><lf>
the mere sense of living <cr><lf>
```

Also, the CP can be moved back one line by typing a minus sign (-) without entering any number.

### 2.2 Displaying edit buffer contents

ED does not display the contents of the edit buffer until you specify which part of the text you want to see. The T command displays text without moving the CP.

The T (Type) Command

The T command displays a specified number of lines of text on the screen, starting from the position of the CP. The T command takes the following forms:

*n*T, −*n*T

where n specifies the number of lines to be displayed. If a negative number is entered, ED displays the *n* lines preceding the CP. If a positive number is entered, it displays the *n* lines following the CP. If no number is specified, ED displays text from the CP to the end of the line.

The CP remains in its original position no matter how many lines are displayed. For example, it the character pointer is at the beginning of the edit buffer and you instruct ED to display four lines (4T), four lines are displayed on the screen but the CP stays at the beginning of line 1.

**After execution**
**Emily Dickinson said, <cr><lf>**
**^"I find ecstasy in living - <cr><lf>**
**the mere sense of living <cr><lf>**
**is joy enough."<cr><lf>**

If the CP is located between two characters in the middle of a line, a T command with no number specified displays only the characters between the CP and the end of the line.

When text is displayed with the T command, you can stop the display by pressing CTRL - S ; when you want display to resume, press any key. If you type a large value for *n* and want to discontinue display, press CTRL - C .

*NOTE:*
*If you want to echo display output to a printer, press* CTRL - P .

## 2.3 Deleting characters
The D (Delete) Command

The D command deletes a specified number of characters. This command takes the following forms:

***n*D, −*n*D**

where n is the number of characters to be deleted. If no number is specified, ED deletes the character to the right of the CP. A positive number deletes multiple characters to the right of the CP (toward the bottom of the file). A negative number deletes characters to the left of the CP (toward the top of the file). If the character pointer is positioned in the edit buffer as follows

**Emily Dickinson said, <cr><lf>**
**"I find ecstasy in living - <cr><lf>**
**the mere sense of living <cr><lf>**
**is joy ^enough."<cr><lf>**

the command 6D deletes six characters after the CP so that the contents of the edit buffer change like this:

**Emily Dickinson said, <cr><lf>**
**"I find ecstasy in living - <cr><lf>**
**the mere sense of living <cr><lf>**
**is joy ^."<cr><lf>**

You can also use the D command to join two lines by deleting the <cr> and <lf> the <cr> and <lf> between them. Remember that the <cr> and <lf> are two characters.

## The K (Kill) Command

The K command kills or deletes whole lines from the edit buffer. This command takes the following forms:

*n***K**, −*n***K**

where *n* is the number of lines to be deleted. A positive number kills lines following the CP, and a negative number kills lines preceding the CP. When no number is specified, ED kills the current line. If the character pointer is at the beginning of the second line,

**Emily Dickinson said, <cr><lf>**
**^"I find ecstasy in living - <cr><lf>**
**the mere sense of living**
**<cr><lf>**
**is joy enough."<cr><lf>**

then the command -K deletes the previous line so that the buffer contents change as follows.

**^"I find ecstasy in living - <cr> <lf>**
**the mere sense of living <cr> <lf>**
**is joy enough." <cr> <lf>**

If the CP is in the middle of a line, a K command kills only the characters from the CP to the end of the line and concatenates the characters preceding the CP with the next line. A –K command deletes all characters from the CP to the beginning of the previous line. A 0K command deletes characters from the beginning of the line to the CP.

You can use the # character to delete all text from the CP to the beginning or end of the buffer. Be careful when using #K, since you cannot recover lines after they have been deleted.

## 2.4 Inserting characters into the edit buffer

The I (Insert) Command

To insert characters into the edit buffer from the keyboard, use the I command. The I command takes the following forms:

**I** or **i Istring^Z** or **istring^Z**

If the command is typed using an upper case I, ED automatically converts the string or characters typed to uppercase form.

The I (or i) command by itself switches ED to the insert mode. In this mode, all keystrokes are added directly to the edit buffer. ED enters characters typed in the line and does not start a new line until you press the RETURN key.

**Example:**

```
D>ED E:QUOTE.TEX
NEW FILE

  :  *i
 1:    Emily Dickinson said,
 2:    "I find ecstasy in living -
 3:    the mere sense of living
 4:    is joy enough."
 5:    ^Z
  :*
```

*NOTE:*
*To exit from the insert mode, you must press* CTRL - Z*. When the ED prompt (*) appears on the screen, ED is in the command mode, not in the insert mode.*

ED provides two ways of converting characters input to uppercase without affecting numbers. The first is to enter the insert command letter in uppercase (as I). This causes all lowercase letters typed (either in the insert mode or in a string following I) to be converted to uppercase. If you enter the insert command character in lowercase, all letters are inserted exactly as typed.

The second method is to enter a U command before inserting text. When this method is used, uppercase conversion remains in effect until you enter a –U command.

The Istring^Z (Insert String) Command

The second form of the I command does not switch ED into the insert mode. Instead, it inserts the specified character string into the edit buffer and returns immediately to the ED prompt. You can use ED's line-editing control characters to edit the string.

*NOTE:*
*Line-editing control characters which can be used in the command mode are listed in section 4, "ED Control Characters."*

To insert a string, first use one of the character pointer movement commands to move the CP to the place where you want to insert the string. For example, if you want to insert a string at the beginning of the first line, use a B command to move the CP to the beginning of the buffer. When the CP is positioned correctly, enter an insert string as follows:

**iIn 1870, ^Z**

This inserts the string "In 1870", at the beginning of the first line, then redisplays the ED prompt. After executing this command, the CP will be located immediately after the string inserted as shown below.

**In 1870, Emily Dickinson said,<cr><lf>**

## 2.5 Replacing characters

The S (Substitute) Command

The S command searches the edit buffer for the specified string, then the string found with a new string. As with the I command, strings specified are automatically converted to uppercase if the command is typed as an uppercase S. The S command takes the following form:

***n*****Ssearch string^Znew string**
or
***n*****ssearch string^Znew string**

where *n* is the number of substitutions to be made. If no number is specified, ED searches for the next occurrence of the search string in the buffer, then replaces it with the replacement string. For example, the command

**sEmily Dickinson^ZThe poet**

searches for the first occurrence of "Emily Dickinson" and replaces it with "The poet". After substitution has been completed, the CP is located at the end of the new string as follows:

**The poet ^said, <cr><lf>**

If uppercase conversion is enabled by a capital S, ED looks for a capitalized search string and inserts a capitalized new string. Note that if you combine this command with other commands, you must terminate the new string with CTRL - Z (^Z).

## 3. Combining ED Commands

You can save keystrokes and reduce editing time by combining editing commands and display commands. Any number of ED commands can be typed on the same line; ED executes the commands only after you press the RETURN key. Use the line-editing control characters and editing keys to manipulate ED command strings.

When you combine several commands on a line, ED executes them in the order in which they are typed (from left to right on the command line). However, there are four restrictions to observe when combining ED commands.

- The total length of the command line must not exceed 128 characters.
- If the command line contains a character string, the length of the command line must not exceed 100 characters.
- Commands which terminate the editing session must not be included on the command line with other commands.
- Commands which require character strings or file names (such as the I, J, R, S, and X commands) must either be the last command on the line or must be terminated with CTRL - Z (^Z), even if no character string or file name is given.

While the examples in the previous section show the edit buffer and the position of the character pointer, the examples in this section show how the screen looks during an editing session. Remember that the character pointer is imaginary, but that you must picture its location because ED's commands display and edit text in relation to the character pointer.

### 3.1 Moving the character pointer

To move the CP to the end of a line without calculating the number of characters, combine an L command with a C command, L-2C. This command string accounts for the <cr><lf> sequence at the end of the line.

Change the C command in this command string to move the CP further to the left. You can use this command string if you must make a change at the end of the line and you don't want to calculate the number of characters before the change, as in the following example.

```
     1:  *T
     1:    Emily dickinson said,
     1:  *L-7CT
said,
     1:  *
```

## Displaying Text

A T command types from the CP to the end of the line. To see the entire line, you can combine an L command and a T command. Type 0lt to move the CP from the middle to the beginning of the line and then display the entire line. In the example below, the CP is in the middle of the line. 0L moves the CP to the beginning of the line. T types from the CP to the end of the line, allowing you to see the entire line.

```
     3:  *T
     sense of living
     3:  *0LT
     3:    the mere sense of living
     3:  *
```

The command 0TT displays the entire line without moving the CP.

To verify that an ED command moves the CP correctly, combine the command with the T command to display the line. The following examples combine the C and B commands with a T command.

```
2:  * 8CT
ecstasy in living -
2:  *

4:  * B#T
1:     Emily Dickinson said,
2:     "I find ecstasy in living -
3:     the mere sense of living
4:     is joy enough."
1:  *
```

## 3.2 Editing

To edit text and verify corrections quickly, combine the edit commands with other ED commands that move the CP and display text. Command strings like the one below move the CP, delete specified characters, and verify changes quickly.

```
1:  * 15C5D0LT
1:     Emily Dickinson,
1:  *
```

To quickly delete entire lines and verify changes, combine the K command with other ED commands as follows:

```
1:  * 2L2KB#T
1:     Emily Dickinson said,
2:     "I find ecstasy in living -
1:  *
```

The abbreviated form of the I (Insert) command can be used to make simple changes in text. To make and verify changes, combine the I command sting with the C command and the 0LT command string as follows. Remember that the insert string must end with CTRL - Z (^Z).

```
1:  * 20Ci to a friend^Z0LT
1:     Emily Dickinson said to a friend,
1:  *
```

# 4. Advanced ED Commands

The basic editing commands discussed previously allow you to use ED for all your editing. However, the following ED commands enhance ED's usefulness.

## 4.1 Moving the CP and displaying text

The P (Page) Command

Although you can display any amount of text on the screen with the T command, it is sometimes more convenient to page through the buffer, viewing whole screens of data and moving the CP to the top of each new screen at the same time. To do this, use ED's P command. The P command takes the following forms:

***n*P**, **−*n*P**

where *n* is the number of pages to be displayed. If you do not specify *n*, ED types the 23 lines following the CP, then moves the CP forward 23 lines. This leaves the CP pointing to the first character on the screen.

To display the current page without moving the CP, enter 0P. The special character 0 prevents the CP from being moved. If you specify a negative number for *n*5, P pages backwards towards the top of the file.

The n: (Line Number) Command

When line numbers are being displayed, ED accepts a line number as a command to specify a destination for the CP. The line number command takes the following form:

***n:***

where *n* is the number of the destination line (the line to which the CP is to be moved). This command places the CP at the beginning of the specified line. For example, the command 4: moves the CP to the beginning of the fourth line.

Remember that ED dynamically renumbers text lines in the buffer each time a line is added or deleted. Therefore, the number of the destination line you have in mind can change during editing.

The :n (Through Line Number) Command

The inverse of the line number command specifies that a command should be executed through a certain line number. You can use this command with only three ED commands: the K (Kill) command, the L (Line) command, and the T (Type) command. The :n command takes the following form:

**:*n*command**

where *n* is the line number through which the command is to be executed. The *:n* part of the command does not move the CP, the command that follows it might.

You can combine *n:* with *:n* to specify a range of lines through which a command should be executed. For example, the command 2::4T displays the second, third, and fourth lines as shown below.

```
1:   * 2::4T
2:     "I find ecstasy in living -
3:     the mere sense of living
4:     is joy enough."
2:       *
```

## 4.2 Finding and replacing character strings

ED supports a find command (F) that searches through the memory buffer and places the CP after the word or phrase you want to find. The N command allows ED to search through the entire source file, instead of just the edit buffer. The J command searches for and juxtaposes (links) character strings.

The F (Find) Command

The simplest find function is carried out by the F command. This command takes the following form:

***n*Fstring**

where *n* is the occurrence of the string to be found. Any number you enter must be positive because ED can only search from the CP to the bottom of the file. The following example finds the second occurrence of the word "living".

```
1:  * 2fliving
3:  *
```

The character pointer moves to the beginning of the third line where the second occurrence of the word "living" is located. To display the line, combine the find command with a type command. Note that if you follow an F command with another ED command on the same line, you must terminate the string in the F command with a CTRL - Z as follows:

```
1:  * 2fliving^Z0lt
3:  * the mere sense of living
```

It makes a difference whether you enter the F command in uppercase or lowercase. If you enter F, ED internally translates the argument string to uppercase. If you specify f, ED looks for an exact match. For example, Fcp/m3 searches for CP/M 3, but fcp/m 3 searches for cp/m 3, and will not find CP/M 3.

If ED does not find a match for the string in the edit buffer, it displays the message

**BREAK "#" AT**

where the symbol # indicates that the search failed during execution of an F command.

The N Command

The N command extends the search function beyond the memory buffer to include the source file. If the search is successful, it leaves the CP pointing to the first character after the search string. The N command takes the form:

**nNstring**

where *n* is the occurrence of the string to be found. If no number is entered, ED looks for the next occurrence of the string in the file. The case of the N command has the same effect on an N command as it does on an F command. Note that if you follow an N command with another ED command, you must terminate the string with CTRL - Z .

When an N command is executed, ED searches the edit buffer for the specified string, but does not issue an error message if the string is not found. Instead, ED automatically writes the searched data from the buffer into the new file, then performs a 0A command to fill the buffer with unsearched data from the source file. ED continues to search the buffer, to write out data, and to append new data into the buffer until it either finds the string or reaches the end of the source file. If ED reaches the end of the source file, ED issues the following message:

**BREAK "#" AT**

Since ED writes data searched to the temporary file before looking for more data in the source file, the contents of the buffer are usually written to the temporary file before the end of the source file is reached and the error message is issued.

*NOTE:*
*After the source file has been exhausted and the edit buffer is emptied, you must use the H command to continue the edit session.*

The J (Juxtapose) Command

Three strings are specified in the J command: a search string, an insert string, and a delete-to string. The J command inserts the insert string after the search string, then deletes all characters between the end of the insert string and the beginning of the delete-to string. This juxtaposes the insert string with characters beginning with the delete-to string. The J command takes the form:

**nJsearch string^Zinsert string^Zdelete-to string**

where *n* is the occurrence of the search string. If no number is specified, ED searches for the next occurrence of the search string in the edit buffer. In the following example, ED searches for the word "Dickinsson", inserts the phrase "told a friend" after it, then deletes everything up to the comma.

```
1:  * #T
1:    Emily Dickinson said,
2:    "I find ecstasy in living -
3:    the mere sense of living
4:    is joy enough."
1:  * jDickinson^Z told a friend^Z,
1:  *0lt
1:    Emily Dickinson told a friend,
1:  *
```

If you combine this command with other commands, you must terminate the delete-to string with CTRL - Z , as in the following example. If an uppercase J is specified as the command letter, ED looks for uppercase search and delete-to strings and inserts an uppercase insert string.

The J command is especially useful when revising comments in assembly language source code, as shown in the following example:

```
236:   SORT   LXI   H,  SW    ; ADDRESS TOGGLE SWITCH
236: * j;^ZADDRESS SWITCH TOGGLE^Z^L^Z0LT
236:   SORT   LXI   H,  SW    ; ADDRESS SWITCH TOGGLE
236: *
```

In this example, ED searches for the first semicolon and inserts ADDRESS SWITCH TOGGLE after the mark, then deletes to the <cr><lf> sequence, which is represented by CTRL - L . In any search string you can use CTRL - L to represent <cr> <lf> when the phrase that you want extends across a line break. You can also use CTRL - I in a search string to represent a tab.

*NOTE:*
*If long strings make your command longer than your screen line length, enter* CTRL - E *to cause a physical carriage return on the screen.* CTRL - E *returns the cursor to the left edge of the screen, but does not send the command to ED. Remember that no ED command line containing strings can exceed 100 characters. When you finish your command, press the* RETURN *key to send the command to ED.*

## The M (Macro) Command

An ED macro command, M, can increase the usefulness of a string of commands. The M command allows you to group ED commands together for repeated execution. The M command takes the following form:

***n*Mcommand string**

where *n* is the number of times the command string is to be executed. A negative number is not a valid argument for an M command. If no number is specified, the special character "#" is assumed and ED executes the command string until it reaches the end of data in the buffer or the end of the source file, depending on the commands specified in the string. In the following example, ED executes the four commands repetitively until it reaches the end of the edit buffer:

```
      1:   *  mfliving^Z-GdiLiving^Z0lt
      2:      "I find ecstasy in Living -
      3:      the mere sense of Living

   BREAK "#" AT ^X

      3:   *
```

The terminator for an M command is a carriage return; therefore, an M command must be the last command on the line. Also, all character strings that appear in the macro must be terminated by CTRL - Z . If a character string ends the combined command string, it must be terminated by CTRL - Z , then followed by <cr> to end the M command.

The execution of a macro command always ends in a BREAK"#" message, even when you have limited the number of times the macro is to be performed and ED does not reach the end of the buffer or source file. Usually, the command letter displayed in the message is one of the commands from the string, and not M.

To abort a macro command, press CTRL - C or the STOP key.

The Z (Sleep) Command

Use the Z command to make the editor pause between operations. The pauses give you a chance to review what you have done. The Z command takes the following form:

```
nZ
```

where *n* is the number of seconds to wait before proceeding to the next instruction.

Usually, the Z command has no real effect unless you use it with a macro command. The following example shown you how you can use the Z command to cause a brief pause each time ED finds the word "living" in a file.

```
1: * mfliving^Z0tt10z
```

**4.3 Moving text blocks**

To move a group of lines from one area of you data to another, use an X command to write the text block into a temporary LIB file, then use a K command to remove these lines from their original location. Finally, use an R command to read the block into its new location from the LIB file.

The X (Transfer) Command

The X command takes the following forms:

***n*X**
***n*Xfilename^Z**

where *n* is the number of lines from the CP towards the bottom of the buffer that are to be transferred to a file. There, n must always be a positive number. The nX command with no file specified creates a temporary file named X$$$$$$.LIB. This file is erased when you terminate the edit session. The nX command with a file specified creates a file of the specified name. If no filetype is specified, LIB is assumed. This file is saved when you terminate the edit session. If the X command is not the last command on the line, the command must be terminated by CTRL - Z . In the following example, just one line is transferred to the temporary file.

```
1:  * X
1:  * t
1:  * Emily Dickinson said,
1:  * Kt
1:  * "I find ecstasy in living -
1:  *
```

If no library file is specified, ED looks for a file named X$$$$$$.LIB. If the file does not exist, ED creates it. If a previous X command has already created the library file, ED appends the specified lines to the end of that file.

Use the special character 0 as the n argument in an X command to delete any file from within ED.

The R (Read) Command

The X command transfers the next n lines from the current line to a library file. The R command can retrieve the transferred lines. The R command takes the following forms:

**R**
**Rfilename**

If no file name is specified, X$$$$$$$ is assumed. If no file type is specified, LIB is assumed. R inserts the library file in front of the CP; therefore, after the file is added to the memory buffer, the CP points to the same character is did before the read, although the character is on a new line number. If you combine an R command with other commands, you must separate the file name from subsequent command letters with CTRL-Z, as is shown in the following example. In this example, ED types the entire file to verify the read.

```
1:  * 41
 :  * R^ZB#T
1:    "I find ecstasy in living -
2:    the mere sense of living
3:    is joy enough.'
4:    Emily Dickinson said,
1:  *
```

**4.4 Saving or abandoning changes: ED exit**
You can save or abandon editing changes with the following three commands.

The H (Head of File) Command

An H command saves the contents of the edit buffer without ending the ED session, then returns to the head of the file. This lets you save current changes and keep editing the file without exiting ED. The H command takes the following form:

```
H
```

followed by a carriage return.

To execute an H command, ED first finalizes the new file, transferring all lines remaining in the buffer and the source file to the new file. Then ED closes the new file, erases any BAK file that has the same file name as the original source file, and renames the original source file to "filename.BAK". ED then renames the new file (which has had the file type $$$) with the original file name. Finally, ED opens the newly renamed file as the new source file for a new edit and opens a new $$$ file. When ED returns the * prompt, the CP is at the beginning of an empty edit buffer.

If you want to send the edited material to a file other than the original file, use the following command.

**A>ED filespec differentfilespec**

If you then restart the edit with the H command, ED renames file "differentfilename.$$$" to "differentfilename.BAK" and creates a new file "differentfilespec" when you finish editing.

The O (Original) Command

An O command abandons changes made since the beginning of the edit and allows you to return to the original source file and resume editing without ending the ED session. The O command takes the form:

**O**

followed by a carriage return. When you enter an O command, ED confirms that you want to abandon your changes by asking:

**O-(Y/N)?**

You must respond with either a Y or an N. When you enter $\boxed{Y}$, ED erases the temporary file and the contents of the edit buffer. When the * prompt returns, the character pointer is pointing to the beginning of an empty edit buffer, just as it is when you start ED.

The Q (Quit) Command

A Q command abandons changes made since the beginning of the ED session and exits ED. The Q command takes the form:

**Q**

followed by a carriage return.

When you enter a Q command, ED verifies that you want to abandon the changes by asking:

**Q-(Y/N)?**

You must respond with either a Y or an N. When you enter Y, ED erases the temporary file, closes the source file, and returns control to CP/M or the menu screen.

*NOTE:*
*You can press the* STOP *key or* CTRL - C *to return control immediately to CP/M. This does not give ED a chance to close the source or new files, but it prevents ED from deleting any temporary files.*

## 1. ED Control Characters

The control characters listed in the table below can be used with ED to control ED operation and edit the ED command line.

For example, you can exit from the insert mode by pressing the CTRL and Z keys together. However, in the insert mode the only control characters which can be used are CTRL - H, CTRL - U, CTRL - X, and DEL.

| | |
|---|---|
| CTRL - C | Warm boots the system. |
| CTRL - E | Outputs a carriage return (<cr>) and line feed (<lf>) to the screen, but does not send the command line to ED. Moves the cursor to the beginning of the next line without erasing previous input. |
| CTRL - H | Deletes the last character on the line which is currently being typed (same as the BS key). |
| CTRL - J | Sends the command line to ED and moves the cursor to the beginning of the current line. Has the same effect as the RETURN key or CTRL - M. |
| CTRL - L | Places a logical <cr><lf> sequence in a search or replacement string. |
| CTRL - R | Places a # sign at the current cursor position, moves the cursor to the next line, and redisplays any effective characters which have been typed on the ED command line since RETURN was last pressed. |

|  |  |
|---|---|
| CTRL - U | Deletes the entire line which is currently being typed and places the cursor at the beginning of that line. The deleted line is not displayed on the screen. |
| CTRL - Z | Exits from the ED insert mode to the ED command mode or terminates a string specified following the S or F command. |
| BS | Deletes the last character typed. |

## 2. ED Error Messages

Error messages displayed by ED are summarized in Appendix B.

When the edit buffer becomes full, a message such as the following is displayed.

**BREAK ">" AT A**

The ">" symbol indicates that the buffer is full, and that no further characters can be added. To recover from this error, use 0W command to write out half of the buffer to the temporary file, or use an O or H command and re-edit the file.

# FILINK

**Purpose** The FILINK program transfers files between EPSON PX-4 and another computer (such as another PX-4, EPSON QX-10, etc.) through the RS-232C communication line.

**Format** **FILINK**

**Explanation** Software which supports the FILINK communication protocol must be executed on the other computer to transfer files. Configuration which support the protocol are as follows.

| Computer | Software |
|---|---|
| PX-8 | FILINK.COM<br>T and C commands of WS.COM<br>/Join, Send and /Join, Receive commands of SC.COM |
| PX-4 | FILINK.COM |
| QX-10 | FILINK.COM |

EPSON supplies optional cables which can be used to connect PX-4 to other EPSON computers and peripherals as follows.

| Other device | Cable |
|---|---|
| PX-8 | #726 |
| QX-10 | #725 |
| CX-20/21 | #724 |
| PX-4 | #726 |

**Using FILINK**

The FILINK program is started by keying in the drive name and FILINK as shown below, or by selecting "FILINK.COM" in the MENU screen.

```
A>D:FILINK  RETURN
```

The following message is then displayed.

```
FILINK © Copyright 1984 by EPSON V1.0
A file transfer via RS-232C port.

The RS-232C status is:
rate=4800  bits=8  stop=2  parity=NONE

Use CONFIG.COM program to change
the RS-232C status.
```

The current RS-232C settings are displayed in the fifth line of the initial screen. However, these settings cannot be changed from the FILINK program. Therefore, use the COFIG program to establish appropriate RS-232C settings before executing FILINK.
Press any key; the following message is then displayed.

```
Press ESC to restart, STOP to exit.
from FILINK or CTRL/STOP to abort
Send or Receive (S/R)
```

If files are to be sent, enter S ; if files are to be received, enter R .

If Send is selected, FILINK asks what file is to be sent. Respond by entering the drive name (if the file is in a drive other than the current drive), file name, and extension. Ambiguous file names can be specified using wildcard characters (* and ?). If the other computer is ready, FILINK starts sending the specified file when the RETURN key is pressed. An example is shown below.

```
Send or Receive (S/R) ? s
Enter file name   *.com   RETURN

DATA1  .COM
     Sending .........
DATA2  .COM
     Sending .............
```

```
DATA3   .COM
    Sendig ....
DATA4   .COM
    Sending ..................
DATA5   .COM
    Sending ......................

Done
eXit or Continue (X/C) ?
```

The name of each file is displayed as it is sent. Periods displayed following the file names each indicate output of 128 bytes of data (these periods are also displayed following file names when files are received).

The message "eXit or Continue (X/C) ?" is displayed after file output (or input) has been completed to ask whether FILINK operation is to be continued or ended. If operation is to be ended, input $\boxed{x}$ to return to the system. If it is to be continued, input $\boxed{c}$ ; operation then resumes with the "Send or Receive (S/R) ?" prompt.

On the receiving side, the screen appears as shown in the example below when ambiguous file names are used to receive files.

```
Send or Receive (S/R) ? r
Enter file name   *.COM   [RETURN]
DATA1   .COM →  DATA1    .ABC
    Receiving .........
DATA2   .COM →  DATA2    .ABC
    Receiving .............
DATA3   .COM →  DATA3    .ABC
    Receiving ....
DATA1   .COM →  DATA1    .ABC
    Receiving ..................
DATA4   .COM →  DATA4    .ABC
    Receiving .....................

Done
eXit or Continue (X/C) ?
```

When files are received without specifying any file name, they are stored on the disk under the file name which is received from the sending side. An example follows.

```
Send or Receive (S/R) r
Enter file name

DATA1  .COM →  DATA1   .COM
    Receiving ..........
DATA2  .COM →  DATA2   .COM
    Receiving .............
DATA3  .COM →  DATA3   .COM
    Receiving ....
DATA1  .COM →  DATA1   .COM
    Receiving ....................
DATA4  .COM →  DATA4   .COM
    Receiving .....................

Done
eXit or Continue (X/C) ?
```

If a file with the same name is already present on the disk (on the receiving side), FILINK displays the following message to ask whether the existing file is to be overwritten.

```
Overwrite (Y/N)?
```

If Y is input in response to this message, the existing file is overwritten with the file which is received; if N is input, a message prompting for input of a different file name is displayed. This file name is only effective for one file, regardless of whether it is specified using wildcard characters. This is illustrated in the exmaple below.

```
Send or Receive (S/R) ? r
Enter file name

FILE2  .DAT →  FILE2   .DAT
Overwrite (Y/N) ? Y
    Receiving ............................
FILE3  .DAT →  FILE3   .DAT
Overwrite (Y/N) ? N
Enter file name     ????6.DAT
```

```
FILE3   .DAT  →  FILE6    .DAT
     Receiving ....................
```

## Functions of Special Keys

The following keys performs special functions.

**(1) ESC**
Pressing the ESC key while the program is waiting to receive files starts program execution over from its beginning.

**(2) STOP**
Pressing the STOP key while the program is waiting to receive files interrupts processing and terminates the program.

**(3) CTRL/STOP**
Pressing the CTRL and STOP keys together while the program is waiting to receive files or is sending files interrupts processing and aborts program execution.

## FILINK Error Messages

When an error occurs during FILINK operation or a condition exists which requires a user decision, it is detected by the program and a message is displayed. Messages displayed for each error/condition and procedures for handling them are as follows.

**(1) File not found**
**Cause:** The specified output file does not exist.
**Handling:** Respecify using the correct file name.

**(2) No file name specified**
**Cause:** An attempt was mde to send without specifying a file name.
**Handling:** Specify the file name.

**(3) Bad file descriptor**
**Cause:** The file name specified does not conform to the rules for specifying file descriptors.
**Handling:** Correctly specify the file name.

**(4) Drive select error**
**Cause:** A drive name other than A: to K: was specified.
**Handling:** Specify a drive name from A: to K:

**(5) Overwrite (Y/N) ?**
**Cause:** This message is issued on the receiving side when a file with the specified name already exists.
**Handling:** Type in Y or N. If Y is input, the existing file is overwritten; if Y is input, FILINK prompt for specification of another file name.

**(6) Communication error**
**Press ESC to restart, STOP to exit from FILINK.**
**Cause:** A communication error (framing or overflow error) occurred, or the same communication parameters (word length, parity, baud rate, etc.) are not set on the sending and receiving computers.
**Handling:** When this message appears, keys other than the following are ignored and the file being transferred must be resent from the beginning.

| | |
|---|---|
| ESC | Restarts the program |
| STOP | Ends the program |

When this message is displayed because the communication parameters have not been correctly set, end FILINK execution and respecify the RS-232C settings with the CONFIG command.

**(7) Directory full**
**Press ESC to restart, STOP to exit FILINK.**
**Cause:** The directory space on the destination disk is full.
**Handling:** When this message appears, keys other than the following are ignored.

| | |
|---|---|
| ESC | Restarts the program. |
| STOP | Ends the program. |

Change to another disk, then start the FILINK program again. The file being transferred when the error occurred must be resent from the beginning.

**(8) Disk full**
**Press ESC to restart, STOP to exit FILINK.**

**Cause:** There is no free space left on the destination disk.
**Handling:** When this message appears, keys other than the following are ignored.

| | |
|---|---|
| ESC | Restarts the program. |
| STOP | Ends the program. |

Change to another disk, then start the FILINK program again.

**(9) Close error**
**Cause:** An error occurred while a file was being closed.
**Handling:** Change to another disk, then start the FILINK program again.

**(10) Sender is not ready**
**Cause:** The sending computer did not become ready to send within 10 seconds of the time the receiving side bega waiting for file transmission. Or, the RS-232C cable is not connected to the receiving computer.
**Handling:** Make sure that the sending computer is ready to send and that the RS-232C cable is connected, then continue operation.

**(11) Receiver is not ready**
**Cause:** The receiving computer did not become ready to receive within 10 seconds of the time the sending computer became ready to send.
**Handling:** Make sure that the receving computer is ready to receive, then continue operation.

**(12) RS-232C is not ready**
**Cause:** The RS-232C cable is not connected to the sending computer.
**Handling:** Make sure that the RS-232C cable is connected, then continue operation.

## Notes Concerning Use

(1) File names on the receiving side
- When file names are specified on the receiving side:

(a) When an unambiguous file name is specified on the receiving side (when wildcard characters are not used in the file specification), the file received is written to disk under that file name.
(b) When an ambiguous file name is specified on the receiving side (when wildcard characters are included in the file name specification), the name under which the file is written to the disk combines the unambiguous portion of the file name specified with those portions of the file name received which correspond to the wildcard characters. Spaces in the file name are padded with dollar signs ($).

- When no file name is specified on the receiving side, the file name received is used as is. Spaces in the file name are not padded with dollar signs.

(2) Setting up the data link between the sending and receiving computers
If the sending computer becomes ready before the receiving computer, factors related to signal timing make it more likely that communication errors (framing errors) will occur at lower baud rates (1200 bps or less). To avoid this, you should make the receiving computer ready first.

(3) Any file in the destination drive whose extension is $$$ will be deleted unconditionally if it has the same file name as that under which a file is being received.

(4) If file reception is interrupted by pressing CTRL + STOP, the file being written is deleted from the disk. However, if file reception is interrupted because the disk becomes full, the file is closed and remains on the disk becomes full, the file is closed and remains on the disk; in this case, the file extension is $$$.

(5) Errors occurring during file transfer
Check data is inserted into file names and data transferred to increase communication reliability. If any data error is detected during file transfer, related characters are displayed on the screen, then FILINK resends the packet in which the error occurred. This continues until data transfer has successfully completed.

# LOAD (Loader)

**Purpose**

The LOAD command generates executable machine code files (.COM files) from Intel HEX format object files generated by the ASM utility and saves them to the specified disk. The file name of the .HEX file is used for the .COM file.

**Format**

**LOAD dr:filename**

**Example**

The following example generates file SAMPLE.COM from file SAMPLE.HEX and saves the new .COM file to the disk in drive E.

```
D>LOAD E:SAMPLE   RETURN

FIRST ADDRESS    0100
LAST ADDRESS     0105
BYTES READ       0006
RECORDS WRITTEN  01

D>
```

FIRST ADDRESS indicates the starting address of the program in memory, and LAST ADDRESS indicates the ending address of the program in memory.

BYTES READ indicates the size of the program, and RECORDS WRITTEN indicates the number of 128-byte records making up the file.

You can use the DIR or STAT commands to verify that a .COM file has been generated from the specified .HEX file. For example, the directory of drive E before and after the example above would differ as follows.

Before LOAD execution:

```
A>DIR SAMPLE.*   RETURN
A: SAMPLE       ASM : SAMPLE   HEX
A: SAMPLE       PRN
```

After LOAD execution:

```
A>DIR SAMPLE.*   RETURN
A: SAMPLE       ASM : SAMPLE   HEX
A: SAMPL        PRN : SAMPLE   COM
```

# PIP

**Purpose**

The PIP program is used to copy files between peripherals, e.g. from disk to disk, from a disk to a printer etc. The name comes from the initial letters of the words Peripheral Interchange Program.

**Format**

**PIP <filespec> = <filespec>**

**Explanation**

PIP can be used with wildcard characters to transfer all types of files. It can also be used to perform other important functions such as:

- Remove part of a file (either at the beginning, end or in the middle of a file).
- Convert all characters to upper case or lower case
- Join a number of files together
- Make a backup file under a different name
- Add sequential numbers to each logical line of text
- Reform the page length of a text file

There are two ways PIP can be used.

1) When only a single operation is required, PIP can be used by following the PIP command with a command string which designates the operation you wish to carry out. On completion of the operation the CP/M prompt is returned or the MENU display if it is switched on. When using PIP from the MENU, the command string can be typed on the MENU command line after selecting PIP.

2) When a number of operations are to be performed, the PIP program can be loaded into memory. The CP/M prompt (A>, B>, etc.) is replaced by a PIP prompt (an asterisk). The use of PIP in this manner is essentially the same as when only one operation is required. This method must be used in certain cases; e.g., when starting or ending file transfer using lowercase. When PIP is chosen from the MENU, the PIP screen will clear and the PIP asterisk prompt will appears unless an additional command string is entered.

Pressing the RETURN key, the STOP key or CTRL - C returns display to the MENU screen or the CP/M system prompt, whichever is set.

When using PIP, you must specify a source device/file and a destination device/file. The source device/file is always specified last.

## 1. COPYING A FILE FROM DRIVE TO DRIVE

Any file excepts a system file can be copied from one disk to another as follows:

**D>***PIP H:=A:INFO.DAT* RETURN

This copies the file INFO.DAT from drive A: to drive H: then returns to the system prompt. The file is recorded under the same filename on the new drive. The filename can be changed, for example:

**D>***PIP H:NEWNAME.DAT=A:INFO.DAT* RETURN

By using this option to copy a file to the same drive, you can make a backup copy of a file under a different name.

Various options can be added to the end of the PIP command string. For example specifying the [V] option causes the file to be verified as it is copied:

**D>***PIP H:=A:INFO.DAT[V]* RETURN

## 2. COPYING A FILE FROM DISK TO PRINTER

This has almost the same effect as pressing CTRL - P then asking the computer to TYPE a file:

**D>***PIP LST:=A:LETTER.TXT* RETURN

The file is printed character by character, and words which come at the end of a line are likely to be split in the middle of the word.

A refinement of this command is to add the option [Pn]; this causes the printer will execute a form feed every n lines. This prevents lines from being printed on the perforations. For example:

```
D>PIP LST:=A:INFO.DAT[P60]    RETURN
```

will force a form feed every 60 lines giving the same effect as a one-inch skip-over on paper which takes 60 lines.

An additional option makes it possible to echo the file to the screen at the same time as it is printed. This is achieved using the [E] option:

```
D>PIP LST:=A:INFO.DAT[E]    RETURN
```

If both options are required, the command would be:

```
D>PIP LST:=A:INFO.DAT[P60E]    RETURN
```

## 3. COPYING A FILE FROM DISK TO SCREEN

The use of this command will have the same effect as the [E] option when copying to any other device. It is also the same as using TYPE command. The format is:

```
D>PIP CON:=A:MEMO.DOC    RETURN
```

One of the options that can be specified here to good effect is [N] or [N2]. This causes line number to be added at the beginning of each line in the form 01 if [N] is specified and 000001 if [N2] is specified:

```
D>PIP CON:=A:MEMO.DOC[N2]    RETURN
```

In addition, all lower case characters can be converted to upper case, and vice versa, using the [U] and [L] options:

```
D>PIP CON:=A:MEMO.DOC[N2U]    RETURN
```

will give

```
000001 THIS IS A LINE OF TEXT
```

```
000002 HAVE A NICE DAY
```

and

D>*PIP CON:=A:MEMO.DOC[NL]* RETURN

will give

```
01 this is a line of text
02 have a nice day
```

**4. COPYING A FILE TO AN EXTERNAL DEVICE**

Files can be copied to external devices such as printers, disk drives and other computers by specifying the relevant output port as the destination. For instance:

D>*PIP TTY:=A:DEMO.BAS* RETURN

will copy the file to a printer or any other device connected to the high-speed serial output port.

## OPTIONAL PARAMETERS AVAILABLE WITH PIP

There are nineteen options which can be used with the PIP command, including those which have been mentioned already:

**1. Block mode transfer**

**Format [B]**

This option causes the data to be transferred in blocks. (This option is provided for use with devices such as paper tape readers, and is ordinarily not used.)

**Example**
D>*PIP A:TEST.FIL=RDR:[B]* RETURN

**2. Echo to screen**

**Format [E]**

This option causes the screen to display all the data transferred so that you can see what is being copied.

**Example**
**D>***PIP LST:=A:TEST.FIL[E]* RETURN

It is advisable to user the [V] entry (i.e., to verify the file) when using the [E] option.

**3. Form feed insertion**

**Format [Pn]**

This makes it possible to force a form feed (ASCII code 12 or hex 0C) every n lines. It is particularly useful when sending files to the printer, because it allows the perforations in continuous forms to be skipped to improve legibility. If n is omitted or given as 1, PIP assumes a form feed is required every 60 lines (that is, the default setting). If your paper length is 66 lines per page, use the parameter as follows:

**D>***PIP LST:=A:REPORT.TXT[P66]* RETURN

Remember that this code is also the 'clear screen' code, so it is advisable to take care when using this option. If this option is used with the [F] parameter, the [F] should come first:

**D>***PIP LST:=A:REPORT.TXT[FP66]* RETURN

**4. Form feed suppression**

**Format [F]**

This is most useful when using PIP with a printer. It suppresses the form feed character (ASCII code 12 or hex 0C) which otherwise would cause the printer to feed a sheet through when it encounters the code.

**Example**
**D>***PIP TTY:=A:TEXT.DOC[F]* RETURN

In conjunction with [Pn] it can be used to change the page lengths by copying a file, removing previously added form feeds (e.g., ones inserted by PORTABLE WORDSTAR) and adding them at a different place.

**5. Hex format**

**Format [H]**

When this option is specified, data transferred is checked to ensure it is in Intel HEX format. If there is a discrepancy, transfer is terminated.

**Example**
A>*PIP DEMO2.HEX=DEMO1.HEX[H]* RETURN

**6. Ignore NULL records**

**Format [I]**

This can be used as an alternative to [H] - it causes PIP to ignore NULL records (hex 00) and ensures the data is in Intel HEX format. It is therefore an extension of the [H] option.

**Example**
A>*PIP DEMO2.HEX=DEMO1.HEX[I]* RETURN

**7. Lower case conversion**

**Format [L]**

This option converts all upper case characters to lower case as they are transferred.

**Example**
D>*PIP LST:=A:LITTLE.DOC[L]* RETURN

**8. Numbering lines**

**Format [N]**
**[N2]**

When sending programs to another device it may be useful to have the lines numbered. PIP regards a line as a series of characters terminated by a carriage return (ASCII code 13 or 0D hex). Specifying [N] will begin the file at column 9 of the screen and display the number of the line with a colon (:) in the seventh position followed by a space. For example

```
D>PIP E:=A:TEST.DOC[N]   RETURN

  1: This is line one
  2: This is line two
       *
 10: This is line ten
       *
100: This is line one hundred
```

Specifying [N2] fills all leading spaces with a zero and replaces the colon with a space.

```
D>PIP A:=E:TEST.DOC[N2]   RETURN

000001  This is line one
000002  This is line two
       *
000010  This is line ten
       *
000100  This is line one hundred
```

**9. Object file transfer**

**Format [O]**

Normally PIP can only copy standard ASCII or HEX files, but using this option allows it to transfer other types of files. Its effect is to ignore the physical end-of-file code a CTRL - Z (ASCII 26 or 1A hex) wherever it occurs in the object file, because in this context it will not be signalling the end of the file.

**Example**

```
D>PIP A:=E:OBJECT.FIL[O]   RETURN
```

It is not necessary to use this optional parameter with a COM file, because PIP adds it automatically. However, when PIP is used with other machine code or object files which do not have the file extension COM, the [O] parameter MUST be used.

**10. Read system files**

**Format [R]**

This option makes it possible to read and copy system files, that is, files which do not appear in the directory and those with a filetype of SYS. It automatically sets the [W] option.

**Example**
**D>***PIP A:=E:.OSTAB.SYS[R]* RETURN

**11. Stop copying at specified string**

**Format [Qstring^Z]**

(^Z indicates CTRL - Z , which is entered by holding down the CTRL key and pressing Z .)

If you only want to transfer part of a file, this option causes PIP to stop file transfer when it finds the specified string. Only text preceding the string is copied.

**Example**
**D>***PIP LST:=A:REPORT.DOC[QTHE END^Z]* RETURN

If you want to search for a string containing lower case characters, start PIP and type the command string following the PIP prompt (*). (CP/M converts everything typed on the command line into uppercase, so lowercase strings cannot be entered from the CP/M command line.

**Example**
******LST:=A:REPORT.DOC[QThe End^Z]* RETURN

The [Q] option can be used with the [S] option to copy a section from the middle of a file.

## 12. Start copying at a specified string

**Format [Sstring^Z]**

This behaves in much the same way as the [Qstring^Z] option, except that it starts copying from the end of the specified string. The same conditions apply as with the [Q] option if you want to detect lowercase characters.

**Example**
**A>***PIP CHAPTER1.DOC=CHAPTER1.DOC [SINTRODUCTION^Z]* RETURN

The [S] and [Q] options can be used together if you want to copy a section from the middle of a file.

**Example**
**CHAPTER1.DOC=CHAPTER2.DOC[SPreface^ZQlast. ^Z]* RETURN

## 13. Tab settings

**Format [Tn]**

Tab settings in the destination file can be changed from those of the original with this option. The number given by n puts tabs settings at columns n, 2n, 3n, and so forth. For instance, [T9] puts tabs settings at columns 9, 19, 29, 39, etc. These settings are used wherever PIP comes across a TAB character in the file it is copying. The TAB characters is CTRL - I (ASCII code 9).

If the word processor or whatever created the text file, inserts spaces instead of using the TAB character, this option has no effect.

**Example**
**D>***PIP CON:=A:PROGRAM.ASM[T10]* RETURN

## 14. Transfer between user areas

**Format [Gn]**

Normally, it is not possible to use files from other than the current user area. However, specifying this option allows transfer of files from user area n to the current area.

**Example**
**D>***PIP E:TEST.DOC=E:DEMO.DOC[G3]* RETURN

**15. Truncate lines of data**

**Format [Dn]**

This option allows truncation of lines at column n; that is, PIP deletes all characters between column n and the next carriage return.

**Example**
**D>***PIP LST:=E:PROGRAM.BAS[D80]* RETURN

**16. Uppercase conversion**

**Format [U]**

This is the opposite of the [L] option; it converts all lowercase characters to uppercase.

**Example D>***PIP TTY:=A:BIGTYPE.TXT[U]* RETURN

**17. Verify the copy**

**Format [V]**

Verifying a file as it is copied acts as a double check on its integrity. When given this option PIP compares the copy it has made with the original as it goes along, ensuring a faithful reproduction containing only those errors that were in the original.

**Example D>***PIP E:=A:PERFECT.COM[V]* RETURN

**18. Write to a read/only file**

**Format [W]**

If you have files which have been set to read-only with STAT, you can copy over them if you give this option (PIP does not ask if you want the existing file erased and overwritten). It is not possible to reverse the process if you make a mistake; use with care

**Example**
D>*PIP A:=E:SECURE.COM[W]* RETURN

**19. Zero parity setting**

**Format [Z]**

The highest bit of each byte is usually the parity bit, and this option sets all highest-order bits to zero, thus converting 8-bit ASCII bytes to 7-bit ASCII bytes. When using this option make sure that none of the characters sent use this bit because otherwise you could get some strange results. For example, graphics characters and some console ESC codes would be changed.

**Example**
D>*PIP E:=A:ORDINARY.FIL[Z]* RETURN

## USING THE PIP PROMPT (*)

If you want to transfer a large number of files or transfer files between two data disks, it may be more convenient to give the PIP command on its own, producing the * prompt:

The command string can then be typed for each operation. For example to copy all COM files to drive E: from drive C:, use PIP as follows:

```
D>PIP  RETURN
*A:=C:*.COM  RETURN

COPYING
PIP.COM
STAT.COM
CONFIG.COM
*
```

To output a file to the screen:

```
D>PIP  RETURN
*CON:=E:MEMO.DOC  RETURN
```

At the * prompt you can type in any PIP commands you like without having to load PIP for each operation. This can save a great deal of time if there are a number of operations to be carried out.

Once you have finished, pressing the RETURN key, or the STOP key will return you to the system prompt on the drive from which you entered PIP (the default drive), or to the MENU if it is switched on.

# STAT

**Purpose**

The STAT program is used to display the STATistics or STATus of the various disk drives. This makes it one of the most useful utility programs.

**Explanation**

Many CP/M users do not use the full facilities provided by STAT. The following information is therefore provided as a list of operations, with a summary at the end.

## USING THE STAT PROGRAM

### 1. CHANGE DEVICE ASSIGNMENTS

**Format STAT logical: = physical:**

The device assignments can be altered with this command. However, it is more likely that the CONFIG program would be used instead since the devices are named in real terms rather than as codes.

For those who are familiar with the STAT command, the following table shows correspondance between the physical devices and those implemented:

| | | | | |
|---|---|---|---|---|
| LST: | TTY: Serial (printer) | CRT: Cartridge Printer | LPT: RS-232C (printer) | UL1: Parallel (printer) |
| PUN: | TTY: not implemented | PTP: LCD display | UP1: RS-232C | UP2: not implemented |
| RDR: | TTY: Keyboard | PTR: not implemented | UR1: RS-232C | UR2: not implemented |
| CON: Output Input | TTY: RS-232C Keyboard | CRT: LCD Keyboard | BAT: LCD RS-232C | UC1: RS-232C RS-232C |

For instance, to tell the computer that the printer is now attached to the serial port instead of the RS-232C port, the command

```
D>STAT LST: = TTY:  RETURN
```

is given, after which all output destined for the printer is sent to the serial port instead of the RS-232C port. Then, if the STAT DEV: command is given, the result will be:

```
CON: is CRT:
RDR: is UR1:
PUN: is UP1:
LST: is TTY:
```

## 2. CHARACTERISTICS

**Format  STAT DSK:**

The complete status of a disk is displayed using this command. It shows the status of both the current disk and that of any others that have been accessed during the same session:

This command displays the characteristics of the disks accessed, such as its capacity; for example:

```
A>D:STAT DSK:  RETURN
```

|  |  |  |
|---|---|---|
| A : | Drive characteristics | (drive name) |
| 72 : | 128 Byte Record Capacity | (no.of 128 byte records allowed) |
| 9 : | Kilobyte Drive Capacity | (formatted capacity of drive) |
| 16 : | 32 Byte Directory Entries | (no.and size of directory entries) |
| 0 : | Checked Directory Entries | (no.of checked directory entries) |
| 128 : | Records/Extent | (no.of records per extent) |
| 8 : | Records/Block | (no.of records per block) |
| 64 : | Sectors/Track | (no.of sectors per track) |
| 0 : | Reserved Tracks | (no.of tracks reserved for CP/M) |

## 3. HELP

**Format STAT VAL:**

This acts as a sort of HELP command. It shows the formats of the various STAT commands which can be given to obtain information or alter device attributes and assignments:

```
D>STAT VAL:  RETURN

  Temp R/O Disk:   d:=R/O
 Set Indicator:    d:filename.typ $R/O $R/W $SYS
                   $DIR
 Disk Status:      DSK:  d:DSK:
 User Status:      USR:
 Iobyte Assign:
 CON: = TTY: CRT: BAT: UC1:
 RDR: = TTY: PTR: UR1: UR2:
 PUN: = TTY: PTP: UP1: UP2:
 LST: = TTY: CRT: LPT: UL1:
```

## 4. DISPLAY DEVICE ASSIGNMENTS

**Format STAT DEV:**

This shows which physical devices are assigned to the various logical devices. To check the devices by name, use the CONFIG program.

```
D>STAT DEV:  RETURN

CON: is CRT:
RDR: is UR1:
PUN: is UP1:
LST: is LPT:
```

## 5. READ/ONLY - PROTECT ALL THE FILES ON A DISK

**Format STAT drivename:=R/O**

An entire disk can be set to read/only with this command. The read/only setting remains effective until either a warm or a cold start is made:

```
D>STAT E:=R/O   RETURN
```

The Read/Only command protects the files on a disk so that they cannot be erased or written to. If you try to write or erase a file with the R/O attribute, the following error message is displayed:

```
D>ERA E:LETTER.DOC   RETURN
BDOS ERROR ON E: R/O
```

Pressing either the RETRUN key, the STOP key, or CTRL - C will return you to the prompt.

It is also possible to protect or un-protect single files with STAT.

## 6. READ/ONLY - PROTECTING A FILE

**Format   STAT drivename:filename.filetype $R/O**

Any file can be set to read/only using this format. This prevents the file from being altered until it is reset to read/write:

```
D>STAT A:DOCUMENT.TXT $R/O   RETURN
DOCUMENT.TXT set to R/O
```

## 7. READ/WRITE - UN-PROTECTING A FILE

**Format   STAT drivename:filename.filetype $R/W**

This resets a file to allow it to be written to as well as read.

```
D>STAT A:DOCUMENT.TXT $R/W   RETURN
DOCUMENT.TXT set to R/W
```

If a file is protected and the whole disk is subsequently protected, the file will still be protected when a warm or cold boot is made to remove protection from the disk.

## 8. HIDE FROM DIRECTORY - SPECIFIED FILE

**Format   STAT drivename:filename.filetype $SYS**

It is possible to give a file SYStem status using this command. This effectively removes its name from the directory so it cannot be used by anyone who does not know it exists:

```
D>STAT A:DOCUMENT.TXT $SYS   RETURN
DOCUMENT.TXT set to SYS
```

If a STATus is carried out on the file when it is set to a SYS file, it shows the filename in brackets

```
D>STAT A:DOCUMENT.TXT   RETURN

 Recs   Bytes   Ext   Acc

  41     6k      1     R/O A:(STAT.COM)

Bytes Remaining On A: 3k
```

## 9. RESTORE TO DIRECTORY - SPECIFIED FILE

**Format   STAT drivename:filename.filetype $DIR**

This countermands the previous format, resetting the file so that it is displayed in the directory:

```
D>STAT A:DOCUMENT.TXT $DIR   RETURN
DOCUMENT.TXT set to DIR
```

## 10. SIZE AND ATTRIBUTES - SPECIFIED FILE

**Format   STAT drivename:filename.filetype**

The size and attributes of the specified file are displayed using this command. It gives specific information about the number of records (Recs), number of bytes (Bytes), number of extents (Ext) and read/write status (Acc) of each file on the disk, followed by the total number of unused bytes remaining. A complete file name can be given, or wildcard characters can be used to specify a number of files:

**a) Information on a particular file**

```
D>STAT A:TESTING.COM   RETURN
```

```
 Recs    Bytes    Ext    Acc

  16      4k       2       R/O A:TESTING.COM

Bytes Remaining On A: 4k
```

b) Information on all files

D>*STAT A:\*.**  RETURN

```
 Recs    Bytes    Ext    Acc

  64      8k       1       R/W A:CONFIG.COM
  22      3k       1       R/W A:FILINK.COM
  58      8k       1       R/W A:PIP.COM
  41      6k       1       R/W A:STAT.COM
  10      2k       1       R/W A:SUBMIT.COM
  24      3k       1       R/W A:TERM.COM
   6      1k       1       R/W A:XSUB.COM

Bytes remaining on C: 1k
```

c) All files with a particular extension

D>*STAT A:\*.COM*  RETURN

```
 Recs    Bytes    Ext    Acc

  58      8k       1       R/W A:PIP.COM

Bytes remaining on A: 0k
```

d) Files containing specific characters

D>*STAT E:DEMO??.BAS*

```
 Recs    Bytes    Ext    Acc

  20      3k       1       R/W E:DEMO1.BAS
  16      2k       1       R/O E:DEMO13.BAS
  14      2k       1       R/W E:DEMO1A.BAS

Bytes remaining on E: 258k
```

## 11. SIZE AND ATTRIBUTES - SPECIFIED FILE

**Format** STAT drivename:filename.filetype $S

Using this form of the command will give the same information as without the $S option, but with the addition of the size of the file. This value is the same as the number of records for sequential access files, and is generally used to show the amount of space that has been reserved for a random access file. This is because a sequential file simply takes up space as it is added to, and a random access file has an amount of space allocated to it when it is created:

```
A>STAT A:DOCUMENT.* $S   [RETURN]

Size  Recs  Bytes  Ext  Acc

 16    6     2k     1    R/W A:DOCUMENT.TXT
  6    6     2k     1    R/W A:DOCUMENT.BAS

Bytes remaining on A: 4k
```

## 12. SPACE REMAINING ON DISK

**Format STAT**

This form of the command will display the amount of space available on the current drive and on any other drives which have been accessed during the current session. It also shows the drives' read/write attributes:

a) **D>*STAT*** [RETURN]

b) **D>STAT** [RETURN]

```
A: R/W, Space:   0k
D: R/O, Space:  33k
H: R/W, Space:  24k
```

## 13. SPACE REMAINING ON SPECIFIED DISK

**Format STAT drivename:**

This form of the command gives the amount of space remaining on the specified drive:

```
D>STAT A:   RETURN

Bytes Remaining On A: 0k
```

## 14. USER STATUS

**Format   STAT USR:**

This format is used to display the current USR number and USR numbers under which there are active files on the disk:

```
D>STAT USR:   RETURN

Active User: 0
Active Files: 0 1
```

This indicates that the current USR number is 0 and that USR numbers 0 and 1 both have active files on the disk.

## SUMMARY OF STAT COMMANDS

| | |
|---|---|
| **STAT** | SPACE REMAINING ON DISK |
| **STAT DEV:** | LOGICAL TO PHYSICAL ASSIGNMENTS |
| **STAT drivename:** | SPACE REMAINING ON SPECIFIED DISK |
| **STAT drivename:=R/O** | READ/ONLY - SET SPECIFIED DISK |
| **STAT drivename:filename.filetype** | SIZE AND ATTRIBUTES - SPECIFIED FILE |
| **STAT drivename:filename.filetype $DIR** | REPLACE IN DIRECTORY - SPECIFIED FILE |
| **STAT drivename:filename.filetype $R/O** | READ/ONLY - SET SPECIFIED FILE |
| **STAT drivename:filename.filetype $R/W** | READ/WRITE - SET SPECIFIED FILE |
| **STAT drivename:filename.filetype $S** | SIZE AND ATTRIBUTES - SPECIFIED FILE |

| | |
|---|---|
| **STAT drivename:filename.filetype $SYS** | |
| | REMOVE FROM DIRECTORY - SPECIFIED FILE |
| **STAT DSK:** | DISK STATUS |
| **STAT logical: = physical:** | |
| | CHANGE DEVICE ASSIGNMENTS |
| **STAT USR:** | USER STATUS |
| **STAT VAL:** | HELP |

# SUBMIT

Purpose

Allows a series of CP/M commands and/or utility programs to be batched for execution in sequence.

Format

**SUBMIT filename**

Explanation

When you want to execute a group of commands repeatedly in a certain sequence, you can include them all in a special file called a submit file (a file whose extension is .SUB) and execute them using the SUBMIT utility instead of typing them individually. Commands included in the submit file are executed as follows.

**D>*SUBMIT filename*** RETURN

One useful function of the SUBMIT utility is to cause a string of commands to be executed automatically when PX-4 is started by the AUTOSTART or WAKE function. Procedures for using the SUBMIT utility are as follows.

**1. Contents of the submit file**

The submit file lists the commands/utilities which are to be executed by the SUBMIT utility. All commands/utilities included in this file must return to the CP/M system prompt upon completion; otherwise, the next command in the sequence will not be executed.

The file can be created using the ED utility, any word processing program, or from BASIC.

The following is an example of a simple series of jobs to be performed by a submit file.

a. Display the directory of the disk in drive A:.
b. Show the sizes of the files.
c. Erase all files whose extension is .BAK.
d. Move all BASIC files to the disk in drive E:.

The sequence of commands which does this is as follows.

```
DIR A:
STAT A:*.*
ERA D:*.BAK
PIP E:=A:*.BAS
```

If this series of commands were held in a file called START.SUB in drive D:, they could be executed as follows.

```
D>SUBMIT START  RETURN
```

When executed, each command and its results are shown on the screen just as if you had entered them from the keyboard.

When command parameters vary, it is possible to insert variables into the submit file and to specify the parameters themselves on the command line when SUBMIT is executed. Variables are inserted in the form "$1", "$2", "$3", etc., where the first parameter specified on the command line is substituted for "$1", the second for "$2", and so on. The format of the command line in this case is as follows:

```
D>SUBMIT filename X1 X2 X3  RETURN
```

where X1 is substituted for all occurrences of "$1", X2 is substituted for all occurrences of "$2", and X3 is substituted for all occurrences of "$3".

For instance, if submit file BEGIN.SUB contained

```
DIR A:*.BAS
B:BASIC $1
STAT A:*.BAS
```

SUBMIT might be executed as follows:

```
D>SUBMIT BEGIN A:PROG1  RETURN
```

The result would be to display the directory of drive A:, start up BASIC from drive B:, execute BASIC program file PROG1 from drive A:, then display the size of all BASIC program files in drive A:.

*NOTE:*
*Ordinarily, BASIC returns to the "Ok" prompt upon completion of program execution rather than to CP/M. To return to CP/M for execution of the next command in the submit file, end the BASIC program with the SYSTEM command.*

With some CP/M programs (for example STAT), the "$" character is used in specifying option parameters. To prevent SUBMIT from interpreting such parameters as variables, add an extra "$" wherever such an option parameter is used. When SUBMIT encounters two "$" symbols in sequence, it ignores the first and the second is used in the normal manner. This is shown in the example below, where STAT sets the Read/Only attribute for all files in drive A:.

```
STAT A:*.* $$R/O
```

*WARNING:*
*The SUBMIT utility writes a temporary file to drive A: when it is executed. If the drive is write protected, an error will occur and execution will abort at that point. This also applies if the size of the RAM disk is set to zero bytes; in this case a "DIRECTORY FULL" error is generated.*

**2. Creating a submit file in BASIC**

If you are not practiced in using the ED utility, you may find it easier to create submit files with BASIC. This can be done very simply by creating a sequential text file and writing the commands to it as data. For example, the series of commands given above could be placed into the file named START.SUB with the following program.

```
10 OPEN"O",#1,"D:START.SUB"
20 PRINT#1,"DIR A:"
30 PRINT#1,"STAT A:*.*"
40 PRINT#1,"ERA D:*.BAK"
50 PRINT#1,"PIP E:=A:*.BAS"
60 CLOSE
70 END
```

Execution of this program creates a file called START.SUB on the disk in drive E:. The commands in this file can then be executed by typing:

```
D>SUBMIT E:START   RETURN
```

**3. Creating a submit file with ED**

Submit files can also be created using the ED utility. In this case, the procedure is as follows:

```
D>ED E:START.SUB   RETURN

NEW FILE
     :  * I   RETURN
    1:     DIR A:   RETURN
    2:     STAT A:*.*   RETURN
    3:     ERA D:*.BAK   RETURN
    4:     PIP E:=A:*.BAS   RETURN
    5:     CTRL-Z
     :  *E   RETURN
```

See the description of the ED utility for more information on using it to create and edit text files.

# TERM

**Purpose** The TERM program makes it possible to connect PX-4 to a host computer through the RS-232C interface or direct modem for use as a terminal.

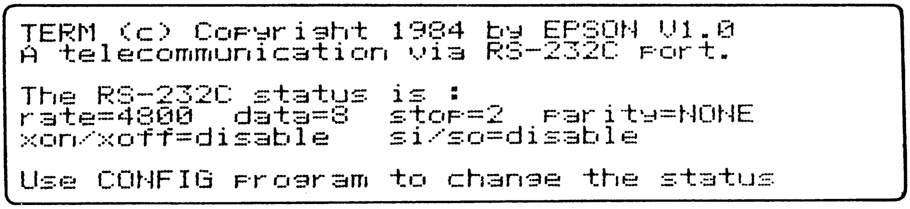
**Format** **TERM**

**Explanation** The TERM program is started by entering the TERM command as shown below or by selecting "TERM.COM" in the MENU screen.

**Example:**

**A>*D:TERM*** RETURN

When the TERM command is entered, the following screen is displayed.



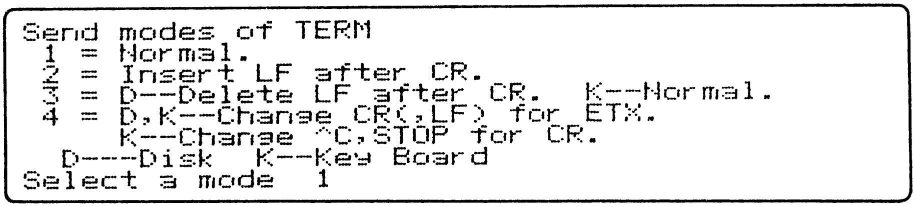
```
TERM (c) Copyright 1984 by EPSON V1.0
A telecommunication via RS-232C port.

The RS-232C status is :
rate=4800  data=8  stop=2  parity=NONE
xon/xoff=disable   si/so=disable

Use CONFIG program to change the status
```

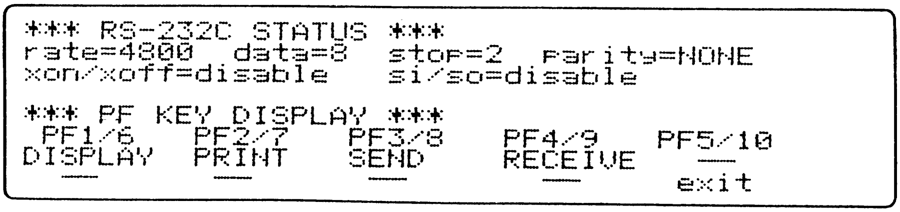
*NOTE:*
*When the direct modem is used, "D-MODEM" is displayed instead of "RS-232C."*

Pressing any key changes the display as follows.



```
Send modes of TERM
 1 = Normal.
 2 = Insert LF after CR.
 3 = D--Delete LF after CR.  K--Normal.
 4 = D,K--Change CR(,LF) for ETX.
     K--Change ^C,STOP for CR.
  D----Disk  K--Key Board
Select a mode  1
```

Enter any of 1 to 4 to select a mode, then press the RETURN key; PX-4 then enters the terminal mode. Pressing SHIFT + ESC in the terminal mode displays the following screen so that the current RS-232C status and PF key functions can be obtained.



```
*** RS-232C STATUS ***
rate=4800  data=8   stop=2   parity=NONE
xon/xoff=disable    si/so=disable

*** PF KEY DISPLAY ***
 PF1/6    PF2/7    PF3/8    PF4/9    PF5/10
DISPLAY  PRINT    SEND     RECEIVE    —
  —        —        —        —       exit
```

PF key functions other than receive and exit are displayed in uppercase letters when they are OFF and in reversed uppercase letters when they are ON. "—" indicates that no function is assigned to the key.

Display returns to the former screen when any of the following occurs.

(1) SHIFT + ESC is pressed again.
(2) Any of the valid PF keys is pressed.
(3) A valid key is pressed to input data while keyboard input display is ON.
(4) Data is received from the RS-232C interface or direct modem.
(5) An error occurs.

The functions of the PF keys are as follows.

PF1: Switches keyboard data display ON or OFF. The function indication alternates as follows.

When ON -- DISPLAY     When OFF --

Display can be turned ON or OFF at any time. However, if keyboard input display is set to ON while receiving data, data input from the keyboard cannot be distinguished from that received. The default setting for keyboard data display is OFF.

PF2: Switches printer output ON or OFF. The function indication alternates as follows.

When ON -- PRINT    When OFF --

"Send" and "Receive" indications are not displayed when this switch is ON.

This function switches output of receive data to the printer (connected to the parallel printer output connector) ON or OFF. Keyboard input data can be printed only when keyboard data display is ON and printer output is ON. However, output to the printer is inhibited during file data transfer. The printer connected must be equipped with a Centronics interface. The default setting for printer output is OFF.

PF3: Starts sending a file. The "Send" indication is displayed in uppercase letters while data is being sent. The "Print" and "Receive" indications are not displayed. First, TERM prompts for input of a file name. When the file exists, TERM then prompts the operator to enter the interval between characters, that following CR-LF sequences, and that following each 128-byte block. Each interval is set as a multiple (0 to 255) of 20 ms. The default settings for the first two intervals are 0 and that for the third is 175. This interval is required when files transferred are to be saved to a disk in a terminal floppy unit. After the intervals have been entered, TERM outputs the contents of the file to the RS-232C interface or direct modem. When data transfer is completed, the function key indications return to their initial state. File output can be stopped by pressing the $\boxed{\text{PF3}}$ key.

PF4: Starts receiving data. $\boxed{\text{SHIFT}}$ + $\boxed{\text{ESC}}$ is ignored until data transfer is completed. When a file name is entered, a file is created for storing data received. If code 1BH (ESC) is received, it is always converted into code 2EH (period ".") before it is output to the printer. Pressing $\boxed{\text{PF4}}$ during data transfer closes the file and stops data reception.

PF10: Terminates TERM operation.

HELP: The TERM command outputs the BREAK signal (a start bit with a duration of 200 ms) to the RS-232C interface when the HELP key is pressed. (The Break signal is used to interrupt transmission by the other computer.)

The other PF keys have no functions.

**Errors**

When an error or a condition which requires a user decision is detected during TERM operation, a message is displayed. Messages displayed for each error/condition and procedures for handling them are as follows.

**(1) RS-232C is not ready.**
**Cause:** An attempt was made to output data to the RS-232C interface without connecting the RS-232C interface cable.
**Handling:** Connect the RS-232C interface cable, then press CTRL + STOP to reset the error and start file output.

**(2) File not found**
**Cause:** The specified output file does not exist.
**Handling:** Respecify using the correct file name.

**(3) No file name specified**
**Cause:** An attempt was made to send without specifying a file name.
**Handling:** Specify the file name.

**(4) Bad file descriptor**
**Cause:** The file name specified does not conform to the rules for specifying file descriptors.
**Handling:** Correctly specify the file name.

**(5) Drive select error**
**Cause:** A drive name other than A: to K: was specified.
**Handling:** Specify a drive name from A: to K:.

**(6) Overwrite (Y/N) ?**
**Cause:** This message is issued on the receiving side when a file with the specified name already exists.
**Handling:** Type in Y or N. If Y is input, the existing file is overwritten; if N is input, TERM prompts for specification of another file name.

**(7) Communication error**
**Press ESC to restart, STOP to exit from TERM.**
**Cause:** A communication error (framing, buffer overflow, or receive overrun error) occurred, or the sending and receiving computers are not set with the same communication parameters (word length, parity, baud rate, etc.).
**Handling:** When this message appears, keys other than the following are ignored and the file being transferred must be resent from the beginning.

| | |
|---|---|
| ESC...................... | Restarts the program. |
| STOP.................... | Ends the program. |

When this message is displayed because the communication parameters have not been correctly set, end TERM operation and respecify the RS-232C settings with the CONFIG command.

**(8) Directory full**
**Press ESC to restart, STOP to exit from TERM.**
**Cause:** The directory space on the disk is full.
**Handling:** When this message appears, keys other than the following are ignored.

| | |
|---|---|
| ESC...................... | Restarts the program. |
| STOP.................... | Ends the program. |

Change to another disk, then start the TERM program again. The file being transferred when the error occurred must be resent from the beginning.

**(9) Disk full**
**Press ESC to restart, STOP to exit from TERM.**
**Cause:** There is no free space left on the disk.
**Handling:** When this message appears, keys other than the following are ignored.

| | |
|---|---|
| ESC | Restarts the program. |
| STOP | Ends the program. |

Change to another disk, then start the TERM program again.

**(10) File close error**
**Cause:** An error occurred while a file was being closed.
**Handling:** Change to another disk, then start the TERM program again.

# XSUB

**Purpose** Allows parameters of commands which used buffered console input to be included in submit files.

**Format** **XSUB**

**Explanation** With some commands (ED, DDT, and PIP), parameters are input from the keyboard following command execution; such commands are said to read buffered console input. For example, if the PIP command is executed without specifying any parameters, an asterisk is displayed to prompt for input of the destination and source files.

When such commands are included in a submit file (see the explanation of the SUBMIT utility), they ordinarily wait for parameters to be typed in from the keyboard instead of looking for them in the file.

The XSUB function makes it possible to pass parameters to such commands from within the submit file. Parameters passed may be specified either as literal character strings or as variables ($n); in the former case, parameters are passed to the command (or subcommand) exactly as specified; in the latter, parameters are specified on the command line together with the SUBMIT command and are substituted for variables at the appropriate points in the submit file.

For example, assume that command procedure files EX1.SUB and EX2.SUB have been saved to drive A: as follows.

**Example 1 - With XSUB**

| Submit file named EX1.SUB | Execution and result |
|---|---|
| xsub<br>dir a:<br>pip<br>$1=$2<br>^z | **D>***SUBMIT A:EX1 CON: A:EX1.SUB* RETURN<br><br>**D>***XSUB* RETURN<br><br>**D>***DIR A:* RETURN<br>**A: EX1 BAK : EX1 SUB** |

```
A: EX2          BAK : EX2       SUB
A: $$$          SUB
D>PIP
*CON:=A:TEST.SUB     RETURN

xsub
dir a:
pip
$1=$2
^z
*

(xsub active)
D>
```

**Example 2 - Without XSUB**

| Submit file named EX1.SUB | Execution and result |
|---|---|
| dir a:<br>pip | |

```
D>SUBMIT A:EX1 CON: A:EX1.SUB     RETURN

D>DIR A:    RETURN
A: EX1          BAK : EX1       SUB
A: EX2          BAK : EX2       SUB
A: $$$          SUB
D>PIP    RETURN
*CON:=A:TEST.SUB     RETURN
xsub
dir a:
pip
$1=$2
^z
*
```

Sometimes you may want to use both methods to pass parameters to commands in a submit file. In this case, the XSUB function can be deactivated by including DEXSUB in the submit file. See the explanation of the DEXSUB utility.

# *Chapter 4*

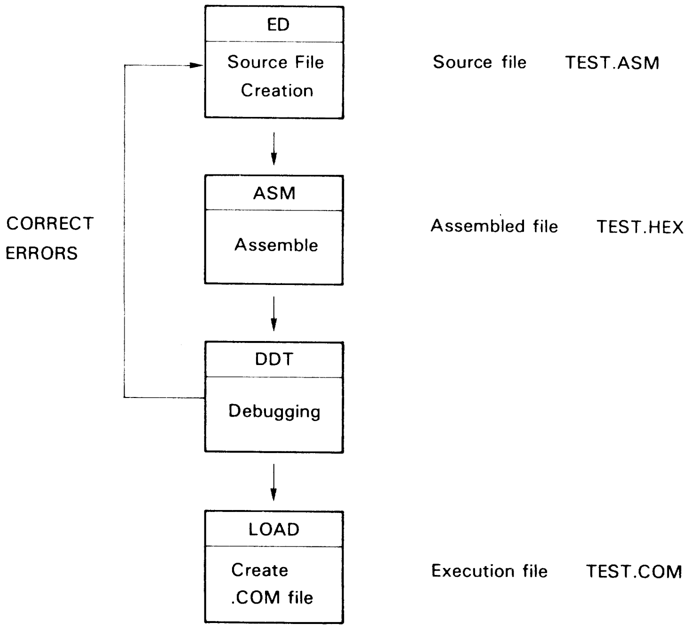
# *USING THE UTILITIES TO PREPARE PROGRAMS*

## 4.1 A BDOS Function Call Program

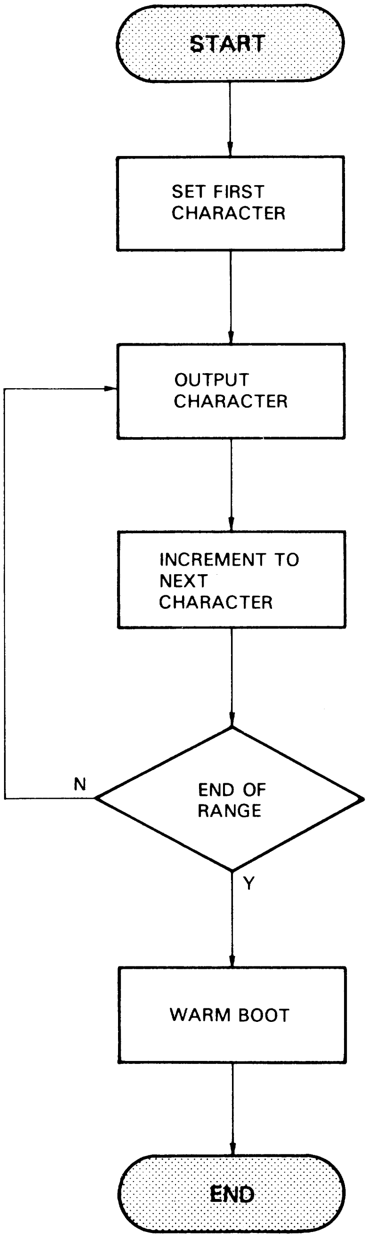
This chapter describes how to create a program using the ED, ASM, DDT and LOAD utilities. The program created outputs to the screen graphics characters with ASCII codes from 128 to 159.

The sequence used to produce the program is as follows:

1. Create the source program using ED.
2. Assemble the source file with ASM to produce a HEX file.
3. Debug the HEX file with DDT.
4. Convert the HEX file to the executable COM file with LOAD.
5. Run the program.



The following flowchart outlines the structure of the program.



## 4.1.1 Creating the Source File with ED

First, use the ED utility to create a source file consisting of 8080 mnemonics. This source file will be named TEST.ASM. In the following example the characters in ITALICS are the ones you should type.

After logging into drive D:, type

```
D>ED TEST.ASM   RETURN
```

on the CCP command line to start editing the file.

Since this is a new file, the following will appear on the screen.

```
NEW FILE
    : *
```

To start adding text to the file, place ED in the insert mode by typing I (for insert) and pressing the RETURN key.

A new line will be displayed on the screen with a line number followed by a colon. This and successive line numbers are generated by ED automatically. Remember that ED is a text line editor, not a word processor; the line numbers are output to the screen by the ED program to help you when typing in the program, and are not saved with the text to the file.

Files created by ED can be read by a word processor such as Portable WordStar. If you are more familiar with an editor such as Portable WordStar, you may find that it is easier to use that ED. In this case, simply type the program without the line numbers.

The screen will now show:

```
NEW FILE   RETURN
    : *I
   1:
```

Now type in the following program. You will notice that the characters are arranged in columns. This is not strictly necessary, but if you do write programs this way they are much easier to read and debug. After all characters in a particular column have been typed, the cursor can be advanced to the beginning of the next column by pressing the TAB key. If you wish to skip a column, simply press the TAB key until the cursor advances to the correct position.

The comments to the right of the program are there to help anyone reading the source listing to understand the program at a later date. The ASM program knows that they are comments because they begin with a semi-colon (;).

Press the RETURN key at the end of each line and the line number for the next line will be generated automatically. If you want a particular line to be blank, simply press the RETURN key without typing any characters into that line.

```
 1:  WBOOT    EQU    0000H
 2:  BDOS     EQU    0005H
 3:  OPUT     EQU    0002H
 4:
 5:  FIRST    EQU    128       ;ASCII code for first character
 6:                            ;to be printed
 7:  LAST     EQU    160       ;ASCII code for character after
 8:                            ;last character to be printed
 9:
10:           ORG    100H      ;location to start assembling
11:           MVI    E,START   ;load start character into E
12:                            ;register
13:           MOV    A,E       ;copy to accumulator for
14:                            ;output
15:  LOOP:    MVI    C,OPUT    ;put BDOS function number in C
16:                            ;register
17:           PUSH   D         ;save DE to stack
18:           CALL   BDOS      ;output the character
19:           POP    D         ;restore DE
20:           INR    E         ;set the next character
21:           MOV    A,E       ;copy into the accumulator for
22:                            ;checking and output
23:           CPI    LAST      ;check if last character has
24:                            ;been output
25:           JNZ    LOOP      ;if not output it
26:  RESET:   JMP    WBOOT     ;exit with a warm boot
27            END
```

(1) If you notice a typing error before you have pressed the RETURN key, use the backspace key to back up to the incorrect character, then type the correct character and retype the remainder of the line. The INS , DEL and cursor keys function as they would on the CP/M command line, and therefore cannot be used for error correction as they would be with a word processor. If the error is near the beginning of a line, it may be faster to complete the line and then correct it as if you had noticed the error after pressing the RETURN key.

(2) If the line has been entered by pressing the RETURN key, the best way to correct it is to use the search and replace facility. For example, suppose you have mistyped CPI as CPP on line 23 and noticed the mistake when you were about to enter line 26. Exit from the insert mode by pressing CTRL - Z . When the asterisk appears type:

    **23:**

    A new line will be generated on the screen

    **23: ***

    then you can now type in SCPP^ZCPI and press RETURN . This is the ED command to search for and replace the string.

    To display the line after making the correction (after the asterisk reappears), type 0 and press RETURN .

    To continue the ED session from line 26, type:

    **26:I**

    after the asterisk and press RETURN . This will place the character pointer at the beginning of line 26 and switch ED to the insert mode.

    Further details of using the ED command can be found in Chapter 3.

When you have finished entering the program, press CTRL - Z to exit from the insert mode. When the asterisk prompt appears, type E and press RETURN to save the file as TEST.ASM.

Before saving the file, you can display it to make sure that you have entered everything correctly by using ED's T command. Alternatively, you can use the TYPE command to display the file after returning to CP/M.

**D>*TYPE TEST.ASM*** RETURN

## 4.1.2 Assembling the Source File with ASM

The next step is to assemble the source file with the ASM utility to generate a HEX file. The source file name is TEST.ASM, so type the following:

**D>*ASM TEST***

and press RETURN . The screen will show the following if the assembly has been successful, then control will return to CP/M.

```
D>ASM TEST
CP/M ASSEMBLER - VER 2.0
0114
001H USE FACTOR
END OF ASSEMBLY
```

**Error Messages**

When the assembler detects errors in the source program during assembly, it displays an "E" followed by the relevant line(s) to indicate that the line contains an error. An example is shown below:

```
OASM TEST
CP/M ASSEMBLER - VER 2.0
S                              INC       E
E                              MOV       A E
E010A 40
0113
001H USE FACTOR
END OF ASSEMBLY
```

The example shown illustrates errors which would result if lines 18 and 19 of the source file were typed as follows.

```
20:     INC     E       ;set the next character
21:     MOV     A E     ;copy into the accumulator for
```

In the above example, the S on line 20 indicates a Syntax Error. Instead of INC E, the syntax should be INR E in 8080 machine code. The E to the left of line 21 indicates that the expression is incorrect. A comma (,) is required between the A and the E.

When you have corrected all errors, assemble the source file again. Now you will have two types of file: .HEX and .PRN. Data in the .PRN file is the same as that in the ASM file, but includes the machine codes which are generated by assembly together with the 8080 mnemonics. The .PRN file can be displayed by executing the TYPE command on the CP/M command line as follows.

```
  0000 =          WBOOT    EQU     0000H
  0005 =          BDOS     EQU     0005H
  0002 =          OPUT     EQU     0002H

  0080 =          START    EQU     128

  00A0 =          LAST     EQU     160


  0100                     ORG     100H
  0100 1E80                MVI     E,START

  0102 7B                  MOV     A,E

  0103 0E02       LOOP:    MVI     C,OPUT

  0105 D5                  PUSH    D
  0106 CD0500              CALL    BDOS
  0109 D1                  POP     D
 S                         INC     E
 E                         MOV     A E
 E010A 40
  010B FEA0                CPI     LAST

  010D C20301              JNZ     LOOP

  0110 C30000     RESET:   JMP     WBOOT
  0113                     END
```

Normally, the next step is to test and debug the .COM file using the DDT utility. However, this is a very short program and it does not require debugging (it was tested at the time this manual was written). If the expected output is not produced after preparing the program, check your listing against that shown in the manual.

In real life programs do not always work the way you expect them to the first time. The DDT utility is an invaluable tool for solving problems encountered and testing the various parts of a program. The next section illustrates how it can be used to trace and make minor modifications to the program in a dynamic way without having to go through the process of altering the .ASM file, reassembling it, and converting to a .COM file every time.

Since the DDT utility can work with either .HEX files or .COM files, it is customary to use the .HEX file in the debugging stage. Therefore, the file TEST.HEX is used for illustrating use of the debugger in the explanation below.

## 4.1.3 Using DDT

This section illustrates some of the uses of DDT in program development. It can be used with either a .COM file or a .HEX file.

To load file TEST.HEX into DDT type:

**D>***DDT TEST.HEX* RETURN

The display will then show

```
DDT VERS 2.2
NEXT  PC
0114 0000
-
```

The "-" sign is the DDT prompt.

The D command of DDT allows the bytes of code to be displayed together with the characters corresponding to any codes within the printable range. When the command:

*–D100 113* RETURN

is typed, the display will show the bytes of the program which are shown in the second block of the TEST.PRN file. To the right are a series of dots which replace the characters in the ASCII code table which are not printable, with two isolated "{" signs. These are present because there is a byte which corresponds to these characters present in the machine code. However, they are not relevent in this case. This facility is useful when searching for text strings.

A better command for showing the program is the L (for list) command. This will disassemble the code and show the mnemonics, but with the actual locations for JUMP and CALL commands. To see this type

*–L100 113* RETURN

and the following listing will appear:

```
0100  MVI   E,80
0102  MOV   A,E
0103  MVI   C,02
0105  PUSH  D
0106  CALL  0005
0109  POP   D
010A  INR   E
010B  MOV   A,E
010C  CPI   A0
010E  JNZ   0103
0111  JMP   0000
0114
```

This shows the program in mnemonic form with JUMPS and CALLS to their absolute locations instead of the labels in the assembly source code listing.

It is also possible to alter some codes at this stage. For example altering the byte at location 101 will change the first character printed. The DDT command S allows this to be done as follows. First type:

```
-S101  RETURN
```

and the contents of location 101 will be shown on the screen. By typing another value (8B for example), this value will be loaded into that location. Pressing the full stop allows you to exit from this alteration mode. The appearance of the screen after this operation would be:

```
-S101  RETURN
0101 80 8B  RETURN
0102 7B .  RETURN
-
```

Now when the program is executed, fewer characters will be output.

If you now exit DDT, you can save the modified file using the SAVE command of CP/M. This is described in the PX-4 Operating Manual.
When saving a modified file with the SAVE command, it is recommended that you save it using a file name which differs from that of the original program. To change the original file, correct the source code and reassemble the file. This way you always know that the .COM file corresponds to the .ASM file. If you change a file using just DDT, you may not be able to remember why or how it was corrected when you need to.

The most important function of DDT is to verify that a program works. This can be done in a number of ways. First, the program can be executed up to any point by adding a break point, then the contents of the registers can be checked to see that they are as you expect them to be. In this case the command:

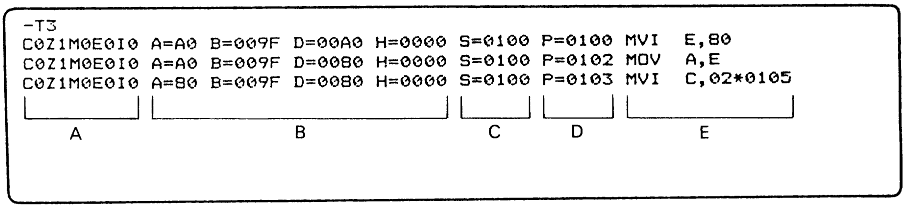
– *G100,111* RETURN

will execute the program which displays characters, but will stop before restarting the system. (Simply typing X at this stage will show the contents of the registers.) This procedure does not always work since you may not be able to regain control of the system.

During debugging, you may need to either change or just determine the value of the program counter. This can be done using the XP command. Before reading the following section, change the program counter back to 100H (the start of the program) by typing the following:

– *XP* RETURN
**P = 110** *100* RETURN
–

The second method of verifying program execution is to execute it step-by-step so that register contents, etc. can be checked after each instruction. The following traces operation of the first three instructions:



```
-T3
C0Z1M0E0I0 A=A0 B=009F D=00A0 H=0000 S=0100 P=0100 MVI  E,80
C0Z1M0E0I0 A=A0 B=009F D=0080 H=0000 S=0100 P=0102 MOV  A,E
C0Z1M0E0I0 A=80 B=009F D=0080 H=0000 S=0100 P=0103 MVI  C,02*0105
```

The first block (A) shows the flag settings, then the contents of the registers are shown in block B, with the accumulator on its own and then B and C, D and E and H and L as pairs. Block C shows the stack pointer and block D the program counter. Block E shows the mnemonic for the instruction and finally on the line of the last requested execution, the location of the next instruction is given.

By tracing more instructions, you can see the calls made to the BDOS; as each character is displayed, it appears on the right of the screen. (Scroll the screen to the right with SHIFT - → to see the right side.)

## 4.1.4 Creating a COM file

In order to create a COM file using the LOAD program, type

**D>***LOAD TEST.HEX* RETURN

The screen will show the following output.

```
FIRST ADDRESS 0100
LAST  ADDRESS 0113
BYTES READ     0014
RECORDS WRITTEN 01
```

This shows how many bytes are in the machine code file, and their start and end addresses when loaded into memory. The number of records shows how much space is taken up on the disk (one record corresponds to 128 bytes).

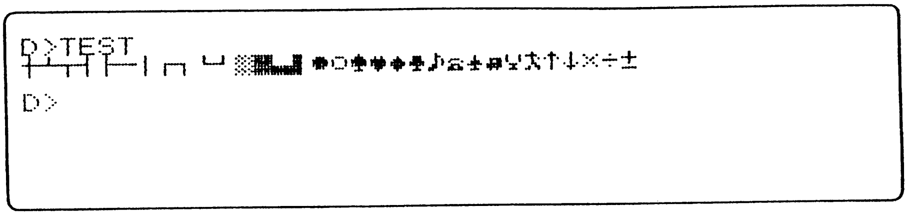
Now that te program has been saved as a .COM file it can be executed.

## 4.1.5 Executing the program

To execute the sample program TEST.COM, type

**D>***TEST* RETURN

The screen will show the graphics characters as below.



```
D>TEST
D>
```

## 4.2 Using a Submit File to Assemble a COM File

When assembling a file, you must enter a number of commands one at a time. In the process of correcting and debugging a program you will have to enter the same sequence of commands several times. You might find it easier to create a submit file to carry out the whole procedure at once.

The SUBMIT command executes the command from a SUB file as if you were entering them from the keyboard. Refer to the PX-4 User's Manual for details of how to use the SUBMIT command.

The following description shows how to use the ED program to make a SUB file for the TEST program.

### 4.2.1 Creating the SUB file with the ED utility

Start the ED utility as follows.

```
D>ED TEST.SUB    [RETURN]
```

When you type the I command, ED enters the insert mode to allow you to type characters into the edit buffer.

After typing in the command, press [CTRL] - [Z] to return to the command mode and use the E command to save the file to disk.

### 4.2.2 Executing the SUB file

After preparing the submit file, you can assemble and execute the TEST program by typing the following.

```
D>SUBMIT TEST    [RETURN]
```

Each command and its results will be displayed on the screen as the file is assembled and executed, just as if you had typed in the commands one by one.

# Appendix A

# DISK WRITE PROTECTION

As a precaution against altering the data on a disk when you do not want to, it is possible to set the disk to be "Write Protected". This can be done for individual files (or for all files on a disk) using the STAT utility in the utility ROM. (See the PX-4 Utility ROM instructions for procedures for using the STAT command.)

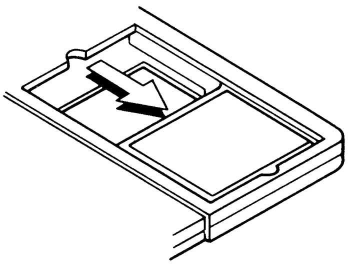
However, if you want to protect an entire disk a more satisfactory method is to use the write protect tab on a 3.5 inch disk, or with 5.25 inch disks to cover the write protect notch.

Once you have write protected a disk, you can retrieve data from it but you cannot add to, delete or rename any of the files. If the entire disk is protected, you cannot add any more files to the disk. Also, note that some application programs may not work properly with a write protected disk.

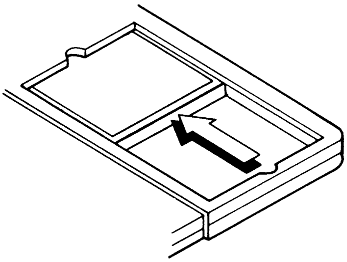
The write protect tab and notch on the disks are used as follows.

**3.5-INCH MICRO-FLOPPY DISK**

Slide the write protect tab toward the edge of the disk as shown below. When the hole is visible the disk is protected.

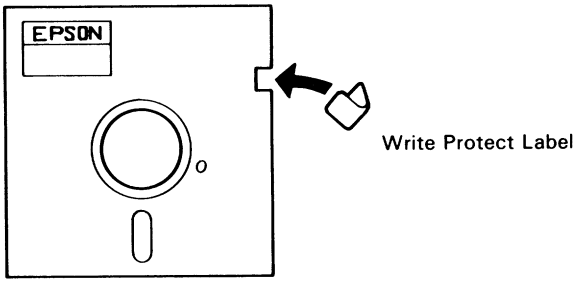


To allow the disk information to be changed again, slide the tab to cover the hole as shown.



## 5.25-INCH MINI-FLOPPY DISK,

Cover the rectangular notch on the right edge of the disk with an adhesive write-protect tab as shown below. Some of these tabs are included in the box in which disks are supplied.



Data can be written to the disk when this notch is exposed.

# Appendix B

# CP/M MESSAGES

Each utility supplied on the disk has its own set of messages. Appendix G in the Operating Manual lists all the CP/M messages alphabetically. The following Table lists only those messages which are generated by the utility programs provided on the disk.

## ASM

### 1. Source Program Error Messages

| Error message | Cause |
|---|---|
| D | Incorrect data expression in data area. |
| E | Incorrect expression which cannot be interpreted by assembler. |
| L | Incorrect use of label (duplicate label). |
| N | Use of non-implemented feature. |
| O | Expession too complicated. |
| P | Phase error; label does ot have the same value on a subsequent pass. |
| R | Improper register specification. |
| S | Syntax error. |
| U | Undefined label. |
| V | Value of operand or expression is out of range. |

### 2. Terminal Error Messages

**i) CANNOT CLOSE FILES**

An output file cannot be closed. This is a fatal error that terminates ASM execution. Check to see that the disk is in the drive, and that the disk is not write-protected.

**ii) NO DIRECTORY SPACE**

The disk directory is full. Erase some of the files to make room for. PRN and .HEX files. The director can usually hold only 64 file names.

## 3. NO FILE — (filename.ext)

The specified source or include file cannot be found on the indicated drive.

## 4. NO SOURCE FILE PRESENT

The assembler cannot find the specified file. You either mistyped the file name on the command line, or the file does not have the .ASM extension.

## 5. NOT FOUND

You specified a write-protected diskette as the destination for the PRN and HEX files, or the diskette is full. Correct the problem before assembling the program.

## 6. SOURCE FILE NAME ERROR

When you assemble a file, you cannot use wildcard characters (* and ?) in the filename. Only one file can be assembled at a time.

## 7. SOURCE FILE READ ERROR

The assembler cannot read the file containing the assembly language program. Part of another file may have been written over your assembly language file, or information was not properly saved on the diskette. Use the TYPE command to identify the error. Assembly language files contain the letters, symbols, and numbers that appear on your keyboard. If your screen displays unrecognizable characters or behaves strangely, the contents of the file have been corrupted.

## DDT

### 1. ?

This message has four possible meanings:

1) DDT does not understand your input.
2) The file cannot be opened. (Is it on the disk?)
3) A checksum error was found in a HEX file.
4) The file was loaded into the area containing the assembler/dis-assembler

### 2. hhhh??=dd

?? indicates that the hexadecimal value dd encountered at address hhhh cannot be represented in 8080 assembly language mnemonics.

### 3. Insufficient memory

There is not enough memory to load the file specified in an R command.

### 4. NO FILE — (filename.ext)

The file specified in an R command cannot be found on the disk.

## ED

### 1. Break "X" at C

The symbol "X" is one of the symbols listed below and C is the command letter being executed when the error occurred.

| | |
|---|---|
| # | Search failure. ED cannot find the string specified in an F, S, or N command. |
| ? | Unrecognizable command letter. ED does not recognizxe the indicated command letter, or an E,H,Q, or O command was included on the command line together with another command. |
| O | The file specified in an R command cannot be found. |
| > | Buffer full. ED cannot place any more characters in the memory buffer, or the string specified in an F,N, or S command is too long. |
| E | Command aborted. A keystroke at the console aborted command execution. |
| F | Disk or directory full. This error is followed by the full disk or directory message. Refer to the recovery procedures listed under these messages. |

### 2. Directory full

There is not enough directory space for the file being written to the destination disk. You can use the OX filespec command to erase any unnecesary files on the disk without leaving the editor.

### 3. Disk full

There is not enough disk space for the output file. This error can occur with the W, E, H, or X commands. If it occurs with the X command, you can repeat the command using a different drive.

### 4. FILE ERROR

Disk or directory is full and ED cannot write anything more on the disk. This is a fatal error, so be sure there is enough space on the disk to hold a second copy of the file before starting ED.

## 5. File exists, erase it

The file name specified for the destination file is already used for another file on the destination disk. Erase or rename the file or select another disk selected to receive the output file.

## 6. * * FILE IS READ/ONLY * *

The file specified when ED was started has the Read Only attribute. ED can read the file, but not cannot change.

## 7. File Not Found

ED cannot find the specified file. Make sure that you have specified the correct drive and that it contans the correct disk.

## 8. Filename required

You tried to start ED command without specifying a filename. Re-enter the ED command followed by the name of the file you want to edit or create.

## LOAD

1. **CHECKSUM ERROR**
   **LOAD ADDRESS hhhh**
   **ERROR ADDRESS hhhh**
   **BYTES READ:**
   **hhhh:**

The HEX file contains invalid data. Regenerate it with the ASM utility.

2. **ERROR: CANNOT OPEN SOURCE, LOAD ADDRESS hhhh**

Displayed if LOAD cannot find the specified file or if no filename is specified.

3. **ERROR: DISK READ, LOAD ADDRESS hhhh**

Results when an error code is returned by a BDOS function call.

4. **ERROR: DISK WRITE, LOAD ADDRESS hhhh**

The address of a record was too far from the address of the previously pressed record. This is an internal limitation of LOAD, but it can be circumvented. Use DDT to read the HEX file into memory, then use a SAVE command to store the memory image file to disk.

5. **ERROR: NO MORE DIRECTORY SPACE, LOAD ADDRESS hhhh.**

The disk directory is full.

6. **INVALID HEX DIGIT**
   **LOAD ADDRESS hhhh**
   **ERROR ADDRESS hhhh**
   **BYTES READ:**
   **hhhh**

The file contains an invalid hex digit.

# *Appendix C*

# *ERROR MESSAGES*

## Error Messages with the COPYDISK

**1. ! Use different drive name.**

This message will come up if you are copying with two drives and you specify the same drive for the original and the drive to copy to.

If it comes up when you only have one drive connected, press the ESC key to return to the option to choose only one drive.

If you are using two drives enter a different drive name for the destination drive.

**2. ! Read error**
**Press RETURN to retry**

This message will come up if there is a problem reading the disk.

Take the disk out of the drive and replace it, then press RETURN . If the problem still occurs it is most likely that the disk has become corrupt or you are trying to read an unformatted disk. Examine the disk on the CP/M command line with the DIR command.

**3. ! Write error**
**Press RETURN to retry**

This message will come up if there is a problem writing to the disk.

Try removing the disk and replacing it then pressing RETURN to retry.

This error can be caused if the disk has become corrupt or if you are using an unformatted disk or the surface of the disk has been damaged.

**4. ! Write protected**
**Press RETURN to retry**

This error will occur if you have write protected the disk and try to format it or copy to it. It may also occur if the disk is not correctly seated in the drive.

Remove the disk and examine the write protection. Make sure the disk is the correct way up in the drive. If all is well replace the disk and press the RETURN to retry.

**5. ! Disk copy fail**
**Press RETURN to retry**

This error will occur when the verification of the copy gives an error. Press the RETURN to retry.

**6. ! Disk not ready**
**Press RETURN to retry**

This can occur when the drives are not plugged in or switched on or the battery fails on the PF-10. It will also occur if the disk is in the drive of a TF-20 or TF-15 and the locking button is not pressed home.

**7. ! Communication error**
**Press RETURN to retry**

This error can occur if there is a problem in transmission of data to the drive.

Check that the cables are correctly in their sockets by removing and replacing them. Make sure that the computer, drives and cables are clear of all magnetic devices or anything else which could cause interference.

# INDEX

## EPSON OVERSEAS MARKETING LOCATIONS

**EPSON AMERICA, INC.**
2780 Lomita Blvd., Torrance, Calif. 90505,
U.S.A.
Phone: (213)539-9174
Telex: 182412

**EPSON DEUTSCHLAND GmbH**
Am Seestern 24, 4000 Düsseldorf 11,
F.R. Germany
Phone: (0211)5952-0
Telex: 8584786

**EPSON (UK) LTD.**
Dorland House, 388 High Road, Wembley,
Middlesex, HA9 6UH, U.K.
Phone: (01)902-8892
Telex: 8814169

**EPSON FRANCE S.A.**
55, rue Deguingand, 92300, Levallois-Perret,
France
Phone: (1)739-6770
Telex: 614202

**EPSON AUSTRALIA PTY. LTD.**
Unit 3, 17 Rodborough Road, Frenchs Forest,
NSW 2086, Australia
Phone: (02)452-5222
Telex: 75052

**EPSON ELECTRONICS (SINGAPORE) PTE. LTD.**
No. 1 Maritime Square, #02-19, World Trade
Centre Singapore 0409
Phone: 2786071/2
Telex: 39536

**EPSON ELECTRONICS TRADING LTD.**
Room 411, Tsimshatsui Centre, East Wing, 66,
Mody Road, Tsimshatsui Kowloon, Hong Kong
Phone: 3-694343/4; 3-7213427; 3-7214331/3
Telex: 34714

**EPSON ELECTRONICS TRADING LTD. (TAIWAN BRANCH)**
1, 8F K.Y. Wealthy Bldg. 206, Nanking, E. Road,
Sec. 2, Taipei, Taiwan R.O.C.
Phone: 536-4339; 536-3567
Telex: 24444

**EPSON CORPORATION**
80 Hirooka, Shiojiri-shi, Nagano 399-07
Japan
Phone: (0263)52-2552 Telex: 3342-214