APC™

CP/M-86™ User's Guide

NEC

APC-S01
158-030000-572

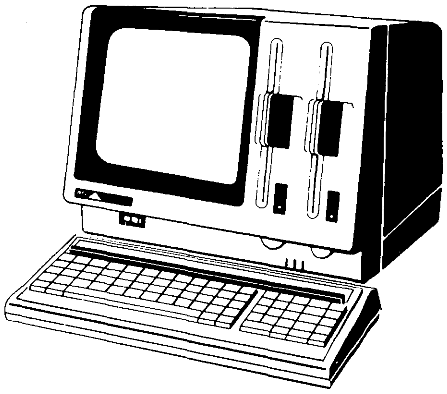
APC™

122183
010106AC860661
12090

NEC
NEC Information Systems, Inc.

158-030000-417

Advanced Personal Computer
TM



CP/M-86 User/Programmer's Guide

NEC
NEC Information Systems, Inc.
819-000100-4001 Rev. 01
8-83

Contents

Contents (cont'd)

Page

Contents (cont'd)

Page

Contents (cont'd)

Page

Contents (cont'd)

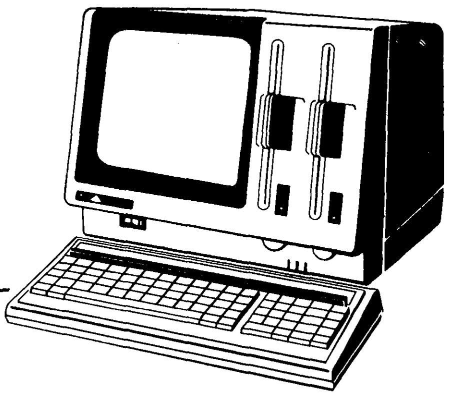
Page

Tables

Illustrations

Chapter 1

Introduction to CP/M-86



CP/M-86 means Control Program for the 8086 Microprocessor. It is an operating system for the NEC Advanced Personal Computer (APC). You can use CP/M-86 with a single diskette drive APC, a dual diskette drive APC, or an APC with either one or two floppy diskette drives and one or two hard disk units.

CP/M-86 for the APC is distributed on two eight-inch floppy diskettes. One disk consists of the standard CP/M-86 files and programs customized for the APC, including the CP/M-86 operating system program itself. Throughout this manual, this diskette is referred to as the CP/M-86 system diskette, or simply the CP/M-86 diskette. The second distribution diskette is referred to as the Program Development Aids diskette. You will probably not use this diskette unless you are either a systems programmer or a user of the graphics extension for CP/M-86.

CP/M-86 manages information stored magnetically on diskettes and hard disks by grouping this information into files of programs or data. CP/M-86 can copy files from a hard disk to a floppy diskette and vice versa, from either type of disk medium to your computer's memory and back, or to a peripheral device such as a printer. CP/M-86 performs these and other tasks by executing various programs according to commands you enter at the keyboard.

Once loaded into memory under the direction of CP/M-86, a program runs through a set of steps that instructs the APC to perform certain tasks. You can use CP/M-86 to create your own CP/M-86 programs, or you can choose from the wide variety of CP/M-86 application programs that entertain, educate, and solve commercial and scientific problems.

This chapter discusses the fundamentals of CP/M-86 with the APC. It describes CP/M-86 startup procedures and initial messages. Then it explains how to enter a CP/M-86 command and make backup copies of your CP/M-86 distribution diskettes.

HOW TO GET CP/M-86 STARTED

Starting or loading CP/M-86 means reading a copy of CP/M-86 from your CP/M-86 system diskette into the memory of the APC. No matter what hardware configuration you have or which drive holds the CP/M-86 system files, you must load CP/M-86 into the APC's memory from a floppy diskette every time you start up the system.

To load CP/M-86, turn on the power and insert the CP/M-86 system diskette into Drive A. Close the drive door. This automatically loads CP/M-86 into memory.

If power is already on and you want to restart CP/M-86, first make sure the CP/M-86 system diskette is in Drive A. Then press the following keys simultaneously: FNC, CTRL, and BREAK STOP. This is called system reset, or booting the system.

At system reset, CP/M-86 is loaded into memory. The first thing CP/M-86 does is display the following message on the screen.

```
NEC Advanced Personal Computer CP/M-86
   Version V.V
Copyright [C], Digital Research Inc.
```

The version number, represented above by V.V, tells you the major and minor revision level of the CP/M-86 version that you own. This display is followed by two lines of messages.

```
A>SUBMIT AUTSTRT
A>
```

The first line names the autostart file on the diskette. This is the file called by the SUBMIT program to automatically load CP/M-86 when you boot the system.

The A> symbol is the CP/M-86 *system prompt*. The system prompt tells you that CP/M-86 is ready to read a command from the keyboard. It also tells you that diskette drive A is the *default drive*. This means that until you tell CP/M-86 to do otherwise, it looks for all program and data files on the diskette in Drive A.

CP/M-86 on the APC supports up to four diskette drives, labelled A through D, and four logical hard disk drives. The hard disk drives are configured as two logical partitions per physical unit. The logical devices are labelled Drive E and Drive F for physical unit 0 (APC-H26) and Drives G and H for unit 1 (APC-H27).

THE COMMAND LINE

CP/M-86 performs certain tasks according to specific commands that you enter from the keyboard. A CP/M-86 command line is composed of a command keyword, an optional command tail, and a carriage return keystroke. On the APC, the carriage return key is labeled RETURN. The ENTER key on the numeric keypad performs the same function.

The command keyword identifies a command (program) to be executed by the microprocessor. The command tail can contain extra information for the command such as a filename, option, or parameter. To end the command line, you must press RETURN or ENTER.

As you type characters at the keyboard, they appear on your screen and the *cursor*, or position indicator, moves to the right. If you make a typing mistake, press BACK SPACE to move the cursor to the left and correct the error.

You can type the keyword and command tail in any combination of uppercase and lowercase letters. CP/M-86 treats all letters in the command line as uppercase.

Generally, you type a command line directly after the system prompt. However, CP/M-86 does allow spaces between the prompt and the command keyword.

A command keyword identifies one of two different type of commands: Built-in commands and Transient Utility commands. Built-in commands reside in memory as part of CP/M-86 and can be executed immediately. Transient Utility commands are stored as program files. They must be loaded into memory to perform their task. You can recognize Transient Utility program files in a disk directory because their filenames end with CMD.

For Transient Utilities, CP/M-86 checks only the command keyword. If you include a command tail, CP/M-86 passes it to the utility without checking it because many utilities require unique command tails.

One of the Built-in commands, DIR, tells CP/M-86 to display the names of all files on a disk or diskette. To demonstrate how CP/M-86 reads command lines, type the keyword DIR after the system prompt, omit the command tail, and press RETURN.

```
    A>DIR
```

CP/M-86 responds to this command by displaying the names of all the files that are stored on the diskette in Drive A. For example, if you have your CP/M-86 system diskette in Drive A, these filenames, among others, appear on the screen.

```
COPYDISK CMD
PIP      CMD
STAT     CMD
```

CP/M-86 recognizes only correctly spelled command keywords. If you make a typing error and press RETURN before correcting your mistake, CP/M-86 echoes the command line with a question mark at the end. For example, if you accidentally mistype the DIR command, CP/M-86 responds

```
A>DJR
DJR?
```

to tell you that it cannot find the command keyword.

The DIR command can accept a filename as a command tail. You can use DIR with a filename to see if a specific file is on the disk. For example, to check that the Transient Utility program COPYDISK.CMD is on your system diskette, enter:

```
A>DIR COPYDISK.CMD
```

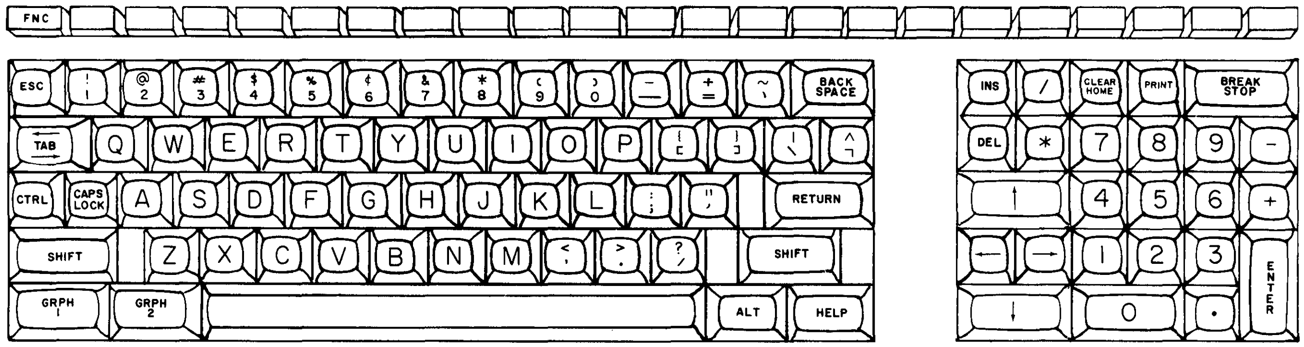
CP/M-86 displays either the name of the file you specified or the message NO FILE.

Be sure to type at least one space after DIR to separate the command keyword from the command tail. If you don't, CP/M-86 responds as shown below.

```
A>DIRCOPYDISK.CMD
DIRCOPYDISK.CMD?
```

THE APC KEYBOARD

The APC keyboard, shown below, provides a powerful complement to CP/M-86.



The keyboard includes the following features that can greatly enhance and expand application programs:

- Programmable Function Keys — Using the KEY Utility provided with CP/M-86 , you can define up to 32 different operations to be performed by the function keys at the top of the keyboard. Any number of these key definition sets can be stored in disk files and used where applicable.
- Repeat Function — Instead of repeatedly pressing a particular key, you can enter a string of characters by holding down the key. The character is repeated until you release the key. (Note, however, that this function is disabled when the TIMEROFF Utility program is run.)
- Key Click — It is possible to turn on or off the key clicker. The clicker produces a brief tone as each key is pressed. A knob on the APC processor unit controls the volume of the tone. Note that if you hold down a key, the click does not repeat.

CP/M-86 also provides a *fifo buffer* for the APC keyboard. This buffer recognizes and remembers the keys pressed in first-in, first-out sequence. This allows you to type up to 64 characters ahead while other processing occurs.

In most cases, the keys on the APC keyboard perform as they are labeled. Some keys, such as HELPand INSgenerate special codes that can be trapped and used in application programs to perform special routines. The function keys, PF1-PF22, can be dynamically assigned values or functions in each application. The ALTkey is a special latching key that allows application programs to trap and take advantage of additional character codes from the keyboard. Still other keys, such as those listed in the following table, automatically perform the special functions described.

Table 1-1 Special Key Functions

KEY	FUNCTION
DEL	Deletes all the characters in a command line and moves the cursor back to the beginning of the line.
CLEAR HOME	Moves the cursor to the home position on the screen.
PRINT	Puts the printer online to echo all console activity; pressing again takes the printer offline.
BREAK STOP	Temporarily stops console listing; pressing again resumes the listing.
BACK SPACE	Moves the cursor back one space, erasing the previous character.
← TAB →	Moves the cursor ahead or backwards one tab stop.

The keys on the keyboard and keypad can be used in combinations that perform unique functions. Table 1-2 describes some of these special keyboard functions. The functions are activated when the mode key and the effective key or keys are pressed simultaneously.

Table 1-2 More Special Keyboard Functions

MODE KEY	EFFECTIVE KEYS	FUNCTION OR ACTION
CTRL	0-9 on the typewriter keyboard	Controls the speed at which information is displayed on the screen. CTRL-0 is the slowest speed; CTRL-9 is the fastest.
	↑ ↓	Scrolls the display up and down.
	A-Z	Generates the ASCII control codes shown in Tables 1-3 and 1-4.

Table 1-2 More Special Keyboard Functions (cont'd)

MODE KEY	EFFECTIVE KEYS	FUNCTION OR ACTION
FNC	CTRL + BREAK STOP	Pressed simultaneously, these three keys reboot CP/M-86. Never use this combination while disk writes are in progress as it may cause data loss.
	PF1-PF16	Provides additional function key use.
CAPS LOCK	A-Z on the typewriter keyboard	Allows typing of uppercase letters only. Note that neither the numbers on the typewriter keyboard nor those on the keypad are affected.
SHIFT	All keys on the typewriter keyboard	Types the top symbol on each key pressed.
	CLEAR HOME	Clears the screen.
	BREAK STOP	Aborts the program currently running.
GRPH 1 GRPH 2	All typewriter keys	Used with and without the SHIFT key, these latching keys allow access to the graphic symbols and characters.

CP/M-86 CONTROL CHARACTER COMMANDS

You can correct typing mistakes by pressing BACK SPACE. However, CP/M-86 also supports the control character commands listed in Table 1-3 to help you edit more efficiently. You can use these control characters to edit command lines or input lines to most programs. To enter a control character, hold down CTRL and simultaneously press the required letter key. Then release both keys.

Table 1-3 Control Character Commands

COMMAND	MEANING	EQUIVALENT APC KEY
CTRL-E	Moves the cursor to the beginning of the following line without erasing your previous input.	
CTRL-H	Moves the cursor left one character position and deletes the character.	BACK SPACE
CTRL-I	Moves the cursor to the next tab stop, where tab stops are automatically placed at each eighth column.	TAB
CTRL-J	Moves the cursor to the left of the current line and sends the command line to CP/M-86.	RETURN
CTRL-M	Moves the cursor to the left of the current line and sends the command line to CP/M-86.	RETURN
CTRL-R	Types a # at the current cursor location, moves the cursor to the next line and retypes any partial command you have typed so far.	
CTRL-U	Discards all the characters in the command line that you have typed so far, types a # at the current cursor position and moves the cursor to the next command line.	
CTRL-X	Discards all the characters in the command line that you have typed so far and moves the cursor back to the beginning of the current line.	

Notice that some control characters have similar meanings. For example, CTRL-J CTRL-M, and RETURN all send the command line to CP/M-86 for processing. Also, CTRL-H has the same effect as BACK SPACE.

Table 1-4 lists additional CP/M-86 control characters used often during editing and processing.

Table 1-4 Additional CP/M-86 Control Character Commands

COMMAND	MEANING	EQUIVALENT APC KEY
CTRL-C	Aborts the program currently running.	SHIFT BREAK STOP
CTRL-P	Echoes all console activity at the printer. Pressing again ends printer echo.	PRINT
CTRL-S	Temporarily stops console listing. Pressing again resumes the listing.	BREAK STOP
CTRL-Z	Separates fields or strings.	

THE APC STATUS LINE

The first line on the APC display screen is the status line shown below.

```
CP/M-86-1.1(1.106:013) CAP ALT GR1 GR2 sp:9 Fri 08/06/82 09:24
```

The left side of this line, through the parentheses, provides information about the version of CP/M-86 in use. The remainder of the status line provides information about the following:

- Special latching keys in use
- Display speed
- System time and date

If any of the special latching keys on the APC keyboard — CAPS LOCK, ALT, GRPH1, and GRPH2 — is active, the status line displays the key name in reverse video.

The current screen display speed, set using CTRL and the numbers on the alphanumeric keyboard, is also given. In the example above, the display speed is represented by sp:9.

The system time, date, and day of the week, set using the TOD Utility program, are also displayed.

WHY YOU SHOULD BACK UP YOUR FILES

Humans have faults, and so do computers. Human or computer errors sometimes destroy valuable programs or data files. By mistyping a command, for example, you could accidentally erase a program that you just created. A similar disaster could result from an electronic component failure or software error.

Data processing professionals avoid losing programs and data by making copies of valuable files. Always make a working copy of any new program you purchase and save the original. If the program is accidentally erased from the working copy, you can easily restore it from the original.

Professionals also make frequent copies of new programs or data files during the time they are being developed. The frequency of making copies varies with each programmer, but as a general rule, make a copy at the point where it takes ten to twenty times longer to reenter the information than it takes to make the copy.

You can make backups in two ways. You can back up files one at a time, or you can make a complete copy of the entire disk or diskette. The choice is usually made based on the number of files on the disk that need to be backed up. It might take less than a minute to make a copy of one file, but it takes two or three minutes to copy an entire diskette. Backing up a hard disk unit takes even longer — 20-40 minutes per partition. However, it is well worth the time spent to avoid disastrous loss of data.

HOW TO MAKE A BACKUP COPY OF YOUR CP/M-86 DISKETTES

Whether you are using a floppy diskette APC system or an APC with hard disk, it is a good idea to make a backup copy of the CP/M-86 diskettes. These diskettes are called the *working copies* and should be used regularly to perform CP/M-86 functions. The *original copy* refers to the distribution diskettes you purchased. They should be kept in a safe place and used only to make new working copies.

The backup diskettes can be either factory-fresh or used. Some eight-inch diskettes come with a notch cut out of the lower right hand side. This notch prevents data from being written to the disk. It is called a *write-protect* notch. To copy data to these diskettes, you must *write-enable* them by placing a small foil tab over the write-protect notch. These tabs are supplied with the diskettes.

You must format new diskettes with the Transient Utility program named FORMAT that comes on your CP/M-86 system diskette. You can also reformat used diskettes with FORMAT, although this is not required. If you are reusing diskettes, make sure than do not contain any information you might need again! FORMAT destroys everything that is on the diskette. COPYDISK copies everything from a source diskette to a destination diskette — including blank space —and writes over any information that might already be stored on the destination diskette.

Formatting a Diskette

To format a diskette, follow this procedure.

Turn on the APC. Insert the CP/M-86 operating system diskette into Drive A. Always insert a diskette so that the label faces toward the screen. The label should go in last. Close the door to the diskette drive.

You will hear the diskette spinning in the drive. The following prompt is displayed.

```
A>
```

Type FORMAT and press RETURN. You may use either uppercase or lowercase characters. When the FORMAT program is loaded, the following prompt appears.

```
*** VOLUME INITIALIZER V1.1 (CP/M-86) ***
    INITIALIZE DEVICE[A,B,C,D] :
```

Press the letter of the drive into which you are inserting the diskette to be formatted.

- If you have a dual-drive APC, insert the diskette to be formatted into Drive B and press B.
- If you have a single-drive APC, remove the CP/M-86 system diskette by squeezing the diskette drive door latch. (The spring-lock mechanism releases the diskette as the drive door opens.) Then, insert the diskette to be formatted and press A.

A prompt similar to the following prompt appears.

```
MEDIA TYPE FD-2D
READY? [R:READY A:ABORT] :
```

The abbreviation following "MEDIA TYPE" identifies the type of diskette you inserted. FD-1D indicates a single-sided, single-density diskette. FD-2D indicates a double-sided, double-density diskette.

The "READY? [R:READY A:ABORT]" prompt allows you to review your actions before proceeding.

If you made a mistake or you want to change your entry and return to the "INITIALIZE DEVICE [A,B,C,D]" prompt, press A. If you are ready to continue the formatting process, press R. As formatting occurs, the number of the cylinder being formatted appears in brackets as shown below.

```
REFORMATTING CYLINDER NO.[00]
```

If a damaged cylinder is detected during formatting, the following prompt displays the number of the damaged cylinder.

```
* BAD CYLINDER NO. [47]
[R:RETRY A:ABORT] :
```

Press R to try to format the cylinder again. If the damaged cylinder cannot be formatted, press A to abort the procedure. The program returns to the "INITIALIZE DEVICE [A,B,C,D]" prompt. Remove the diskette and label it damaged. Insert another diskette in the appropriate drive and try again.

When all cylinders on a diskette are successfully formatted, the following prompt appears.

```
* NORMAL END
[A:AGAIN E:END] :
```

Remove the formatted diskette by squeezing the diskette drive door latch and then releasing it. The spring-lock mechanism releases the diskette as the drive door opens.

To repeat the process, press A. To end the program, press E. TheCP/M-86 "A>" prompt reappears.

Copying a Diskette

To make a copy of the CP/M-86 system diskette, use the COPYDISK Utility. It physically copies all information from one diskette to another.

Insert the CP/M-86 operating system diskette into Drive A. Type COPYDISK and press RETURN. When the program is loaded, the following prompt appears.

```
 Full Disk Copy/Verify Utility V1.1
Copy or Verify or Copy & Verify (C,V,B)?
```

Select one of the three options:

C to copy the contents of one diskette to another;

V to verify the contents of one diskette with another;

B to both copy and verify a diskette.

When copying diskettes, it is advisable to also verify the information. When you have typed your response, the following prompt appears.

```
Source Disk Drive (A-D) ?
```

Press A to indicate that you are copying from that drive. The following prompt appears.

```
Destination Disk Drive (A-D) ?
```

Identify the diskette drive you are copying to.

- If you have a dual-drive APC, insert a formatted diskette into Drive B. Press B at the prompt to indicate you are copying to that drive.
- If you have a single-drive APC, press A in response to the prompt.

The following prompt names both the source and destination drives to allow you to review your entries.

```
Copying Disk A: To Disk B:
Is This What You Want To Do (Y/N)?
```

If you made a mistake or want to change the entries, press N in response to the prompt. If you are ready to continue, press Y. What happens next depends on the type of APC you have.

- If you have a dual-drive APC the following display appears as each cylinder of the diskette in Drive A is copied to the diskette in Drive B.

```
Copy Started
  Reading Cylinder [00]
  Writing Cylinder [00]
```

- If you have a single-drive APC, the COPYDISK program reads a portion of the source diskette, then prompts you to remove the source diskette and insert the destination diskette. The program then writes to the destination diskette and prompts you to reinsert the source diskette. This procedure is repeated until the entire source diskette is copied. The system displays the following series of messages.

```
Copy Started
  Reading Cylinder [00]
Insert "DESTINATION" Diskette In Drive A:
Strike Any Key When Ready
  Writing Cylinder [00]
Insert "SOURCE" Diskette In Drive A:
Strike Any Key When Ready
```

When copying is completed, the COPYDISK program verifies the contents of each diskette to insure that an exact duplicate was made. This procedure also depends on the type of APC you have.

- If you have a dual-drive APC, the following display appears as the corresponding cylinders on the diskettes are verified.

```
Verify Started
  Reading Cylinder [00]
  Writing Cylinder [00]
```

- If you have a single-drive APC, you must insert and remove the source and destination diskettes when prompted as the information on each diskette is verified. The procedure, represented by the following series of messages, is repeated until verification is completed.

```
Verify Started
  Reading Cylinder [00]
Insert "DESTINATION" Diskette In Drive A:
Strike Any Key When Ready
  Verifying Cylinder [00]
Insert "SOURCE" Diskette In Drive A:
Strike Any Key When Ready
```

When verification is completed, the following prompt appears.

```
Copy/Verify Another Disk (Y,N) ?
```

To repeat the process with the CP/M-86 Program Development Aids diskette, press Y. This time, whether you are using a single-drive or dual-drive APC, remove the CP/M-86 system diskette from Drive A and replace it with the original CP/M-86 Program Development Aids diskette before you identify the source drive to the program.

When you have finished making backup copies of your diskettes, press N. The program returns to the CP/M-86 "A>" prompt.

Use the working copies as your CP/M-86 diskettes to make more backups, to try the examples shown in the rest of this manual, and to start CP/M-86 the next time you power up your APC.

WHAT'S ON YOUR CP/M-86 DISKETTES?

Taking a directory of your CP/M-86 distribution disk reveals a number of files and programs. Most of the programs are standard Transient Utilities; others are programs provided to demonstrate certain features of the APC.

The following tables show alphabetical listings of the contents of the CP/M-86 diskettes. A brief description of the file or program is given.

Table 1-5 Contents of Distribution Diskette 1

FILE OR PROGRAM	DESCRIPTION
ASM86.CMD	CP/M-86 Assembler Utility program. Translates 8086 assembly language into hexadecimal machine code format.
ASSIGN.SYS	Used by the graphics extension to assign device drivers.
AUTSTRT.SUB	File containing autostart program submit instructions. When CP/M-86 loads, the contents of this file are examined. Any command instructions in this file are automatically loaded and executed. The file is empty initially.
AUXCG.CHR	Auxiliary character data file.
BACH.CMD	Demonstration program showing the musical capability of the APC.
CHR.CMD	Auxiliary Character Generator Utility program.
CONFIG.CMD	Hardware Configuration Utility program.
COPYDISK.CMD	Full Disk Copy Utility program.
CPM.SYS	The CP/M-86 system file.
DDGN2A.SYS	Device driver for the Lear Siegler ADM5. Used by the graphics extension.
DDGN2B.SYS	Device driver for the ADDS Viewpoint. Used by the graphics extension.
DDGN2C.SYS	Device driver for the Televideo 910. Used by the graphics extension.
DDGN2D.SYS	Device driver for the Datamedia Colorscan-10. Used by the graphics extension.

Table 1-5 Contents of Distribution Diskette 1 (cont'd)

FILE OR PROGRAM	DESCRIPTION
DDHI3M.SYS	Device driver for the Houston Instruments Hiplot DMP-3/4-443 Multipen Plotter. Used by the graphics extension.
DDHI7M.SYS	Device driver for the Houston Instruments Hiplot DMP-6/7 Multipen Plotter. Used by the graphics extension.
DDIDS.SYS	Device driver for the Integral Data Systems Monochrome Printers: Micro Prism 480, Prism 80, and Prism 132. Used by the graphics extension.
DDIDSC.SYS	Device driver for the Integral Data Systems Color Printers: Prism 80 and Prism 132. Used by the graphics extension.
DDPMPV.SYS	Device driver for the Printronix MVP printer. Used by the graphics extension.
DDMX80.SYS	Device driver for the Epson MX-80 with Graftrax Plus. Used by the graphics extension.
DDNECAPC.SYS	Device driver for the NEC APC. Used by the graphics extension.
DDOKID.SYS	Device drive for the Okidata Microline 92 printer. Used by the graphics extension.
DDPRTX.SYS	Device driver for the Printronix P300 and P600 printers. Used by the graphics extension.
DDSTRB.SYS	Device driver for the Strobe Model 100 Graphics Plotter. Used by the graphics extension.

Table 1-5 Contents of Distribution Diskette 1 (cont'd)

FILE OR PROGRAM	DESCRIPTION
DDT86.CMD	CP/M-86 Dynamic Debugging Tool Utility program. Used for interactive 8086 program debugging.
DDVRET.SYS	Device driver for the VT100. Used by the graphics extension.
DD7220.SYS	Device driver for the Hewlett-Packard 7220 Graphics Plotter. Used by the graphics extension.
DD7470.SYS	Device driver for the Hewlett-Packard 7220 Graphics Plotter. Used by the graphics extension.
DISP.CMD	Auxiliary Character Display Utility program. Used to display the currently loaded auxiliary character set.
DISP1.CMD	ROM Character Display Utility program.
ED.CMD	CP/M-86 Text Editor Utility program.
FORMAT.CMD	CP/M-86 Floppy Diskette Formatting Utility program. Analyzes and prepares diskettes to accept CP/M-86 formatted files and programs.
GENCMD.CMD	CP/M-86 Command File Generator Utility program. Translates assembled hexidecimal machine code into executable command file format.
GRAPHICS.CMD	Graphics Utility program.
HDBACKUP.CMD	Hard Disk Backup Utility program. Used to backup from hard disk to diskette.

Table 1-5 Contents of Distribution Diskette 1 (cont'd)

FILE OR PROGRAM	DESCRIPTION
HDFORMAT.CMD	Hard Disk Format Utility program. Analyzes and prepares one or both of the logical drive devices for each hard disk to accept CP/M-86 formatted files and programs. This utility program must be run before the first time CP/M-86 attempts file-related operation to a hard disk unit.
HELP.CMD	Utility program providing information about how to use each CP/M-86 command.
HELP.HLP	Text file containing the information accessed during the execution of HELP.CMD. File contents can be customized.
KEY.CMD	Function Key Assignment Utility program.
LDCOPY.CMD	Utility program used to move the CP/M-86 system loader to other formatted disks.
LOADER.CMD	File containing instructions necessary for CP/M-86 system loading. The file is automatically accessed during LDCOPY. Not an executable program.
PIP.CMD	Peripherals Interchange Utility program. Used to transfer information between logical devices. In a single-drive system, PIP does not function for file copying operations when a diskette change is necessary. PIP1 must be used for file copying operations in a single-diskette drive system.
PIP1.CMD	Peripherals Interchange Utility program for a single diskette drive system. Program allows single and multiple file transfers from one diskette to another in the same drive. PIP1 copies files only from Drive A to Drive A.

Table 1-5 Contents of Distribution Diskette 1 (cont.)

FILE OR PROGRAM	DESCRIPTION
POW.CMD	Automatic Power Off Utility program. Used to turn off the APC by software control.
SETCOM.CMD	Set Communications Utility program. Used to set the operating parameters of the serial interface of the APC.
STAT.CMD	Status Utility program. Used to examine and alter the status of files, disks, and logical device assignments.
SUBMIT.CMD	Submit (Batch Processing) Utility program. Sends files containing batched command instructions and parameters to CP/M-86 for processing. Files processed (for example, the AUTSTRT.SUB file) are identified by the SUB filetype.
TIMEROFF.CMD	Utility program used when running any communications software that supports interrupts. It turns off the dynamic status line update and display from the APC clock/calendar. The correct time is restored by TIMERON.CMD.
TIMERON.CMD	Utility program used after running any communications software that supports interrupts. It restores the update and display of the correct time to the APC status line.
TOD.CMD	Time of Day Utility program. Used to set the time of day, day of the week, and date into the APC hardware clock.
VISUAL.CMD	Demonstration program showing the APC character set.

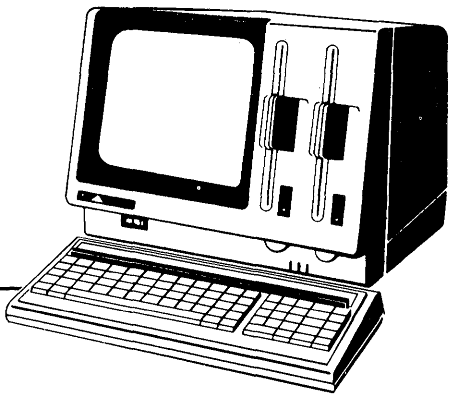
Table 1-6 Contents of Distribution Diskette 2

FILE OR PROGRAM	DESCRIPTION
CBIOS.LST	Source listing of the CBIOS for the APC.
DDNECLIB.LST	Source listing of a library used by the graphics extension for device drivers with the NEC APC.
DDNECAPC.LST	Source listing of the NEC APC device driver used by the graphics extension.
NECASCII.LST	Source listing of ASCII codes that are used in graphics for the APC.

In addition to the files and command programs listed in Tables 1-5 and 1-6, the CP/M-86 operating system provides six Built-in commands that do not appear on the diskette directory. The following table lists the Built-in commands on the CP/M-86 system diskette. These commands reside in memory after CP/M-86 has been loaded. They are accessed at the command line in the same manner as command files.

Table 1-7 Built-in Commands

COMMAND	FUNCTION
DIR	Displays a list of filenames from the disk directory. Does not display system files.
DIRS	Displays a list of system files from the disk directory.
ERA	Erases a filename from a disk directory and releases the area occupied by the file.
REN	Renames a file within a disk directory.
TYPE	Displays on the screen the contents of a file. Generally used for text files.
USER	Changes the currently active user area number.



Chapter 2

Files, Disks, Drives, and Devices

CP/M-86's most important task is to access and maintain files on your storage devices — hard disks and/or floppy diskettes. It can create, read, write, copy, and erase program and data files. This chapter tells you what a file is, how to create, name, and access a file, and how files are stored. It also tells how to indicate to CP/M-86 that you have changed diskettes or that you want to change your default drive.

WHAT IS A FILE?

A CP/M-86 file is a collection of related information stored on a disk or diskette. Every file must have a unique name because that name is used to access the file. A directory is also stored on each disk and diskette. The directory contains a list of the filenames stored on that disk and the location of each file.

In general, there are two kinds of files: program files and data files. A program file is an executable file, that is, a series of instructions the APC can follow step by step. A data file is usually a collection of information: a list of names and addresses, the inventory of a store, the accounting records of a business, the text of a document, or similar related information. For example, the APC cannot "execute" names and addresses, but it can execute a program that reads names and addresses from a data file and prints them on mailing labels.

A data file can also contain the source code for a program. Generally, a program source file must be processed by an assembler or compiler before it becomes an executable program file. In most cases, an executing program processes a data file. However, there are times when an executing program processes other executable program files. For example, the executable copy program PIP can copy one or more command program files.

HOW ARE FILES CREATED?

There are many ways to create a file. You can create a file by copying an existing file to a new location, perhaps renaming it in the process. Under CP/M-86, you can use the Transient Utility program PIP (or PIP1 with a single drive APC) to copy and rename files. The second way to create a file is to use a text editor. The CP/M-86 text editor ED can create a file and assign it the name you specify. Finally, some programs, such as ASM-86, create output files as they process input files.

NAMING FILES — WHAT'S IN A NAME?

CP/M-86 identifies every file by its unique file specification. A file specification can have three parts:

● *d:*	drive specifier	one character	optional
● *filename*	filename	1-8 characters	
● *typ*	filetype	0-3 characters	optional

It is recommended that you create file specifications from letters and numbers. Because the CP/M-86 command processor recognizes the following special characters as delimiters (separators), they must not be included within a filename or filetype.

```
< > . , ; : = ? * [ ]
```

A file specification can be simply a one to eight character filename, such as:

```
MYFILE
```

When you make up a filename, try to let the name indicate what the file contains. For example, if you have a list of customer names for your business, you could name the file

```
CUSTOMER
```

so that the name is eight or fewer characters but still gives you some idea of what is in the file.

As you begin to use CP/M-86 with the APC, you'll find that files fall naturally into families. To keep file families separated, CP/M-86 allows you to add an optional one-to-three-character family name, called a filetype, to the filename. When you add a filetype to the filename, separate the filetype from the filename with a period. Try to use three letters that tell something about the file's family. For example, you could add the following filetype to the file that contains a list of customer names:

```
CUSTOMER.NAM
```

When CP/M-86 displays file specifications in response to a DIR command, it fills in filenames shorter than eight characters and filetypes shorter than three characters with blanks so that you can compare filetypes quickly. For example, in response to the DIR command, CP/M-86 might display the following:

```
PIP     .CMD     COPYDISK.CMD
LDCOPY  .CMD     WP      .INT
FIRSTDRA.FT      TEST    .INT
LETTERTO.DAD     FORMAT  .CMD
```

Like data files, the executable program files that CP/M-86 loads into memory from a disk or diskette all have different filenames. However, because executable programs are all in the 8086 family of programs that run with CP/M-86, they are all identified by the filetype CMD.

CP/M-86 recognizes other already established file families. The following table describes some of these standard filetypes recognized on the APC.

Table 2-1 CP/M-86 Filetypes

FILETYPE	MEANING
CMD	8086 or 8088 machine language program
BAS	CBASIC source program
$$$	Temporary file
A86	ASM-86 source file
H86	Assembled ASM-86 program in hexadecimal format
SUB	List of commands to be executed by SUBMIT
SYS	CP/M-86 system file
KEY	Data file containing APC function key assignments
CHR	Auxiliary APC character set file

ACCESSING FILES — DO YOU HAVE THE CORRECT DRIVE?

When you type a file specification in a command tail, the Built-in or Transient Utility looks for the file on the disk or diskette in the drive named by the system prompt. For example, if you type the command

```
A>dir copydisk.cmd
```

CP/M-86 looks in the directory of the disk in Drive A for COPYDISK.CMD. But if you have another drive, B for example, you need a way to tell CP/M-86 to access the disk in Drive B instead. For this reason, CP/M-86 lets you preceed a filename with a drive specifier and a colon to identify the drive location of the file. For example, in response to the command

```
A>dir b:myfile.lib
```

CP/M-86 looks for the file MYFILE.LIB in the directory of the diskette in Drive B. The APC recognizes drives A-D for floppy diskettes, drives E and F for hard disk unit 0, and drives G and H for hard disk unit 1.

You can also preceed an executable program filename with a drive specifier, even if you are using the program filename as a command keyword. For example, if you type the following command

```
A>b:pip
```

CP/M-86 looks in the directory of the disk in Drive B for the file PIP. If CP/M-86 finds PIP on Drive B, it loads PIP into memory and executes it.

Unlike the filename and filetype that are stored in the disk directory, the drive specifier for a file changes as you move a diskette from one drive to another or copy a file from one device to another. Therefore, a file has a different file specification when you change its location from one drive to another.

ACCESSING MORE THAN ONE FILE

Certain CP/M-86 Built-in and Transient Utilities can select and process several files when special *wildcard* characters are included in the filename or filetype. A file specification containing wildcards can refer to more than one file because it gives CP/M-86 a pattern to match: CP/M-86 searches the diskette or disk directory and selects any qualifying file whose filename or filetype matches the pattern.

The two wildcard characters are ?, which matches any single character in the same position, and *, which matches any character at that position, as well as any other characters remaining in the filename or filetype. The rules for using wildcards are listed below. A ? matches any character in a name, including a space character.

- A * must be the last, or only, character in the filename or filetype. CP/M-86 internally replaces a * with ? characters to the end of the filename or filetype.
- When the filename being matched is shorter than eight characters. CP/M-86 treats the name as if it ends with spaces.
- When the filetype being matched is shorter than three characters, CP/M-86 treats the filetype as if it ends with spaces.

Suppose, for example, you have a disk with the following six files:

A.CMD, AA.CMD, AAA.CMD, B.CMD, A.A86, and B.A86

Several cases are listed below where a name with wildcards matches all, or a few of these files:

????????.???	matches all six names listed above
.	is treated as ????????.???
????????.CMD	matches the first four names
*.CMD	is treated as ????????.CMD
?.CMD	matches A.CMD and B.CMD
?.???	matches A.CMD, B.CMD, A.A86, and B.A86
?.*	is treated as ?.???
A?.CMD	matches A.CMD and AA.CMD
A???????.CMD	matches A.CMD, AA.CMD, and AAA.CMD
A*.CMD	is treated as A???????.CMD

Remember that because CP/M-86 can use wildcard patterns only to search a directory, wildcards are valid only in filenames and filetypes. You cannot use a wildcard in a drive specifier.

HOW CAN I ORGANIZE AND PROTECT MY FILES?

Under CP/M-86 you can organize your files into groups, protect your files from accidental change, and specify how your files are displayed in response to a DIR command. CP/M-86 supports these features by assigning user numbers and attributes to files and recording them in the diskette or disk's directory.

User Area Numbers

CP/M-86 provides 16 distinct user directory areas with which to further separate and distinguish files. These user areas are identified by the numbers 0-15. Generally, when you load CP/M-86, user area 0 is active. By using the Built-in command USER to display and change the current user number, you can assign user area numbers to your files to separate them into as many as 16 distinct file groups. CP/M-86 assigns the currently active user number to a file when the file is created. Most commands access only those files that have the current user number. For example, if the current user number is 7, the DIR command displays only the files that were created under user number 7. However, with the [G*n*] option of the PIP Transient Utility program, you can copy files from one user area into another.

File Attributes

File attributes control how a file can be accessed. These attributes represent transparent labels added to the file that provide additional information about how the file is processed. There are two kinds of attributes that control file accessing.

DIR/SYS ATTRIBUTE

The first type of attribute is related to how the filename is stored in the directory. This is the DIR/SYS attribute. When you create a file, it is automatically marked with the DIR attribute. All files having the DIR attribute are displayed in response to the Built-in DIR command. However, if you do not want a filename to appear in a directory listing, you can assign the file the SYS attribute. Files having the SYS attribute are considered *system files* and are not displayed in response to the DIR command. You can use the STAT Transient Utility command to assign the SYS attribute to a file or change its attribute back to DIR.

It is very useful to assign the SYS attribute to those command files (that is, those with a filetype of CMD) that are in user area number 0. If you give a command file in user number 0 the SYS attribute, you can read and execute that file from any user number on the same drive. This feature gives you a convenient way to make your commonly used programs available under any user number, without having to maintain a copy of each command program in every user number.

To obtain a directory listing of those files marked with the SYS attribute, use the DIRS command. However, only those system files in the currently active user number will be displayed.

RW/RO ATTRIBUTE

The RW/RO file accessing attribute can be set to either RW (Read-Write) or RO (Read-Only). A file with the RW attribute can be read or written to at any time unless the disk is write-protected, or the drive containing the disk is set to Read-Only. If a file is marked RO, any attempt to write data to that file produces a Read-Only error message. Therefore you can use the RO attribute to protect important files. You can use the STAT Transient Utility program to assign the Read-Write or Read-Only attribute to a file or group of files.

You can also use STAT to assign the Read-Only attribute to a drive or reassign the drive to Read-Write. The default state of a drive is RW. However, when you change the diskette in a floppy diskette drive, CP/M-86 protects the new diskette by assigning the drive the RO attribute. At other times, you may want to use STAT to protect the contents of a diskette or disk in a particular drive during a processing session. In either case, to restore the drive to the RW state, press CTRL-C.

HOW ARE FILES STORED ON A DISK?

CP/M-86 records the filename, filetype, user number, and attributes of each file in the directory area of the diskette or disk. In the directory, CP/M-86 also records which diskette or disk sectors belong to which file. For floppy diskettes, the directory is large enough to store this data for up to 64 files on single-sided diskettes and 256 files on dual-sided diskettes. CP/M-86 automatically recognizes the type of diskette inserted in the APC drives, even when the diskette is changed.

CP/M-86 allocates directory and storage space for a file as records are added to the file. When you erase a file, CP/M-86 reclaims storage in two ways: it makes the file's directory space available to catalog a different file, and frees the file's storage space for later use. It is this *dynamic allocation* feature that makes CP/M-86 powerful. You don't have to tell CP/M-86 how big your file will become because CP/M-86 automatically allocates more storage for a file as it is needed, and releases the storage for reallocation when the file is erased.

CHANGING DISKETTES

CP/M-86 cannot, of course, do anything to a file unless the disk that holds the file is inserted into a drive and the drive is in ready status. This is not a problem with hard disks, which are permanently mounted. Floppy diskettes, however, must be inserted into a drive to be used. When a diskette or disk is mounted in a drive, it is *online* and CP/M-86 can access its directory.

When using floppy diskettes, you will sometimes need to take one diskette out of a drive and insert another that contains different files. You can replace an online diskette whenever you see the system prompt at your console. However, if you are going to write on the diskette, you must also tell CP/M-86 that you have changed a diskette by pressingCTRL-C directly after the system prompt. In response, CP/M-86 resets the drive for the new diskette. CTRL-C also resets the attributes of all logged-in drives to Read-Write.

If you forget to press CTRL-C after you change diskettes, CP/M-86 automatically protects the new disk. If you try to write to the new diskette using a text editor or copy program, CP/M-86 notices that the original diskette is no longer in the drive and writes the message:

Bdos err on *d*: RO

where *d*: is the drive specifier of the new disk. If you get this message, you must press CTRL-C once to return to the system prompt and press CTRL-C a second time to log in the new diskette.

CHANGING THE DEFAULT DRIVE

At any given time during operation of CP/M-86, there is one drive called the *default* drive. Drive A is usually the default drive when you start CP/M-86. Unless you put a drive specifier in your command line, CP/M-86 and the utilities look in the directory of the disk in the default drive for all program and data files. The CP/M-86 system prompt identifies the current default drive. For example, the message:

```
    A>
```

tells you that the A drive is the default drive. When you give commands to CP/M-86, you should remember which is the default drive. Then you will know which files an application program can access if you do not add a drive specifier.

If you have more than one drive, you might want to change the default drive. Do this at the system prompt by typing the drive specifier of the desired default drive followed by a colon. Then press RETURN.

```
    A>b:
```

This command, for example, changes the default drive to B. Unless you change the default drive again, all system prompt messages appear as:

```
    B>
```

The system prompt now indicates that CP/M-86 and its utilities will check in the directory of the diskette in Drive B for any file that does not have a drive specifier included in the file specification.

HOW CP/M-86 SEARCHES FOR COMMANDS

If a command keyword does not identify a Built-in command, CP/M-86 looks on the default or specified drive for a program file. It looks for a filename equal to the keyword and a filetype of CMD. For example, suppose you type the command line:

```
    A>ED MYPROG.BAS
```

CP/M-86 goes through the following steps to execute the command.

1. CP/M-86 first finds that the keyword ED does not identify one of the Built-in commands.

2. CP/M-86 searches for the utility program file ED.CMD in the directory of the default drive. If it does not find the file under the current user number, it looks under user number 0 for ED.CMD with the SYS attribute.
3. When CP/M-86 locates ED.CMD, it copies the program to memory and passes control to ED.
4. ED remains operational until you enter a command to exit ED.
5. CP/M-86 types the system prompt and waits for you to type another command line.

If CP/M-86 cannot find either a Built-in or a Transient Utility, it reports a keyword error by repeating the command line you typed on your screen, followed by a question mark. This tells you that one of the following errors has occurred.

- The keyword is not a Built-in command.
- No corresponding .CMD file appears under the current user number or with the SYS attribute under user 0.
- No corresponding .CMD file appears under the current user number or with the SYS attribute under user 0 on the specified drive when you have included a drive specifier.

For example, suppose your default disk contains only standard CP/M-86 utilities and you type the following command line:

```
A>EDIT MYPROG.BAS
```

CP/M-86 goes through the following steps to report the error.

1. CP/M-86 first examines the keyword EDIT and finds that it is not one of the Built-in commands.
2. CP/M-86 then searches the directory of the current user number on the default drive for EDIT.CMD. Next, it searches user number 0 for EDIT.CMD with the SYS attribute.
3. When the file cannot be found, CP/M-86 writes the message:

   ```
   EDIT?
   ```

   at the screen to tell you that the command cannot be executed.
4. CP/M-86 displays the system prompt and waits for you to type another command line.

OTHER CP/M-86 DEVICES

CP/M-86 manages all the peripheral devices attached to your computer. These can include storage devices such as disk drives, input devices such as keyboards and modems, and output devices such as printers, modems, and screens.

To keep track of input and output devices, CP/M-86 uses *logical device names.* The table below shows CP/M-86 logical device names and indicates whether the device is input or output.

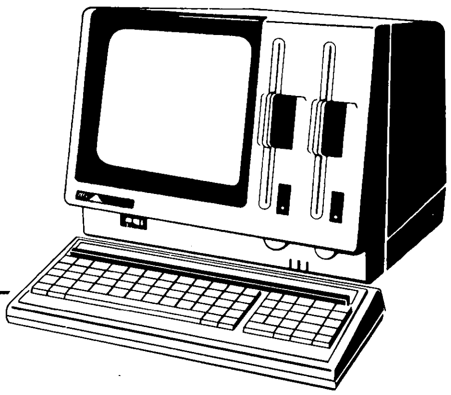
Table 2-2 CP/M-86 Logical Devices

DEVICE NAME	DEVICE TYPE
CON:	Console input and output
AXI:	Auxiliary input
AXO:	Auxiliary output
LST:	List output

CP/M-86 associates physical devices with the logical device names. For example, the default console input device is the keyboard and the default console output device is the screen. If you want CP/M-86 to manage an optional peripheral, you must use the STAT command to assign an alternate peripheral to the logical device name. For example, a STAT command can change the console input device from the keyboard to a teletype. STAT can assign a printer to the LST: logical output device name. Note that a logical output device can be assigned only one physical device at a time.

On CP/M-86 serial interface printers, SETCOM allows you to specify the parameters (baud rate, parity, and number of data bits) for the standard APC serial RS 232C port. These parameters must match those of the serial interface printer attached to the serial RS 232C port.

After using the STAT or CONFIG utility to redirect the printer output to the serial RS 232C port, execute the SETCOM utility to set the matching parameters for the serial port before attempting printer-bound output.



Chapter 3

Command Summary

This chapter provides a handy reference for all standard CP/M-86 commands. Descriptions of both Built-in and Transient Utility commands are provided in alphabetical order. Each command description includes entry syntax and options, as well as a short explanation of command operation with examples. More complicated commands are described in detail elsewhere. For example, ED (the CP/M-86 text editor) is described in Chapter 4, while ASM-86 and DDT-86 are described in the *CP/M-86 Programmer's Guide.*

The following section describes how command line file specifications are presented in the rest of the chapter. These specifications include required and optional command entry notations.

FILE SPECIFICATIONS

The file specifications included in this chapter consist of three distinct parts. To avoid confusion, each part is given a formal name that is used throughout the chapter. The following names describe the parts of a file specification:

- *drive specifier* is the optional disk drive, A through H, that contains the file or group of files to which you are referring. If a drive specifier is included in your command line, it must be followed by a colon.
- *filename* is the one-to-eight-character first name of a file or group of files.
- *filetype* is the optional one-to-three-character family name of a file or group of files. If the filetype is present, it must be separated from the filename by a period.

The following represents the general form of a file specification:

d:filename.typ

In the above form, *"d:"* represents the optional drive specifier, *"filename"* represents the one-to-eight-character filename, and *".typ"* represents the optional one-to-three-character filetype.

Each file specification, also referred to as a *filespec* in this chapter, names a particular file or group of files in the directory of the online disk given by the drive specifier. For example,

```
B:MYFILE.A86
```

is a file specification that indicates drive "B:", filename "MYFILE", and filetype "A86".

Valid combinations of the elements of a CP/M-86 file specification are shown in the following list.

- *filename*
- *d:filename*
- *filename.typ*
- *d:filename.typ*

If you do not include a drive specifier, CP/M-86 automatically supplies the default drive. If you omit the period and the filetype, CP/M-86 automatically includes a filetype of three blanks.

Some CP/M-86 commands accept wildcards in the filename and filetype parts of the command tail. For example,

```
B:MY*.A??
```

is a file specification with drive-specifier "B:", filename "MY*", and filetype "A??". This file specification might match several files in the directory.

HOW COMMANDS ARE DESCRIBED

The alphabetical list of Built-in and Transient Utility commands in this chapter is given in a specific form.

- The description begins with the command keyword in uppercase. Where appropriate, a phrase describing the command's purpose follows in parentheses.
- The *Syntax* section gives you one or more general forms to follow when you compose the command line.

- The *Type* section identifies the keyword as a Built-in or Transient Utility command. Built-in commands are always available for your use, while Transient Utility commands must be present on an online disk as a CMD program file.
- The *Purpose* section defines the general use of the command keyword.
- The *Remarks* section points out exceptions and special cases.
- The *Examples* section lists a number of valid command lines that use the command keyword. To clarify examples of interactions between the user and the operating system, the characters entered by the user are shown in color. CP/M-86's responses are shown in normal type.

The notation in the syntax lines describes the general command form using the following rules:

- Words in capital letters must be typed and spelled as shown, but you can use any combination of uppercase or lowercase letters.
- A lowercase word has a general meaning that is defined further in the text for that command. When you see the word *option*, for example, you can choose from a list of options provided.
- Italics identify variable names. They should be replaced with your real data in command lines.
- You can substitute a number for *n*.
- The symbolic notation *"d:"*, *"filename"*, *".typ"* and *"filespec"* have the general meanings described in the previous section.
- You must include one or more space characters where a space is shown, unless otherwise specified. For example, the PIP options do not need to be separated by spaces.
- Items enclosed within braces { } are optional. You can enter a command without the optional items. The optional items add effects to your command line as described.
- An ellipsis . . . tells you that the previous item can be repeated any number of times.
- When you can enter one or more alternative items in the command line, a vertical bar | separates the alternatives. Think of this vertical bar as the *or* bar.
- CTRL represents the Control Key on the APC keyboard.
- All other punctuation shown must be included in the command line.

SYNTAX NOTATION EXAMPLES

This section provides some examples of syntax notation using the STAT and PIP commands.

The CP/M-86 Transient Utility command STAT (status) displays the amount of free space in kilobytes for all online drives. It also displays the amount of space in kilobytes used by individual files. STAT can also assign the Read-Only (RO) or Read-Write (RW), and the System (SYS) or Directory (DIR) attributes to a file.

The Syntax section of the STAT command shows how the command line syntax notation is used:

Syntax:

```
STAT { filespec {RO | RW | DIR | SYS } }
       |         |                    | |
       |         |______optional______| |
       |                                |
       |____________optional____________|
```

The notation above indicates that the command tail following the command keyword STAT is optional. STAT alone is a valid command, but you can include a file specification in the command line. Therefore, the following is a valid command:

STAT *filespec*

Furthermore, the file specification can be followed by another optional value selected from the following:

RO
RW
DIR
SYS

Therefore, the following is also a valid command:

STAT *filespec* RO

The STAT command also accepts wildcards in the file specification. The following are, therefore, all valid command lines:

```
STAT
STAT X.A86
STAT X.A86 RO
STAT X.A86 SYS
STAT *.A86
STAT *.* RW
STAT X.* DIR
```

The CP/M-86 command PIP (Peripheral Interchange Program) is the file copy program. PIP can copy information from your screen to the disk or printer. PIP can combine two or more files into one longer file. PIP can also rename files after copying them. PIP offers another example of command line notation.

Syntax:

PIP *dest-filespec=source-filespec{,filespec...}*

For this example, *dest-filespec* is further defined as a destination file specification or peripheral device (printer, for example) that receives data. Similarly, *source-filespec* is a file specification or peripheral device (keyboard, for example) that transmits data. PIP also accepts wildcards in the filename and filetype. (See the PIP command summary for details regarding other capabilities of PIP.)

Some of the valid command lines that come from this syntax are shown below.

```
PIP  NEWFILE.DAT = OLDFILE.DAT
PIP  B: = A:THISFILE.DAT
PIP  B:X.BAS = Y.BAS, Z.BAS
PIP  X.BAS = A.BAS, B.BAS, C.BAS
PIP  B: = A:*.BAK PIP B: = A:*.*
```

THE ASM-86 (ASSEMBLER) COMMAND

Syntax:

ASM86 *filespec* { $parameter-list /}

Type:

Transient Utility

Purpose:

The ASM-86 Utility converts 8086 assembly language source statements into machine code form. The operation of the ASM-86 assembler is described in detail in the *CP/M-86 Programmer's Guide.*

Remarks:

The *filespec* names the character file that contains an 8086 assembly language program to translate. If you omit the filetype, a filetype of A86 is assumed. The assembler uses the drive specifier portion of the *filespec* as the destination drive for output files unless you include a parameter in the command tail to override this default.

Three output files are produced by the assembler and are assigned the filetypes listed below.

LST	contains the annotated source listing.
H86	contains the 8086 machine code in hexadecimal format.
SYM	contains all programmer-defined symbols with their program relative addresses.

The assembler assigns the same filename as the source filename to the LST, H86, and SYM files.

You can control the assembly process by including optional parameters in the *parameter-list*. Each parameter is a single parameter letter followed by a single letter device name. The parameters can be separated by blanks, but each parameter letter must be followed immediately by the device name.

The parameter letters are A, H, P, S, and F. The device names are the letters A through H, corresponding to the drive letters.

In addition, the letters X, Y, and Z have special meaning when used as device names:

X sends output to the screen.

Y sends output to the printer.

Z suppresses output.

Use the A parameter letter to override the default drive specifier to obtain the source file. The valid parameters are AA through AH.

Use the H parameter letter to override the default drive specifier to receive the H86 file. Valid parameters are HA through HH, and HX, HY, and HZ.

Use the P parameter letter to override the default drive specifier to receive the LST file. Valid parameters are PA through PH, PX, PY, and PZ.

Use the S parameter letter to override the default drive specifier to receive the SYM file. Valid parameters are SA through SH, SX, SY, and SZ.

Use the F parameter letter to select the format of the hexadecimal output file. Valid parameters are FI and FD. The FI parameter selects Intel format hex file output. The FD parameter selects Digital Research format hex file output. FD is assumed if neither FI nor FD appears as a parameter. Use FI when the program is going to be combined with a program generated by an Intel compiler or assembler.

When conflicting parameters appear on the command line, the rightmost parameter prevails.

Examples:

```
A>ASM86 MYPROG
```

The ASM86.CMD file must be on Drive A. The source file MYPROG.A86 is read from Drive A, and MYPROG.LST, MYPROG.H86, and MYPROG.SYM are written to Drive A.

```
B>ASM86 MYPROG.ASM $PX
```

The ASM86.CMD file must be on Drive B. The source file MYPROG.ASM is read from Drive B. The listing is written to the screen, and the MYPROG.H86 and MYPROG.SYM files are placed on Drive B.

```
A>ASM86 B:MYPROG $PY HC
```

The source file MYPROG.A86 is read from Drive B, the listing is sent to the printer, the file MYPROG.H86 is written to Drive C, and file MYPROG.SYM is placed on Drive B.

```
A>B:ASM86 MYPROG $SZ
```

The ASM86.CMD file must be on Drive B. The MYPROG.A86 file is read from Drive A. The MYPROG.LST and MYPROG.H86 files are written to Drive A. No MYPROG.SYM file is generated.

THE CHR (AUXILIARY CHARACTER GENERATOR) COMMAND

Syntax:

CHR

Type:

Transient Utility

Purpose:

The CHR utility permits you to create, change, or load into memory an auxiliary character set stored in a data file with the filetype CHR. Any number of auxiliary character sets can be created, each containing up to 256 character patterns identified by hexadecimal values 00 to FF. These auxiliary character set files are then available for use during application programs. Each auxiliary character within a set is constructed within an 8 by 16 matrix using the commands described below.

Remarks:

The CHR program has two functions: 1) loading an existing character set from disk to memory, and 2) updating or creating the contents of a character set. After you enter the command line as shown above, the following prompt appears:

```
UPDATE OR LOAD (U,L)?
```

If you enter L, you must then identify the input filename in one of the following formats:

- *filename*
- *d:filename*

where the assumed filetype is CHR. If you enter RETURN instead of a filename, the CHR program automatically searches the default disk for filename AUXCG.CHR, the default auxiliary character file. If the input file is located, the auxiliary character set is loaded into memory starting at hexadecimal address 0D80000.

If you enter U to update an auxiliary character file, the filespec you enter for the input file determines whether you will be maintaining an existing file or creating a new one. If the filename entered does not exist, the CHR Utility presumes you are creating a new file with the name entered on the drive specified (or the default drive if none was given).

An output file specification must then be entered. The output file stores the changed version of the character set. The entry format is the same as for the input filespec; if you enter RETURN instead of a filespec, the output file is the same as the input file.

After you identify the output file, the Auxiliary Character Generator command list entry screen appears.

The following table lists the commands available and the keys used for constructing, searching for, and loading auxiliary characters.

Table 3-1 Auxiliary Character Generator Utility Commands

COMMAND	KEY USED	FUNCTION
BACK	B	Displays the previous character in the auxiliary set according to the previous hexadecimal code. Pressing B at the first character in the file (code=00) displays the last character (code=FF).
BIT OFF	SPACE	Turns off the bit or graphic block at the current cursor position by overtyping a space. Then moves the cursor forward one position.
BIT ON	*	Turns on the bit at the current cursor position. The graphic block at the position appears highlighted. Moves the cursor forward one position.
CURSOR DOWN	↓	Moves the cursor down one row at a time within the matrix.
CURSOR HOME	HOME	Moves the cursor to the home position, the upper left corner of the matric (0,0).

Table 3-1 Auxiliary Character Generator Utility Commands (cont'd)

COMMAND	KEY USED	FUNCTION
CURSOR LEFT	←	Moves the cursor to the left one column at a time within the matrix.
CURSOR RIGHT	→	Moves the cursor to the right one column at a time within the matrix.
CURSOR UP	↑	Moves the cursor up one row at a time within the matrix.
DISPLAY	D	Displays or redisplays the current character following modifications.
END	E	Displays a prompt confirming the end of auxiliary character updating. Enter Y to end the program. When creating a new file, a prompt asking CREATE *filename?* appears. When updating an existing file, a prompt asking UPDATE *filename?* appears.
LOAD	L	Loads all characters in the auxiliary set being updated or created into memory.
NEXT	RETURN	Displays the next character in the auxiliary set according to the next hexadecimal code in sequence. Pressing RETURN at the last character in the file (code=FF) displays the first character (code=00).
SEARCH CODE	C	Prompts for input of a hexadecimal code, then displays the character corresponding to that code. Pressing RETURN instead of entering a code redisplays the last character shown on the screen.

THE CONFIG (HARDWARE CONFIGURATION) COMMAND

Syntax:

CONFIG

Type:

Transient Utility

Purpose:

The CONFIG Utility allows you to identify the hardware characteristics of the APC to the operating system. You can also make changes that affect the performance of applications and utility programs.

Remarks:

The CONFIG command has five options. They are displayed in the CONFIGURATION MENU that appears on the screen when the command is entered.

1. FLOPPY DISK READ AFTER WRITE CONFIGURATION
2. LIST DEVICE CONFIGURATION
3. AVAILABLE DISK DRIVE SELECTION
4. UPDATE MEMORY RESIDENT CP/M-86 1.1
5. FINISH AND UPDATE DISK RESIDENT CP/M-86 1.1

To select an option, press the number of the option.

Floppy Disk Read After Write Configuration

When the operating system is set to floppy diskette read after write (FDRAW), the Basic I/O System (BIOS) writes a sector and reads the sector that was just written to determine whether a write error has occurred. This process catches single-sector errors before diskette information is destroyed by multiple bad writes. However, it causes an increase in I/O time of approximately 200% for writes. Normally, the system is set for non-FDRAW. However, you may change this configuration by selecting option 1 on the CONFIGURATION MENU.

When you select option 1, the system responds with a message that identifies the current FDRAW setting and prompts you for a change, as in the following example.

```
Currently " non read after write " mode!
          Mode change ?  (Y/N) :
```

Press Y to change the mode to read after write. Press N to leave the current mode unchanged. After the response is entered, the current mode message and prompt are erased from the screen.

List Device Configuration

Option 2 allows you to reconfigure the list device. When this option is selected, the system displays prompts at the top of the screen and a grid in the bottom portion of the screen.

First, you are prompted to select the primary list output device interface. Press P (uppercase is required) to select the parallel centronix interface. Press uppercase S to select the serial RS 232-C interface.

Next, the system prompts you to select the number of data bits for the interface you selected. The options are 7 or 8 bits. Your entry must match the output device's requirements in order to produce legible output.

If you selected 7 bits, the system then prompts you for the parity convention required by the list device. Press E for even parity, O for odd parity.

The last part of this option allows you to change the code mapping table that the BIOS uses to send output to the list device. The current code mapping is displayed in the table at the bottom of the screen.

Use the code map when the output character code from the application generates a different symbol from the one on an ASCII-type list device for the same output code. When the code map is set up as a one-to-one map, the incoming code is the same as the outgoing code to the printer. You might want to change the one-to-one map to allow the APC to interface directly with an EBCDIC printer.

The column and row values of the character map show the least and most significant digits (4 bits per digit) of the incoming code. The two-digit hexadecimal numbers at each column and row intersection are the output codes generated for the respective incoming codes.

Use the four cursor movement keys to position the cursor in the table. The keys allow non-destructive access to each of the values in the code map. After all of the modifications (if any) are made, press ESC to return to the main menu.

Available Disk Drive Selection

Option 3 from the CONFIGURATION MENU displays the current drive configuration and allows you to change it. Select the configuration of the drives that exactly matches the hardware configuration of your APC system. Maximum configuration for the APC is four floppy diskette and two hard disk drives.

Diskette drives are accessible as logical drives A:, B:, C:, and D:. When hard disk units are configured, the four logical disk drives are E: and F: for unit 0, G: and H: for unit 1.

To return to the main menu, press ESC.

If you select a non-existent drive, an endless "disk-not-ready" error message sequence is displayed. To recover from that error, reboot the system.

Update Memory Resident CP/M-86 1.1

When you select option 4 from the CONFIGURATION MENU, the system updates the copy of CP/M-86 that is currently loaded in memory with the changes you have made. The message "Memory resident CP/M-86 1.1 has been updated" is displayed at the bottom of the screen. The changes are in effect until CP/M-86 is rebooted from diskette or a new configuration is loaded.

Finish and Update Disk Resident CP/M-86 1.1

Select this option to exit from CONFIG. The system prompts you to determine whether to update the CP/M-86 file on the diskette with changes made during the session or to exit without saving the changes.

When you press 5, the system displays the following prompt.

Update *d*:CPM.SYS FILE ? (Y/N) :

Press N to exit CONFIG without updating the disk resident image. Press Y to save the new configuration as the CPM.SYS file. If you select Y and the diskette already contains a file named CPM.SAV, the following prompt is displayed.

d:CPM.SYS already exists!
ERASE *d*:CPM.SAV file ? (Y/N) :

Press Y to erase the old CPM.SAV file and replace it with the current CPM.SAV file. Press N to exit without changes to the files on the diskette.

THE COPYDISK (COPY DISK) COMMAND

Syntax:

```
COPYDISK
```

Type:

Transient Utility

Purpose:

The COPYDISK Utility copies all the information on one diskette to another diskette, including the CP/M-86 system tracks if they are present on the source diskette. COPYDISK also offers the option of verifying the contents of one diskette against the contents of a second diskette. The verification routine can be performed in conjunction with the copy routine or separately.

Before copying to a brand new diskette, you should first prepare it using FORMAT, the diskette formatting program also located on the CP/M-86 distribution diskette. If you copy to a used diskette, COPYDISK writes all the information, including blank space, from the source diskette over the information on the destination diskette. COPYDISK does not copy to or from a hard disk.

Remarks:

To display instructions on how to use COPYDISK, enter the keyword HELP with the command tail COPYDISK or see Chapter 1 of this manual.

To successfully copy from one diskette to another, you must make sure that your destination diskette is not write-protected. Check that there is a foil tab covering any existing write-protect notch on the edge of your diskette before inserting the diskette into the destination drive.

COPYDISK is an exact cylinder-for-cylinder, track-for-track, sector-for-sector copy utility, and is the fastest way to copy an entire diskette. However, if many files have been created and erased on the source diskette, the records belonging to a particular file might be randomly placed on the diskette. In this case, it might be more efficient (although slower) to use PIP (or PIP1) to copy the files individually and thus put all the records in sequential order on the new diskette.

Examples:

```
A>COPYDISK
```

When you invoke COPYDISK, it prompts you first for the type of routine you want to perform: copy, verify, or both. The program then prompts you for the source and destination diskettes. With a dual-drive APC, you typically copy and/or verify from Drive A to Drive B. However, if you have a single-drive APC, you must copy/verify from Drive A to Drive A. COPYDISK gives you instructions on the screen when it is time to change diskettes. The COPYDISK utility does not accept keyboard input when data transfers to or from the diskette are occurring.

The example below shows the values you enter and COPYDISK's responses for performing a copy/verify routine with a dual-drive APC. COPYDISK copies from the master diskette on Drive A to the new diskette on Drive B. COPYDISK then verifies the data copied.

```
  Full Disk Copy/Verify Utility V1.1
Copy or Verify or Copy & Verify (C, V, B)? B

Source Disk Drive (A-D) ? A

Destination Disk Drive (A-D) ? B

Copying Disk A: To Disk B:
Is This What You Want To Do (Y/N) ? Y

Copy Started
  Reading Cylinder   [00]  (These cylinder numbers are in-
  Writing Cylinder   [00]  cremented as the operation occurs.)

Copy Completed

Verify Started
  Reading Cylinder    [00]
  Verifying Cylinder  [00]

Verify Completed

Copy/Verify Another Disk (Y/N) ? N

A>
```

THE DDT-86 (DYNAMIC DEBUGGING TOOL) COMMAND

Syntax:

DDT86 {*filespec*}

Type:

Transient Utility

Purpose:

The DDT-86 Utility allows you to monitor and test programs developed for the 8086 processor.

The DDT-86 single letter commands, their parameters and options are described in Table 3-2. The actual command letter is printed in uppercase. The italicized letters following the command represent parameter arguments. Those shown in brackets are optional. Replace the arguments with the appropriate parameter values as described in the list following the table.

Table 3-2. DDT-86 Commands

COMMAND		MEANING
A*s*	(Assemble)	Enter Assembly Language Statements
B*s,f,sl*	(Block Cmp)	Compare Blocks of Memory
D{*W*} {*s* {*,f*}}	(Display)	Display Memory in Hex and ASCII
E*filespec*	(Execution)	Load Program for Execution
F*s,f,bc*	(Fill)	Fill Memory Block — Byte
FW*s,f,wc*	(Fill)	Fill Memory Block — Word
G {*s*} {*,bl* {*b2*}}	(Go)	Begin Execution
H*wc1,wc2*	(Hex)	Hexadecimal Sum and Difference

Table 3-2. DDT-86 Commands (cont'd)

COMMAND		MEANING
I*command tail*	(Input)	Set Up Input Command Line
L {*s* {,*f*}}	(List)	List Memory in Mnemonic Form
M*s*,*f*,*d*	(Move)	Move Memory Block
R*filespec*	(Read)	Read Disk File to Memory
S{*W*}*s*	(Set)	Set Memory Values
T{*n*}	(Trace)	Trace Program Execution
TS{*n*}	(Trace)	Trace and Show All Registers
U{*n*}	(Untrace)	Monitor Execution without Trace
US{*n*}	(Untrace)	Monitor and Show All Registers
V	(Verify)	Show Memory Layout after Disk Read
Wfilespec{,*s*,*f*}	(Write)	Write Content of Block to Disk
X{*r*}	(Examine)	Examine and Modify CPU Registers

PARAMETER:	REPLACE WITH VALUE FOR:
bc	byte constant
b1	breakpoint one
b2	breakpoint two
d	destination for data
f	final address
n	number of instructions to execute
r	register or flag name
s	starting address
s1	second starting address
W	word 16-bit
wc	word constant

The overall operation of DDT-86, along with each single letter command, is described in detail in the *CP/M-86 Programmer's Guide.*

Remarks:

If the file specification is not included, DDT-86 is loaded into user memory without a program to be tested. You must not use the DDT-86 commands G, T, or U until you have loaded a program to be tested. The program is usually loaded using the E command.

If the file specification is included, both DDT-86 and the test program file specified by *filespec* are loaded into user memory. Use G, T, or U to begin execution of the test program under supervision of DDT-86.

If the filetype is omitted from the file specification, a filetype of CMD is assumed.

DDT-86 cannot directly load programs in hexadecimal (H86) format. You must first convert them to command file form (CMD) using the GENCMD Utility described later in this chapter.

Examples:

```
A>DDT86
```

The DDT-86 Utility is loaded from Drive A to user memory. DDT-86 displays the "-" prompt when it is ready to accept commands.

```
A>B:DDT86 TEST.CMD
```

The DDT-86 Utility is loaded from Drive B to user memory. The program file TEST.CMD is then loaded to user memory from Drive A. DDT-86 displays the address where the file was loaded and the "-" prompt.

THE DIR (DIRECTORY) BUILT-IN COMMAND

Syntax:

```
DIR  {d:}
DIR  {filespec}
DIRS {d:}
DIRS {filespec}
```

Type:

Built-in

Purpose:

The DIR and DIRS Built-in commands display the names of files catalogued in the directory of an online disk or diskette. The DIR Built-in command lists the names of files in the current user number that have the Directory (DIR) attribute. DIR accepts wildcards in the file specification.

The DIRS command displays the names of files in the current user number that have the System (SYS) attribute. Therefore, even though you can access System (SYS) files that are stored in user 0 from any other user number on the same drive, DIRS only displays those user 0 files if the current user number is 0. DIRS accepts wildcards in the file specification.

Remarks:

If the drive and file specifications are omitted, the DIR command displays the names of all files with the DIR attribute on the disk in the default drive and current user number. DIRS displays the SYS files similarly.

If the drive specifier is included, but the filename and filetype are omitted, the DIR command displays the names of all DIR files in the current user number on the disk in the specified drive. DIRS displays the SYS files similarly.

If the file specification contains wildcard characters, all filenames that satisfy the match are displayed on the screen.

If no filenames match the file specification, or if no files are cataloged in the directory of the disk in the named drive, the DIR command displays the message:

```
NO FILE
```

If system (SYS) files reside on the specified drive, DIR displays the message:

```
SYSTEM FILE(S) EXIST
```

If non-system (DIR) files reside on the specified drive, DIRS displays the message:

```
NON-SYSTEM FILES(S) EXIST
```

You cannot use a wildcard character in the drive specifier.

Examples:

```
A>DIR
```

All files with the DIR attribute cataloged in the current user number in the directory of the disk mounted in Drive A are displayed on the screen.

```
A>DIR B:
```

All DIR files in the current user number on the disk in Drive B are displayed on the screen.

```
A>DIR B:X.A86
```

If the file X.A86 is present on the disk in Drive B, the DIR command displays the name X.A86 on the screen.

```
A>DIR *.BAS
```

All DIR files with filetype BAS in the current user number on the disk in Drive A are displayed on the screen.

```
B>DIR A:X*.C?D
```

All DIR files in the current user number on the disk in Drive A with a filename that begins with the letter X, and a filetype containing C as the first character and D as the last character are displayed on the screen.

```
A>DIRS
```

The preceding command displays all files in the current user number on Drive A that have the system (SYS) attribute.

```
A>DIRS *.CMD
```

This command displays all files in the current user number on Drive A with a filetype of CMD that have the system (SYS) attribute.

THE DISP (AUXILIARY CHARACTER DISPLAY) COMMAND

Syntax:

DISP

Type:

Transient Utility

Purpose:

The DISP Utility displays the auxiliary character set currently loaded in memory. The character set is loaded using the CHR Utility. When you invoke DISP, the characters for all 256 hexadecimal codes in the currently loaded set are displayed in a matrix on the screen. Codes without characters assigned appear as blanks or miscellaneous images. If no character set is currently loaded, all positions appear as miscellaneous images. Figure 3-1 shows the grid format used to display the auxiliary character set (using DISP) and the ROM character set (using DISP1).

Each character in the auxiliary character RAM or in ROM is described by a two-digit hexadecimal code in the table. The column number is the most significant digit. The intersecting row number is the least significant digit. For example, a character displayed at column 7 row 6 has a corresponding hexadecimal value of 76.

Characters are displayed in red on APC color systems. They are highlighted on monochrome APC systems.

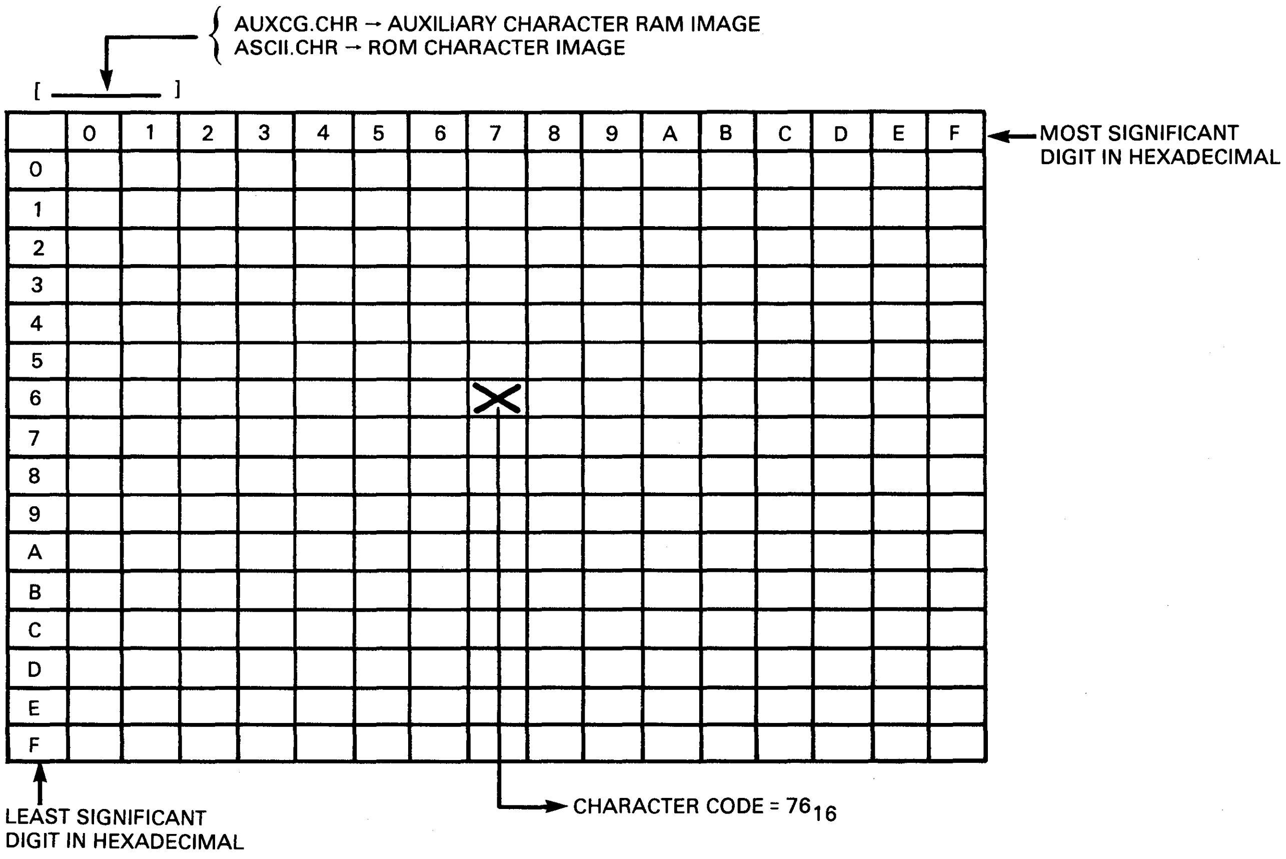

AUXCG.CHR → AUXILIARY CHARACTER RAM IMAGE
ASCII.CHR → ROM CHARACTER IMAGE
[ ]
0 1 2 3 4 5 6 7 8 9 A B C D E F
MOST SIGNIFICANT DIGIT IN HEXADECIMAL
LEAST SIGNIFICANT DIGIT IN HEXADECIMAL
CHARACTER CODE = 76₁₆

Figure 3-1. Grid Format for Display of Auxiliary and ROM Character Images

THE DISP1 (ROM CHARACTER DISPLAY) COMMAND

Syntax:

DISP1

Type:

Transient Utility

Purpose:

The DISP1 Utility displays the ROM character set.

Remarks:

DISP1 displays the ROM character images in a grid format, just as DISP displays the auxiliary RAM character set.

THE ED (CHARACTER FILE EDITOR) COMMAND

Syntax:

ED *input-filespec* {*d:* | *output-filespec*}

Type:

Transient Utility

Purpose:

The ED Utility lets you create and edit a file on either hard disk or diskette. It is a *line-oriented* and *context* editor. This means that you create and change character files line by line, or by referencing individual characters within a line.

The ED Utility lets you create or alter the file named in the file specification.

This utility uses a portion of memory as the active text *buffer* where you add, delete, or alter the characters in the file. You use the A command to read all or a portion of the file into the buffer. You use the W or E command to write all or a portion of the characters from the buffer back to the file.

An imaginary *character pointer* (CP) can be moved to any location in the buffer: the beginning of the buffer, between two characters in the buffer, or at the end of the buffer.

You interact with the ED Utility in either *command* or *insert* mode. In command mode, ED displays the "*" prompt on the screen. When the "*" appears, you can enter the single letter commands from the table below to read text from the buffer, move the CP, or change the ED mode of operation.

Table 3-3 ED Command Summary

COMMAND	ACTION
*n*A	Append *n* lines from original file to memory buffer.
0A	Append file until buffer is one half full.
#A	Append file until buffer is full (or end of file).
B, -B	Move CP to the beginning (B) or bottom (-B) of buffer.
*n*C, -*n*C	Move CP *n* characters forward (C) or back (-C) through buffer.
*n*D, -*n*D	Delete *n* characters before (-D) or from (D) the CP.
E	Save new file and return to CP/M-86.
F*string*{CTRL-Z}	Find character string.
H	Save the new file, then reedit, using the new file as the original file.
I	Enter insert mode; use CTRL-Z to exit insert mode.
I*string*{CTRL-Z}	Insert string at CP.
J*search-str*CTRL-Z*ins-str*CTRL-Z*del-to-str*{CTRL-Z}	Juxtapose strings.
*n*K, -*n*K	Delete (kill) *n* lines from the CP.
*n*L, -*n*L, OL	Move CP *n* lines.
*n*Mcommands	Execute commands *n* times.
n, -*n*	Move CP *n* lines and display that line.

Table 3-3 ED Command Summary (cont'd)

COMMAND	ACTION
n:	Move to line *n*.
:*n*command	Execute command through line *n*.
N*string*{CTRL-Z}	Extended find string.
O	Return to original file.
*n*P, -*n*P	Move CP 23 lines forward and display 23 lines at console.
Q	Abandon new file, return to CP/M-86.
R	Read X$$$$$$$.LIB file into buffer.
R*filespec*{CTRL-Z}	Read filespec into buffer.
S*deletestring*CTRL-Z*insertstring* {CTRL-Z}	Substitute string.
*n*T, -*n*T, OT	Type *n* lines.
U, -U	Uppercase translation.
V, -V, OV	Line numbering on/off, display free buffer space.
*n*W	Write *n* lines to new file.
*n*X	Write or append *n* lines to X$$$$$$$. LIB.
*n*X*filespec*{CTRL-Z}	Write *n* lines to filespec or append if previous x command applied to the same file.
OX	Delete file X$$$$$$$.LIB.
OX*filespec*{CTRL-Z}	Delete filespec.
*n*Z	Wait *n* seconds.

Chapter 4 provides a detailed description of the overall operation of the ED Utility and the use of each command.

Remarks:

Include the output file specification only if the file named by the input file specification is already present and you do not want the original file replaced. The file named by the output file specification receives the altered text from the input file, which remains unchanged.

If the output file specification contains only the drive specifier, the output filename and filetype become the same as the input filename and filetype.

If the input filename entered is not found on disk, the ED Utility creates the file and writes the message:

```
NEW FILE
```

If no output *filespec* is given, the input file is preserved with a filetype of BAK before it is replaced. If you issue an ED command line that contains a *filespec* with filetype BAK, ED creates and saves your new edited version of the BAK file, but deletes your source file, leaving no backup. If you want to save the original BAK file, use the REN command first to change the filetype from BAK, so that ED can rename its input file to BAK.

If you include the optional output filespec and give it the same name as the input filespec, ED again creates and saves your new edited version of the output filespec, but deletes the original input filespec because it has the same name as the output file. You cannot, of course, have two files with the same name in the same user number on the same drive.

If the filename given by the input filespec is already present in the directory, you must issue the A command to read portions of the file to the buffer. If the size of the file does not exceed the size of the buffer, the command:

```
#a
```

reads the entire file to the buffer.

The i (Insert) command places the ED Utility in insert mode. In this mode, any characters you type are stored in sequence in the buffer starting at the current CP.

Any single letter commands typed in insert mode are not interpreted as commands, but are simply stored in the buffer. To return from insert mode to command mode, press CTRL-Z.

Single letter commands are normally typed in lowercase. Commands that must be followed by a character sequence end with CTRL-Z if they are to be followed by another command letter.

Any single letter command typed in uppercase tells ED to internally translate to uppercase all characters up to the CTRL-Z that ends the command.

When enabled, line numbers that appear on the left of the screen take the form:

nnnnn:

where *nnnnn* is a number in the range 1 through 65535. Line numbers are displayed for your reference and are not contained in either the buffer or the character file. The screen line starts with a colon (:) when the CP is at the beginning or end of the buffer.

Examples:

```
A>ED MYPROG.A86
```

If the file is not already present, this command line creates the file MYPROG.A86 on Drive A. The command prompt

```
:*
```

appears on the screen. This tells you that the CP is at the beginning of the buffer. If the file is already present, issue the command

```
#a
```

to fill the buffer. Then type the command

```
Op
```

to fill the screen with the first 23 lines of the buffer. Type the command

```
e
```

to stop the ED Utility when you are finished changing the character file. The ED Utility leaves the original file unchanged as MYPROG.BAK and the altered file as MYPROG.A86.

```
A>ED MYPROG.A86 B:NEWPROG.A86
```

The input file is MYPROG.A86 on the default Drive A. The original file remains unchanged when the ED Utility finishes, with the altered file output as NEWPROG.A86 on Drive B.

```
A>B:ED MYPROG.A86 B:
```

The ED.CMD file must be on Drive B. The original input file is MYPROG.A86 located on Drive A. It remains unchanged, with the altered program output on Drive B as MYPROG.A86.

THE ERA (ERASE) BUILT-IN

Syntax:

```
ERA filespec
```

Type:

Built-in

Purpose:

The ERA Built-in command removes one or more files from the directory of a disk. Wildcard characters are accepted in the command tail. Directory and data space are automatically reclaimed for later use by another file.

Remarks:

Use the ERA command with care since all files that satisfy the file specification are removed from the disk directory.

Command lines that take the form:

```
ERA. (d:)*.*
```

require your acknowledgment since they reclaim all file space. Enter y in response to the All (Y/N)? prompt if you want to remove all files; enter n if you want to avoid erasing any files. The message

```
NO FILE
```

appears on the screen if no files match the file specification.

Examples:

```
A>ERA X.A86
```

This command removes the file X.A86 from the diskette in Drive A.

```
A>ERA *.PRN
```

All files with the filetype PEN are removed from the diskette in Drive A.

```
B>ERA A:MY*.*
```

Each file on Drive A with a filename that begins with MY is removed from the diskette.

```
A>ERA B: *.*
```

All files on Drive B are removed from the disk. However, to complete the operation, you must enter Y when the ERA command prompts you for confirmation with the message:

```
ALL (Y,N)?
```

THE FORMAT (DISKETTE FORMATTING) COMMAND

Syntax:

```
FORMAT
```

Type:

Transient Utility

Purpose:

The FORMAT Utility initializes diskettes to a recording format acceptable to CP/M-86. This initialization includes preparing all cylinders (00-76) as well as the directory and file allocation table areas on the diskette. In addition, FORMAT analyzes the entire diskette surface for any defective cylinders, tracks, or sectors. In the event FORMAT reports that a particular area on the diskette cannot be formatted, you can instruct FORMAT to reattempt formatting the cylinder. If the cylinder cannot be formatted, remove and discard the diskette.

This program formats only floppy diskettes. To format a hard disk, use the HDFORMAT Utility described later in this chapter.

Remarks:

To display instructions on how to use FORMAT, enter the keyword HELP with the command tail FORMAT.

Because FORMAT erases all information currently on the diskette, care should be taken when specifying the drive that contains the diskette being formatted. It is advisable to check the directory of any used diskettes being reformatted to ensure no necessary data will be destroyed. If you have an APC with two diskette drives, it is also advisable to use Drive B as the initialize device. Because your CP/M-86 diskette is generally in Drive A, using Drive B as the initialize device provides some protection against inadvertently destroying your system diskette. If you have an APC with one diskette drive, be sure to remove the system diskette from the drive immediately after the FORMAT program loads.

Examples:

```
A>FORMAT
```

When you invoke FORMAT, it prompts you first for the name of the drive that contains the diskette to be formatted:

```
*** VOLUME INITIALIZER V1.1 [CP/M] ***
    INITIALIZE DEVICE [A,B,C,D] :
```

FORMAT automatically identifies the media type of the disk in the drive you select as one of the following:

- single-sided, single-density (FD-1D)
- dual-sided, double-density (FD-2D)

A READY? [R:READY A:ABORT] prompt gives you the opportunity to review your actions before proceeding. If you made a mistake or want to change your entries, enter A to abort the process. The program returns to the INITIALIZE DEVICE [A, B, C, D] prompt. Otherwise, if you are ready to continue, press R.

As formatting occurs, the number of the cylinder being formatted appears in brackets as follows:

```
REFORMATING CYLINDER NO. [nn]
```

If a damaged cylinder is detected during formatting, the following prompt appears with the number of the cylinder in brackets:

```
* BAD CYLINDER NO. [nn]
[R:RETRY A:ABORT] :
```

Press R to try to format the cylinder again. If the cylinder cannot be formatted, press A to abort the FORMAT program. The program returns to the INITIALIZE DEVICE [A, B, C, D] prompt. Remove the diskette and label it as damaged. Insert another diskette in the appropriate drive and try again.

When all cylinders on a disk are successfully formatted, the following prompt appears:

```
* NORMAL END
[A:AGAIN E:END]
```

To repeat the process, press A. Otherwise, to return to the A> prompt, press E.

THE GENCMD (GENERATE CMD FILE) COMMAND

Syntax:

GENCMD *filespec* {8080 CODE [A*n*, B*n*, M*n*, X*n*] DATA [A*n*, B*n*, M*n*, X*n*]
STACK [A*n*, B*n*, M*n*, X*n*] EXTRA [A*n*, B*n*, M*n*, X*n*] X1[...]}

Type:

Transient Utility

Purpose:

The GENCMD Utility uses the hexadecimal output of ASM-86 and other language processors to produce a CMD file. An optional parameter list follows the file specification.

You need to know how to use GENCMD when you write assembly language programs that become Transient Utility commands. The operation of GENCMD is described in detail in the *CP/M-86 System ReferenceGuide.*

The parameter-list consists of up to nine keywords with a corresponding list of values. The keywords are:

8080
CODE
DATA
STACK
EXTRA
X1
X2
X3
X4

The keyword 8080 identifies the CMD file as an *8080 Memory Model* where code and data groups overlap. The remaining keywords define segment groups that have specific memory requirements. The values that define the memory requirements are separated by commas and enclosed in square brackets ([]) following each keyword. The bracketed keywords and related values must be separated from other keywords by at least one blank.

The values included in brackets are defined below, where *n* represents a hexadecimal constant of from one to four digits. The value *n* represents a *paragraph* value where each paragraph is 16 bytes long. The paragraph value corresponds to the byte value *n* *16, or hhhh0, in hexadecimal.

A*n*	Load Group at Absolute Location *n*
B*n*	Begin Group at Address *n* in the Hexadecimal File
M*n*	The Group Requires a Minimum of *n* *16 Bytes
X*n*	The Group Can Address up to *n* *16 Bytes

Remarks:

Use the 8080 keyword for programs converted from 8-bit microprocessors to CP/M-86 or programs that require a single, continuous segment for all defined segments. The programs load into an area with overlapping code and data segments. The code segment in the program must begin at location 100H when the 8080 model is used.

Use A*n* for any group that must be loaded at an absolute location in memory. Do not use an A value in the command tail unless you know that the requested absolute area will be available when the program runs.

Use B*n* when your input Hex file does not contain information that identifies the segment groups. This value is not necessary when you H86 file is the output from the Digital Research ASM-86 assembler, unless the ASM-86 parameter FI was included.

Use the M*n* value when you include a data segment that has an uninitialized data at the end of the segment.

Use X*n* when your program can use a larger data area, if available, than the minimum given by M*n*.

Examples:

```
A>GENCMD MYFILE
```

The file MYPROG.H86 is read from Drive A. The output file MYPROG.CMD is written back to Drive A. The input H86 file includes information that marks the program as operating with a particular memory model.

```
B>GENCMD MYFILE     CODE[A40]     DATA[M30,XFFF]
```

The file MYFILE.H86 is read from Drive B. The MYFILE.CMD output file is written to Drive B. The code group must be loaded at location 400 hexadecimal. The data group requires a minimum of 300 hexadecimal bytes, but if available, the program can use up to FFF0 bytes.

GRAPHICS (GSX-86) COMMAND

Syntax:

GRAPHICS {*d:*|NO}

Type:

Transient Utility

Purpose:

The GRAPHICS (GSX-86) Utility enables you to use graphics products: GSS-GRAPH, GSS-DRAW, GSS-KERNEL, and GSS-PLOT. GSS-GRAPH allows you to graph and plot data by making simple menu selections. GSS-DRAW provides an advanced capability: using simple symbols to create more complex graphics. GSS-KERNEL is a utility library. Using GSS-PLOT, you can graph with just a few calls from a high-level language.

Remarks:

The GRAPHICS command uses information from the ASSIGN.SYS file and the device driver files to interface with graphics programs. ASSIGN.SYS is the Assignment Table file. It assigns specific device drivers (printers, plotters, and display consoles) to logical device numbers. On the distribution diskette, ASSIGN.-SYS is set up with the device driver file DDNECAPC.SYS in order to have an appropriate interface with the APC.

The Assignment Table consists of the filenames of all device drivers and a logical device number for each device driver, as shown in the following example.

```
1    DDNECAPC.SYS    ;NEC APC DRIVER
```

The number '1' is the logical device number. It may be one or two digits. It is recommended that you assign terminals device numbers from 01-10, plotters 11-20, and printers 21-30.

The filename of the device driver appears next in the Assignment Table. It must be one of the device driver files on the CP/M-86 distribution diskette. Device driver filenames begin with the letters "DD" and have the extension ".SYS". For a complete list of device driver filenames and their associated devices, see Table 1-5.

The colon (:) separates the device driver entry from any following comments.

If you need to change or add logical device numbers or device driver filenames, edit the ASSIGN.SYS file as you would any other file using the ED command. The largest device driver must be listed first in the Assignment Table. Use the STAT command to display the approximate size of files. The first device driver is called the default driver, the one the system always uses unless another driver is specified. If the system has enough room for the largest driver, it will always have enough room for any other driver that GSX-86 uses.

Enter the command GRAPHICS with or without a drive specifier. The program searches the ASSIGN.SYS file on the specified or default drive for the assignment table that lists the drivers available. (Both the ASSIGN.SYS file and the device drivers must be on the specified disk.)

When the graphics program is correctly installed and loaded, the following message is displayed.

GSX-86 installed; GIOS is *dddd* bytes long at *xxxx.ooo*

This means that GSX-86 and the correct device driver were installed. Here *dddd* is the exact size of the device driver. Note that size for making changes to the Assignment Table.

Disabling Graphics Mode

To disable GSX-86, enter GRAPHICS NO. This frees the memory space formerly used by GSX and the device driver. When the graphics program is disabled, the following message is displayed.

GSX-86 not installed.

See Appendix D for a complete list of GSX-86 error messages.

THE HDBACKUP (HARD DISK BACKUP) COMMAND

Syntax:

HDBACKUP

Type:

Transient Utility

Purpose:

HDBACKUP saves (backs up) the contents of the APC's 10 megabyte hard disks (APC-H26 and APC-H27) on diskette and restores the disks from diskette. HDBACKUP allows saving or restoring of any one of the logical drives associated with each disk unit — E or F for the APC-H26 and G or H for the APC-H27.

Remarks:

HDBACKUP requires dual-sided, double-density diskettes, with a format of 26 sectors per track. Any of the APC's diskette drives may be used when saving or restoring data with HDBACKUP. Because each hard disk drive's capacity is many times larger than the diskette capacity, five diskettes are required to perform a backup of one hard disk drive.

During a SAVE operation, HDBACKUP places a serial number on each diskette. This ensures that a subsequent load operation using these diskettes will not be finished until all data has been restored.

Note that data saved on diskettes can be accessed or used only by HDBACKUP.

To initiate HDBACKUP, enter the program name at the CP/M-86 command entry prompt. You may use uppercase or lowercase. The following display appears, prompting for entry of an operation specifier:

```
****Full Hard Disk Back Up V.1 1 (CP/M)
******Save All/Load All [S,L] ? :_
```

Press S to save (backup) data from hard disk to diskette.

To load (restore) data from diskette to hard disk, press L.

```
**Copy or Verify or Copy & Verify [C,V,B] ? :_
```

Press C to copy data to or from hard disk. Press V to verify contents of a hard disk. Press B to copy data to or from hard disk and verify that copied data is correct.

HDBACKUP prompts for the hard disk drive being used.

**Which hard disk drive: [*available hard disk drives*] ? :

The brackets contain the names of all logical hard disk drives available — either [E,F] or [E,F,G,H], depending on system configuration. Press the letter that corresponds to the hard disk drive you intend to back up or restore. The program prompts for the diskette drive being used.

**Please insert a floppy disk
and indicate the floppy disk : [*available diskette drives*] ?

Available diskette drive names, either [A,B] or [A,B,C,D], appear in brackets.

Insert a diskette in an available drive, then enter the name of the diskette drive. A confirmation display appears, providing information about the operation selected.

For a SAVE operation, the display appears with the appropriate values in brackets and *d*1 and *d*2 representing disk drives:

Hard disk track size: [*hhh*] (Decimal)
Hard disk end cylinder: [*ccc*] (Decimal)
05 floppies are needed to save data from hard disk *d*1:
Current floppy in use: *ff*
Total number of floppy disk needed: 05
Backup date: DAY *mm/dd/yy* Time

Copying hard disk *d*1 to floppy disk *d*2
Is this what you want to do? (Y/N) :

For a LOAD operation, the display appears with the appropriate values appearing in brackets and *d*1 and *d*2 representing drive names.

```
Hard disk track size: hhh (Decimal)
Hard disk end cylinder: ccc (Decimal)
Floppy disk track size: ddd (Decimal)
Current floppy disk serial no.: nn (Decimal)
Current floppy in use: ff
Total number of floppy disk needed: 05
Backup date: DAY mm/dd/yy Time

Copying floppy disk d1 to hard disk d2:
Is this what you want to do? Y/N :
```

To cancel a SAVE or LOAD operation and return to the prompt, enter N. To continue, enter Y.

While HDBACKUP is copying data, the screen displays the following messages, with *ccc* representing the cylinder number presently being copied.

```
** Copy started
Current hard disk cylinder [ccc] (Decimal)
Current floppy disk cylinder [ccc] (Decimal)
```

During verification, the screen displays the cylinders being checked as follows.

```
** Verify started
Current hard disk cylinder [ccc] (Decimal)
Current floppy disk cylinder [ccc] (Decimal)
```

Because five diskettes are required to backup a single hard disk drive, HDBACKUP pauses at several points during a SAVE or LOAD operation to allow the diskette to be changed. During SAVE, this occurs when the diskette being copied is filled. When performing a LOAD operation, the diskette must be changed when all data has been read from it.

During a SAVE or LOAD operation, the following prompt appears to indicate that the diskette should be changed. HDBACKUP displays the names of all available diskette drives in brackets.

```
Operations on floppy disk d: have been completed.

** Please insert a floppy disk
   and indicate the floppy disk: [available diskette drives]?
```

In response to the prompt, remove the previous diskette, insert another diskette in one of the available drives, and enter the name of that drive. Note that the drive specified does not have to be the same one used for the previous diskette.

When a new diskette is inserted and a drive name is entered, HDBACKUP displays information about the diskette in use and the operation being performed. A confirmation prompt requires Y to continue the operation as specified, N to change the entries. When the operation is completed, the following prompt appears:

```
** Back-up has completed. **
Again? [Y/N] ? :
```

Press Y to return to the operation specifier prompt. Otherwise, to end HDBACKUP, press N.

See Appendix D for HDBACKUP error messages.

THE HDFORMAT (HARD DISK FORMAT) COMMAND

Syntax:

HDFORMAT

Type:

Transient Utility

Purpose:

HDFORMAT is a utility that allows formatting the APC's 10 megabyte hard disks (APC-H26 or APC-H27). The utility allows formatting of one or both of the logical drive divisions of each hard disk. The APC-H26 is segmented into logical drives E and F; the APC-H27 is segmented into logical drives G and H.

Each hard disk is formatted to contain the following physical address structure:

- 26 sectors per track
- 8 tracks per cylinder
- 181 cylinders per disk (179 available for data)

The hard disk unit must be formatted using HDFORMAT before any I/O operations can take place on the disk.

Remarks:

When errors are detected during formatting, the utility initiates seven retry attempts to format the track. Any bad track addresses found during formatting are logically reallocated to the next available track. HDFORMAT creates an error map at cylinder 0, track 2, sector 0 to control this reallocation and correct logical address access. Note that prior to formatting, the disk may contain a permissible number of tracks already identified by the factory prior to shipment as "permanent bad tracks." Permanent bad tracks are identified in the address identification section of the track and are logically inaccessible.

Initiate HDFORMAT by entering the program name at the CP/M-86 command entry prompt. After the program is initiated, the following display appears, prompting for entry of a drive specifier.

```
*** HARD DISK FORMATTER V1.1 (CP/M) ***
FORMAT DRIVE :
```

To end the program at this point, press CTRL-C. To continue, enter any one of the following values in response to the drive specifier prompt.

Table 3-4 Disk Drive Specifiers For HDFORMAT

ENTRY	PROGRAM ACTION
0	Logical drives E and F of the APC-H26 disk will be formatted.
1	Logical drives G and H of the APC-H27 disk will be formatted.
E	Logical drive E of the APC-H26 disk will be formatted.
F	Logical drive F of the APC-H26 disk will be formatted.
G	Logical drive G of the APC-H27 disk will be formatted.
H	Logical drive H of the APC-H27 disk will be formatted.

The first time you use HDFORMAT, you must enter 0 and format logical drives E and F.

If you press 0 or 1, the following message appears.

*HARD DISK UNIT *u* CONSISTS OF DRIVE *d*1:,*d*2:

where *u* is the unit (0 or 1) and *d*1 and *d*2 represent drives E and F or G and H.

After the drive specifier is entered, the following prompt appears requiring confirmation or rejection of the drive selection.

```
READY [Y:YES N:NO] :
```

Press N to redisplay the prompt for drive specifier. Press Y to initiate the format operation. During formatting, the screen displays the following message:

```
*FORMATTING CYLINDER NO. [nnn]
```

where *nnn* represents the cylinder number (0-180) being formatted. The appropriate drive or unit number appears in parentheses.

If a bad track or a permanent bad track is encountered during formatting, the cylinder and track numbers (represented below by *ccc* and *tt*) are reported on the screen as follows. Note that an asterisk before the address information denotes a permanent bad track identified by the factory.

```
*** BAD TRACK INFORMATION (drive or unit number) ***
      P[CCCTT]     P="*"   : PERMANENT BAD TRACK
                           CCC=CYLINDER NO.
                           TT=TRACK NO.
[ccctt] *[ccctt]  [ccctt]
```

If no bad tracks are encountered, the following messages appear.

```
*** BAD TRACK INFORMATION (drive or unit number) ***
NO BAD TRACK
```

When formatting is completed, the following prompt appears.

```
*NORMAL END
 [A:AGAIN E:END] :
```

To continue formatting hard disk drives or units, press A. The program then redisplays the drive specifier prompt. If you press B, the following message appears on the screen.

```
PLEASE REBOOT SYSTEM
ENTER FNC-CTR-BREAK
```

Hold down the FNC, CTRL, and BREAK STOP keys to reboot the system.

See Appendix D for error messages that may appear during operation of HDFORMAT.

THE HELP (HELP) COMMAND

Syntax:

```
HELP {topic} {subtopic1 subtopic2 ... subtopic8} {[P]}
HELP [E]
HELP [C]
```

Type:

Transient Utility

Purpose:

The HELP command provides summarized information for all the CP/M-86 commands described in this manual. HELP with no command tail displays a list of all the available topics. HELP with a topic in the command tail displays information about that topic, followed by any available subtopics. HELP with a topic and a subtopic displays information about the specific subtopic.

HELP with the E and C command tails is used to modify the HELP.HLP text file to provide customized HELP information.

Remarks:

After HELP displays the information for your specified topic, it displays the special prompt HELP> on the screen. You can continue to specify topics for additional information, or simply press RETURN to return to the CP/M-86 system prompt.

You can abbreviate the names of topics and subtopics. Usually one or two letters is enough to specifically identify the topics.

HELP with the [P] option prevents the screen display from stopping every 23 lines.

Examples:

```
A>HELP
```

The command above displays a list of topics for which help is available.

```
A>HELP STAT OPTIONS
```

The command above includes the subtopic for STAT named OPTIONS. In response, HELP displays information about options associated with the STAT command.

```
A>HELP ED
```

The command above displays general information about the ED Utility.

When referencing a subtopic, you must type the topic name and the subtopic, otherwise the HELP program cannot determine which main topic you are referencing. You can also enter a topic and subtopic following the program's internal prompt, HELP>, as shown below.

```
A>HELP ED COMMANDS
```

This form of HELP displays information about commands internal to ED.

Customizing HELP.HLP

The HELP.CMD file is the command file that processes the text stored in the text file HELP.HLP and displays it on the screen. Although HELP.HLP is a text file, it cannot be directly edited. To edit the file, you must use HELP.CMD to convert HELP.HLP into a file named HELP.DAT before you can edit its contents.

Use the following forms of the HELP command to change HELP.HLP to HELP.DAT, then back to HELP.HLP again following modifications.

HELP [E]	The HELP[E] command accesses the file HELP.HLP on the default drive, removes the header record, and creates a file called HELP.DAT on the default drive. You can now invoke the ED Utility or a word processing program to edit or add your own text to the HELP.DAT file.
HELP [C]	The HELP [C] command accesses your edited HELP.DAT file on the default drive, generates a new index for the entries record, and builds a revised HELP.HLP file on the default drive. HELP.CMD can now display your new HELP.HLP file.

To add topics and subtopics to the HELP.DAT file, you must use a specified format. Use the following format to enter a topic heading in the HELP.DAT file:

```
///nTOPICNAME
```

Three backslashes are topic delimiters and must begin in column one of each entry. In the format statement above, *n* is a number from 1 to 9 that signifies the level of the topic. A main topic always has a level number of 1. The first subtopic has a level number of 2. The next subtopic has a level number of 3, and so forth up to a maximum of nine levels. TOPICNAME, the name of the topic, can be a maximum of twelve characters. The entire line must be terminated by pressing RETURN.

Use the following guidelines to properly edit and insert text into the HELP.DAT file.

- Topics should be ordered in ascending alphabetical order.
- Subtopics should be ordered in ascending alphabetical order within their respective topics.
- Levels must be indicated by a number from 1 to 9.

Some examples of topic and subtopic lines in the HELP.HLP file are shown below.

```
///1NEW UTILITY
///2COMMANDS
///3EXAMPLES
```

The first example shown above illustrates the format of a main topic line. The second example shows how to number the first subtopic of that main topic. The third example shows how the next level subtopic should be numbered. Any topicname with a level number of 1 is a main topic. Any topicname with a level number from 2 to 9 is a subtopic within its main topic.

THE KEY (FUNCTION KEY ASSIGNMENT) COMMAND

Syntax:

```
KEY
```

Type:

Transient Utility

Purpose:

The KEY Utility allows you to create and dynamically assign a series of data strings to the APC function keys PF1 through PF16. The function key definitions are stored in either the default CPM.SYS file or in files with the KEY filetype. KEY allows you to create, display, change, and delete the function key definitions. The program also allows you to change the currently loaded function key assignments.

A total of 32 functions can be defined in each KEY file (or in CPM.SYS) considering the shifted function of each key. The shifted function of PF1 through PF16 is accessed by simultaneously pressing the function key and FNC.

Remarks:

The KEY program has two functions: 1) loading a set of existing function key assignments from disk, and 2) updating or creating a function key file. After you enter the command line as shown above, the following prompt appears:

```
SOFT KEY DEFINITION PROGRAM V1.1
  UPDATE OR LOAD (U,L) ?
```

If you enter L, you must then identify the input filename in one of the following formats:

- filename
- d:filename

where the assumed filetype is KEY. If you press RETURN instead of entering a filespec, the KEY program automatically loads the current function key assignments stored in CPM.SYS.

If you enter U to update a function key file, the filespec you enter for the input file determines whether you will be maintaining an existing file or creating a new one. If the filename entered does not exist, the KEY Utility presumes you are creating a new function key assignment file. If you press RETURN instead of entering a filename, the default input file is CPM.SYS.

An output file specification must then be entered. The output file stores the new or changed function key set. The entry format is the same as for the input filespec; if you press RETURN instead of entering a filespec, the output file is the same as the input file.

After you identify the output file and confirm the start of the updating procedure, KEY displays a list of commands available. The following table describes these commands.

Table 3-5 KEY Utility Commands

COMMAND	ACTION
{F}*##*,[*XXXXXXXXXXXXXXX*]	Assigns a function to a particular key. The *##* identifies the number of the key (1-16) you are assigning. The optional F before the number indicates a function assignment to the key in its shifted mode. The X's represent the actual function assigned, expressed in no more than 15 characters.
D	Displays all function key assignments currently in memory.
A	Deletes all function key assignments currently in memory.
E	Ends function key assignment. You then have the option of also updating the default key assignments in CPM.SYS with the current assignments. If you do, the original assignments are saved in a file named CPM.SAV.

In addition to data strings, you can also include standard ASCII and CP/M-86 control codes within function key assignments. To invoke a carriage return as a part of the data string, use the & character.

Examples:

The following represent valid examples of function key assignments that you can specify after you enter input and output file specifications.

```
01,PIP B:*.*↑
```

The character ↑ performs a carriage return after the command PIP B:*.* is entered by pressing PF1.

```
F16,KEY↑
```

The character ↑ performs a carriage return after the command KEY is entered by pressing FNC and PF16 simultaneously.

THE LDCOPY (COPY LOADER ROUTINE) COMMAND

Syntax:

LDCOPY

Type:

Transient Utility

Purpose:

The LDCOPY Utility copies the CP/M-86 system loader routine from your distribution (or working) diskette to another diskette. (It does not copy the loader routine either to or from hard disk.) The system loader routine must be present in conjunction with CPM.SYS on system track 0 to allow CP/M-86 to be loaded from a particular diskette. Note, however, that the loader routine does not appear in a directory listing of the destination diskette.

Remarks:

After invoking LDCOPY at the command line as shown above, you must identify the source diskette drive from which the routine will be copied. The source drive you specify must contain the file LOADER.CMD which is automatically called by the LDCOPY Utility.

Place the source diskette containing LOADER.CMD in the drive specified, then press RETURN. LDCOPY then copies the system loader routine into memory. You must then identify the destination diskette drive to which the loader routine will be copied. After you insert a diskette in the appropriate drive and confirm the procedure, LDCOPY copies the system loader from memory onto system track 0 of the destination disk. Because the system loader is still stored in memory, you can continue to specify destination diskettes and copy the routine to a number of diskettes.

Examples:

Invoke the LDCOPY Utility as shown above. The series of prompts and responses shown below represent the procedure for copying the system loader from a diskette in Drive A to a diskette in Drive B.

```
Source Drive Name A
Source on A. Then Type Return RETURN

Function Complete
Destination Drive Name (Or Return To Reboot) B

Function Complete
Destination Drive Name (Or Return to Reboot) RETURN

A>
```

THE PIP (PERIPHERAL INTERCHANGE PROGRAM) COMMAND

Syntax:

PIP *dest-file* {[G*n*]} *dev=source-file* {[*options*]} *dev*{[*options*]}

Type:

Transient Utility

Purpose:

The PIP Utility is one of the most versatile and useful programs provided on the CP/M-86 distribution diskette. PIP is used to transfer files between the peripheral devices (floppy diskette and hard disk drives, keyboard, console, printer, and so forth) of your APC. PIP performs the following operations:

- copies one or more files from one disk or diskette and/or user number to another;
- renames files copied, if desired;
- combines two or more files into one;
- copies a character file from disk or diskette to the printer or other auxiliary logical device;
- creates a file on hard disk or floppy diskette from input from the console or other logical input device;
- directly transfers data from a logical input device to a logical output device.

If you are using a hard disk unit, it is a good idea to PIP the CP/M-86 system files from the distribution diskette (or working copy) to the hard disk. You will still need the working copy of CP/M-86 to boot the system, but once the system is up, you can run all CP/M-86 utilities and built-in commands from the hard disk.

The following sections describe the various PIP operations in more detail, including syntax information and examples.

Single File Copy

Syntax:

PIP *d*: {[G*n*]} = *source-filespec* {[*options*]}

PIP *dest-filespec* {[G*n*]} = *d*: {[*options*]}

PIP *dest-filespec* {[G*n*]} = *source-filespec* {[*options*]}

In the examples of command line syntax above, the following symbols are used:

d	drive specifier
G*n*	user number specification
source-filespec	source file specification
dest-filespec	destination file specification
options	list of options that affect or limit operation

Purpose:

The first form shown above is the simplest way to copy a file. PIP looks for the source-filespec on the default or optionally specified drive. PIP copies the file to the drive specified by *d*: and gives it the same name as the *source-filespec*. You can use the [G*n*] option to place the destination file in any user number *n*. The [G*n*] option is the only one that can be specified with the destination filespec; however, several options can be included with the source filespec to control or affect the PIP operation. These source file options are listed in Table 3-6.

The second form of the PIP command shown above is a variation of the first. PIP looks for the file named by the destination filespec on the drive specified by *d*:, copies it to the default or optionally specified drive, and gives it the same name as the destination filespec.

The third form shows how to rename the file after you copy it. You can copy the file to the same drive and user number, or to a different drive and/or user number. PIP looks for the file specified by *source-filespec*, copies it to the location specified in *dest-filespec*, and gives it the name indicated by *dest-filespec*.

Remember: unless otherwise specified by the [G*n*] option, PIP copies files from and to the current user number.

Remarks:

During PIP, the data is first copied to a temporary file on the destination disk to ensure that the entire file can be constructed within the space available on the disk or diskette. PIP creates this temporary file even when the file named by *dest-filespec* already exists on the destination disk. PIP gives this temporary file the filename speciied by the destination filespec with a filetype of $$$. If there is not enough room on the destination disk for the temporary file, PIP does not perform the copy and displays an error message. To ensure that there is enough room on the destination disk, before invoking PIP, use the STAT Utility (described later in this chapter) to see how much free space exists on the disk.

PIP will not backup a file from hard disk to diskette when the file size is greater than the maximum available space on the destination diskette. To copy such a large file, use HDBACKUP.

When the copy operation is successfully completed, PIP changes the temporary filetype of $$$ to the actual filetype specified for the destination. If the copy operation succeeds and a file with the same name already exists on the destination disk, the old file is erased before PIP renames the temporary file.

Note that all currently assigned file attributes (SYS, DIR, RW, RO) are transferred with the source file. If the destination file already exists with a Read-Only (RO) attribute, PIP prompts you for confirmation before erasing the existing file. You can also use the W option with the source-filespec to write over files marked RO.

Examples:

```
A>PIP B:=oldfile.dat [v]
A>PIP B:oldfile.dat=A: [v]
```

Both forms shown above cause PIP to read the file oldfile.dat from Drive A and create an exact copy on Drive B. This is called the short form of PIP. Either the source or destination only names a drive and does not include a filename. This form can be used to copy a file to a different drive or user number. However, it cannot be used to copy a file from one drive or user number to the same drive and user number. The [v] option verifies that the source file and destination file are identical. It is a good idea to always use this option when copying files.

```
A>PIP B:newfile.dat=A:oldfile.dat [v]
```

This command copies the file oldfile.dat from Drive A to Drive B and renames it newfile.dat. The file remains unchanged on Drive A. This is known as the long form of PIP because both a source and a destination filename are given in the command line.

```
A>PIP newfile.date = oldfile.dat [v]
```

Using the long form of PIP, you can copy a file from one drive and user number to the same drive and user number. This effectively gives you two copies of the same file on one drive or within one user number; however, each file has a distinct name in the directory.

```
A>PIP B:PROGRAM.BAK = A:PROGRAM.DAT [G1 v]
```

This command copies the file PROGRAM.DAT from user number 1 on Drive A to the currently selected user number on Drive B and changes the filetype on Drive B to BAK.

```
B>PIP program2.dat = A:program1.dat [E V G3]
```

This command uses several options defined in Table 3-5. PIP copies the file named programl.dat from user number 3 on Drive A to the currently selected user number on Drive B, and renames the file program2.dat. The E option included with the source filespec echoes the transfer of data on the system console (generally the display screen). See Table 3-6 for a complete list and description of all options.

Multiple File Copy

Syntax:

PIP *d*: {[G*n*]} = {*d*:} *wildcard-filespec* {[*options*]}

Purpose:

When you use the wildcard characters (described in Chapter 2) in the source filespec, PIP locates and copies qualifying files one at a time to the destination drive. Note that no *dest-filespec* can be included when you use wildcards: the destination filename is the same as the source filename for any qualifying file. Also note that the destination drive and/or user number must be different from those specified for the source. Otherwise, PIP aborts the copy operation and displays an error message.

PIP displays the word COPYING followed by each qualifying filename as the copy operation proceeds.

Examples:

```
A>PIP B:=A:*.CMD
```

This command causes PIP to copy all the files on Drive A with the filetype CMD to Drive B.

```
A>PIP B:=A:*.*
```

This command causes PIP to copy all the files on Drive A to Drive B. This command is distinct from the COPYDISK Utility in several ways. COPYDISK copies all information from one disk to another, including the system tracks (0-1). Because COPYDISK performs a physical copy of all cylinders on the source disk, all blank space and other nonessential data are copied. The arrangement and structure of the files copied using COPYDISK are the same as that on the source disk. Because of CP/M-86's dynamic allocation of file space and extents, this can mean that a file is located over several different physical locations on the disk.

The PIP command above, however, is a logical copy. Only the qualifying files are copied from the source disk. In addition, as each file is copied, it is written in a contiguous area of the disk. This type of copy ensures more effective use of the copied files on the new disk.

```
A>PIP B:=a:PROG???.*
```

This command causes PIP to copy all files beginning with the characters PROG and having any filetype from Drive A to Drive B.

```
A>PIP B: [G1]=A:*.A86
```

This command causes PIP to copy all files with a filetype of A86 in the default user number of Drive A to user number 1 on Drive B.

Combining Files

Syntax:

PIP *dest-file* {[G*n*]} =*source-file*1 {[*opt*]}, *source-file*2 {[*opt*]}, ...}

Purpose:

This form of the PIP command allows you to *concatenate* (combine) two or more source files into a single destination file. PIP copies the files specified in the source from left to right and combines them into a single file with the name specified by *dest-file*. The [G*n*] option can be used with the destination filespec to place the concatenated file in a particular user number. You can also specify any number of options for each source file.

Remarks:

In general, PIP copies files character by character. PIP performs this transfer until it locates a CTRL-Z character, which it presumes designates the end of the file. You can stop file concatenation at any time during a character by character transfer by pressing any key on the keyboard.

All of the PIP options, described in Table 3-6, perform a character by character transfer except the following:

G*n*, K, O, R, V, and W

When concatenating files using any of these options, PIP only searches the last record of each file being copied for an end-of-file (CTRL-Z) character. For example, use the [O] option when concatenating machine code files. This option causes PIP to ignore any embedded CTRL-Z characters in the file. If you do not use one of the options shown above when concatenating files and PIP locates an embedded CTRL-Z, the transfer stops.

Examples:

```
A>PIP NEWFILE=FILE1,FILE2,FILE3
```

The three source files named FILE1, FILE2, and FILE3 are joined from left to right and copied to NEWFILE.$$$. NEWFILE.$$$ is renamed to NEWFILE upon successful completion of the copy operation. All source and destination files are on the diskette in Drive A.

```
A>PIP B:X.A86 = Y.A86, B:Z.A86
```

The file Y.A86 on Drive A is joined with Z.A86 from Drive B and copied to the temporary file X.$$$ on Drive B. The file X.$$$ is renamed to X.A86 on Drive B upon successful completion of the copy operation.

Copying Files To and From Auxiliary Devices

Syntax:

```
PIP dest-filespec {[Gn]} = source-filespec {[options]}
    AXO:                   AXI:  {[options]}
    CON:                   CON:  {[options]}
    PRN:                   NUL:
    LST:                   EOF:
```

Purpose:

This special form of PIP command line allows you to copy files between various input and output devices. The files transferred in this way must contain printable characters.

Each peripheral device attached to the APC is assigned to a logical device name. These assignments can be altered using the STAT Utility described later in this chapter. However, in general, the following represent the logical and physical device assignments.

LOGICAL DEVICE	DESCRIPTION
CON:	The physical device designated as the console. When used as a source, CON: is usually the keyboard. When used as a destination, it is usually the display screen.
AXI:	Auxiliary input or output device.
AXO:	Auxiliary output device.

LST:	The output device designated as the destination for a listing. LST: is usually the printer.

Three logical device names can be used to perform special functions:

NUL:	A logical source device name that sends 40 hexadecimal zeroes (nulls) to the destination.
EOF:	A logical source device name that sends a CTRL-Z (end-of-file) to the destination.
PRN:	A logical source device name usually assigned to a printer. Unlike LST: , PRN: expands the tabs to every 8 columns, includes line numbers with all outputs, and produces a page eject every 60 lines.

Examples:

```
A>PIP B:FUNFILE.SUE = CON:
```

In general, this command sends whatever you type at the keyboard to the file named FUNFILE.SUE on Drive B. To end keyboard output to the file, type a CTRL-Z.

```
A>PIP LST:=CON:
```

In general, this command sends whatever you type at the keyboard directly to the list device, usually the printer. To end keyboard output to the printer, press CTRL-Z.

```
B>PIP PRN:=CON:,MYDATA.DAT
```

This command causes all input from the keyboard to be directly output to the printer until CTRL-Z is pressed. At that time, PIP reads the file MYDATA.DAT on the default Drive B and copies its contents to the printer. Because PRN: is the destination device, tabs are expanded, line numbers are added, and page ejects occur every 60 lines.

```
A>PIP LST:=B:DRAFT.TXT[T8]
```

The file DRAFT.TXT on Drive B is written to the printer device. The [T8] option expands any tab characters in the file to the nearest column that is a multiple of 8.

```
A>PIP PRN:=B:DRAFT.TXT
```

The file DRAFT.TXT on Drive B is written to the printer device. Tabs are automatically expanded, line numbers are added, and pages automatically eject after 60 lines.

Multiple Command Mode

Syntax:

```
PIP
```

Purpose:

This form of the PIP command loads the utility into memory and allows you to continue entering command lines after each transfer is performed.

Remarks:

After PIP is loaded into memory, an asterisk (*) appears on the screen as the command line prompt. At this prompt, you can type any valid command line previously described for PIP. To return to CP/M-86 system prompt, press either RETURN or CTRL-C at the asterisk prompt.

Examples:

```
A>PIP
*NEWFILE=FILE1,FILE2,FILE3
*B:PROGRAM.BAK=A:PROGRAM.DAT[G1]
*A:=B:*.CMD
*B:=*.*
*
```

This command loads the PIP program into memory, displays the input prompt (*), then awaits entry. The effects of the PIP command line examples shown above are the same as previously described.

Using Options With PIP

Purpose:

Options enable you to process your source file or input in special ways. You can expand the position of tab characters, translate from uppercase to lowercase, extract portions of text, verify that the copy is correct, and so forth.

The PIP options are listed in Table 3-6. The character *n* represents a number; the character *s* represents a sequence of characters terminated by CTRL-Z.

When using these options in a PIP command line, the option must immediately follow the file or device it affects. *The option or option list must be enclosed in square brackets* - [ ]. For those options requiring a numeric value, no blanks can occur between the letter and the value.

The only option allowed for a destination file specification is [G*n*]. For a source file specification, any number of options is allowed. Multiple options for a single *source-filespec* can be separated by blanks, but the entire option list must be enclosed in square brackets.

Table 3-6 PIP Options

OPTION	FUNCTION
D*n*	Delete any characters beyond column *n*. This parameter follows a source file that contains lines too long to be handled by the destination device — an 80 character printer, for example. The number *n* should equal the maximum column width of the destination device.
E	Echo transfer at the console. When this parameter follows a source name, PIP displays the source data on the screen as the copy is taking place. The source must contain character data.
F	Filter form-feeds. When this parameter follows a source name, PIP removes all form-feeds embedded in the source data. To change form-feeds set for one page length in the source file to another page length in the destination file, use both the F parameter to delete the undesired form-feeds and the P parameter (described below) to simultaneously set the new form-feed length.

Table 3-6 PIP Options (cont'd)

OPTION	FUNCTION
G*n*	Get source from or send it to user number *n*. When this parameter follows a source name, PIP searches the directory of user number *n* for the source filename. When it follows the destination filename, PIP places the destination file in the user number specified by *n*. The number must be in the range 0-15.
H	Hex data transfer. PIP checks all data for proper Intel hexadecimal file format. The console displays error messages when errors occur.
I	Ignore 00 records in the transfer of Intel hexadecimal format files. The I option automatically sets the H option.
K	Kill console display during transfer.
L	Translate uppercase alphabetic characters in the source file to lowercase in the destination file. This parameter follows the source device or filename.
N	Add line numbers to the destination file. When this parameter follows the source filename, PIP adds a line number followed by a colon to each line copied. Numbers start with 1 and increase by one. If N2 is specified, PIP adds leading zeroes to the line number and inserts a tab after the number. If the T parameter is also set, PIP expands the tab.
O	Object file transfer for machine code (nonprintable, noncharacter) files. PIP ignores embedded CTRL-Z codes during concatenation and transfer. Use this option to combine object code files.
P*n*	Set page length. The parameter *n* specifies the number of lines per page. When this parameter modifies a source file, PIP includes a page eject at the beginning of the destination file and at every *n* lines. If *n* is not specified, or if *n*=1, PIP automatically inserts page ejects every 60 lines. When you also use the F option, PIP ignores form-feeds in the source data and inserts new form-feeds in the destination data at the page length specified by *n*.

Table 3-6 PIP Options (cont'd)

OPTION	FUNCTION
Q*s*	Quit copying from the source device after the string *s* is read. When used with the S option, this parameter can extract a portion of a source file. The string argument must be terminated by CTRL-Z.
R	Read system (SYS attribute) files. Normally, PIP ignores files marked with the system attribute in a disk directory. But when this parameter follows a source filename, PIP copies system files, including their attributes, to the destination.
S*s*	Start copying from the source device at the string *s*. The string argument must be terminated by CTRL-Z. When used with the Q option, this parameter can extract a portion of a source file.
T*n*	Expand tabs. When this parameter follows a source filename, PIP expands tab characters (CTRL-I) in the destination file. PIP replaces each CTRL-I from the source file with enough spaces to position the next character in a column divisible by *n*.
U	Translate lowercase alphabetic characters in the source to uppercase in the destination. This parameter follows the source device or filename.
V	Verify that data has been copied correctly. PIP compares the destination data to the source data to ensure that the two are exactly the same. The destination must be a disk file.
W	Write over files marked with the RO (Read-Only) attribute. Normally, if a PIP command tail includes an existing RO file as the destination, PIP does not overwrite the file unless you confirm the operation. When this parameter follows the source name, PIP automatically overwrites the RO file without further operator intervention. If the command tail contains multiple source files, this parameter need follow only the last file in the list.
Z	Zero the parity bit. When this parameter follows the source name, PIP sets the parity bit of each data byte in the destination file to zero. The source must contain character data.

Examples:

```
A>PIP NEWPROG.A86=CODE.A86[L],DATA.86[U]
```

This command creates a file named NEWPROG.A86 on Drive A by concatenating the files CODE.A86 and DATA.A86 also on Drive A. During the copy operation, CODE.A86 is translated to lowercase and DATA.A86 is translated to uppercase. The destination file NEWPROG.A86 contains this result.

```
A>PIP CON:=WIDEFILE.A86[D80]
```

This command writes the character file WIDEFILE.A86 from Drive A to the console, but deletes all characters beyond column 80 in the file.

```
A>PIP LST:=B:LONGPAGE.TXT[FP65]
```

This command writes the file LONGPAGE.TXT from Drive B to the printer. As the file is printed, any form-feed characters in the disk file are removed and replaced by a form-feed after every 65 lines.

```
B>PIP LST:=PROGRAM.A86[NT8U]
```

This command writes the file PROGRAM.A86 from Drive B to the printer. The N parameter tells PIP to automatically number each line. The T8 parameter expands tabs to every eighth column on the output listing. The U parameter translates all lowercase letters to uppercase as the file is printed.

```
A>PIP PORTION.TXT=LETTER.TXT[SDearSirCTRL-ZQSincerelyCTRL-Z]
```

This command extracts a portion of the LETTER.TXT file from Drive A by starting to copy from the first occurrence of the character string "Dear Sir." All subsequent characters from LETTER.TXT up to and including the first occurrence of the character string "Sincerely" are copied to the file PORTION.TXT.

```
B>PIP B:=A:*.CMD[VWR]
```

This command copies all files with the filetype CMD from Drive A to Drive B. The V parameter tells PIP to read the destination files to verify that the data was correctly transferred. The W parameter lets PIP automatically overwrite any destination files that already exist on Drive B with the RO attribute. The R parameter tells PIP to read any qualifying system files (those marked with the SYS attribute) from Drive A.

THE PIP1 (PIP FOR SINGLE-DRIVE OPERATION) COMMAND

Syntax:

PIP1

Type:

Transient Utility

Purpose:

The PIP1 utility allows single and multiple file transfers from one diskette to another in the same diskette drive, Drive A. It is used to backup files on a single-drive APC system. PIP1 does not replace PIP. PIP must be used to correct, verify, and concatenate files.

Remarks:

PIP1 does not operate with files located on a hard disk drive. Drive specifications are unnecessary, because you only use Drive A. Specifying a drive will cause an error message to be displayed. A diskette change is necessary during the copying phase.

Wildcard characters operate as they do for PIP. All file specifications are padded to the right with spaces if fewer than eight characters are entered for the file name, or fewer than three characters are specified for the filetype. To copy multiple files, use *.* or ????????.???. No renaming is possible.

When PIP1 copies a file, it marks the source diskette with the filename SSSSSSSS.SSS. It marks the destination diskette with the filename DDDDDDDD.DDD. The identification files are opened on the diskette to protect the user from inadvertently mixing up the source and destination diskettes. This identification method prohibits copying files from a diskette to itself, or from more than one source to a destination. When all operations are complete, PIP1 deletes the two identification files from the diskette directory.

Copying to multiple destinations is allowed as long as the destination does not change during the transfer of one file. Once the transfer is complete on a file, a new destination is acceptable.

If you abort a PIP1 operation, the identification files may remain on either the source or destination diskettes. Erase the files by using the ERA command.

The SSSSSSSS.SSS and DDDDDDDD.DDD filenames and .SSS filetype are reserved for exclusive use of PIP1.

Error messages inform you of possible errors, either in sequencing source and destination diskettes, or having a problem with space allocation on the destination medium. Examples and definitions of possible error messages are shown in Appendix D.

The only key entries that PIP1 recognizes are RETURN, R, A, and E. If you press RETURN before it is requested, only one RETURN is accepted from the system input buffer, and all others are discarded. After an error message, you can recover by pressing R (for RETRY) or inserting the correct diskette and pressing RETURN.

After the last file is transferred and the identification files are erased, PIP1 prompts with the following message.

```
A (AGAIN) or E (END) ?
```

Press A to execute the same sequence again. Do not enter a new command line at this time. You may use a new source diskette and/or destination diskette for subsequent passes through PIP1. This allows multiple sources (maybe different files) to be transferred to different destination diskettes.

Examples:

The following table lists examples of the effect of wildcard characters with the PIP and PIP1 commands.

Table 3-7 Wildcard Characters for PIP and PIP1

COMMAND		RESULT
.	= *.*	Full disk copy of all files.
.	= *.CMD	All .CMD files are copied.
CPM.SYS	= CPM.SYS	Only CPM.SYS is copied.
CPM.OLD	= CPM.SYS	CPM.SYS is copied into CPM.OLD.
.	= File1.*	All files with name File1 are copied.
.	= FIL*1.*	All files beginning with FIL are copied.
.	= FIL*.CP*	All files beginning with FIL and ending with .CP are copied.
.	= ABC???.???	All six letter files beginning with ABC are copied.

THE POW (POWER OFF) COMMAND

Syntax:

```
POW
```

Type:

Transient Utility

Purpose:

The POW Utility automatically turns off the main power supply of the APC. The command can be incorporated into a batch processing job stream (see the SUBMIT command) for the purpose of shutting down the system without operator intervention at the end of processing. The command can also be invoked or called from within other utility programs for the same purpose.

Remarks:

Be sure that all data files opened in the course of processing have been closed before invoking the POW command. To restore power to the APC, turn the main power switch to the OFF position, then to the ON position.

THE REN (RENAME) BUILT-IN COMMAND

Syntax:

REN {*d:*}*newname*{*.typ*} = *oldname*{*.typ*}

Type:

Built-in

Purpose:

The REN Built-in command lets you change the name of a file in the directory of a floppy diskette or hard disk.

The *oldname* in the filespec identifies an existing file. The *newname* is a filename not currently in the directory. The REN command changes the name of the file in the directory from *oldname* to *newname.*

Remarks:

REN does not make a copy of the file nor does it change the contents of the file. REN only changes the name of the file in the directory. REN does not allow the use of wildcard characters.

You can include a drive specifier as part of the *newname* specification. If you also use a drive specifier as a part of *oldname* filespec, it must be the same as any drive specifier included as part of *newname*. If you omit the drive specifier in the command line, REN assumes the file to be renamed is on the current default drive.

If the file identified by *oldname* is not found in the directory, REN displays the following message:

```
NO FILE
```

If the file identified by *newname* already exists in the directory, REN displays the following message:

```
FILE EXISTS
```

Examples:

```
A>REN NEWASM.A86=OLDFILE.A86
```

The file OLDFILE.A86 will be renamed NEWASM.A86 in the directory on Drive A.

```
B>REN A:X.PAS = Y.PLI
```

The file Y.PLI located on Drive A will be renamed to X.PAS.

```
A>REN B:NEWLIST=B:OLDLIST
```

The file OLDLIST changes to NEWLIST on Drive B. Note that because a drive specifier (B:) was included with the *newname* filespec, the drive specifier in the *oldname* filespec is unnecessary. The command line above is equivalent to the following command.

```
A>REN B:NEWLIST=OLDLIST
```

THE SETCOM (SET COMMUNICATIONS) COMMAND

Syntax:

```
SETCOM
SETCOM {speed, length, parity, stop bit}
SETCOM,
```

Type:

Transient Utility

Purpose:

The SETCOM Utility sets the working conditions of the RS-232C serial interface of the APC. Parameters are set according to the requirements of the device with which the APC is communicating. The parameters can be invoked directly in the command line as shown in the second form above, or after SETCOM is loaded using the first form of the command line. The third form automatically sets the default parameters described.

Remarks:

The SETCOM parameter options control four areas affecting communication through the RS-232C interface. Options must be entered in the order shown above with commas separating the parameters. If a parameter is omitted, the comma must be entered with a blank for the option. In this case, the default value is automatically assigned.

• *speed*	Sets the baud rate (bits/second) of the data transfer. Any of the following values is acceptable: 200, 300, 600, 1200, 2400, 9600, or 19200. The default speed is 300 bits/second.
• *length*	Sets the length (in bits) of the data word being transferred. Any of the following values is acceptable: 5, 6, 7, or 8. The default word length is 7 bits.
• *parity*	Sets the type of parity required for data being transferred. Even parity is set by entering E; odd parity is set by entering O; no parity check is performed by entering N. The default parameter for parity is E.
• *stop bit*	Sets the number of stop bits required during transfers. Two values, 1 and 2, are acceptable. The default parameter is 1.

Examples:

```
A>SETCOM 2400, 8, 0, 2
```

This command line sets a baud rate of 2400 bits/second for transfer of data words 8 bits long. No parity check is required, but 2 stop bits are required.

```
A>SETCOM 2400, 8, , 2
```

This command line sets the same conditions for serial transfer as the previous example, except the parity is set to E by using the default value (represented by the blank).

THE STAT (STATUS) COMMAND

Syntax:

```
STAT
STAT d:=RO
STAT filespec {RO | RW | SYS | DIR | SIZE}
STAT {d:}DSK: | USR:
STAT VAL: | DEV:
```

Type:

Transient Utility

Purpose:

The various forms of the STAT Utility provide you with information about the APC peripheral devices, disk attributes, and files. With STAT you can change the attributes of files and drives. You can also use STAT to assign or reassign physical devices to a variety of logical device names.

Note that the options available for the third form shown above can be denoted in three ways: l) precede the option with a dollar sign - $; 2) enclose the option in square brackets - []; 3) enter the option without any delimiters, as shown above.

Remarks:

The notation RW indicates that a drive is in the Read-Write state; that is, data can be both read from and written to the drive.

The notation RO indicates that a drive or disk is in the Read-Only state; that is data can only be read from but not written to the disk or drive.

The default state of all drives is RW and becomes RO either when you set it using STAT or when you change a disk without pressing CTRL-C.

Setting a Drive To Read-Only Status

Syntax:

STAT *d*: = RO

Purpose:

Use this form of the STAT command to set a particular drive (*d*:) to the Read-Only status. CP/M-86 will not permit data to be written to the disk in the drive until the Read-Write attribute is restored. Use CTRL-C to reset the drive to RW status.

Example:

```
A>STAT B:=RO
```

This command line sets Drive B to Read-Only status.

Determining Free Space on Disk

Syntax:

STAT {*d*:}

Purpose:

When you enter the command STAT without a command tail, the amount of free storage space available on all online disks is reported on the screen. However, note that CP/M-86 recognizes disks as online for the STAT command only if the drive has been accessed since CP/M-86 was last started or reloaded.

To find the amount of free space on a particular diskette or disk, include the drive specifier in the command tail.

Remarks:

If the drive specifier names a drive that was not previously online, CP/M-86 automatically places the drive in an online status.

Using the form of the STAT command shown above displays information on the screen as follows:

d: RW, Free Space: *nn*K

where *d* identifies the drive being reported, *nn* is the number of kilobytes of unused storage space on the disk in the drive, and RW is the current attribute (in this case, Read-Write) of the drive.

Examples:

```
A>STAT
```

This command reports the free space and attribute status of all online disks. If two disks are online, the above command causes a display similar to the following to appear.

```
A: RW,  Free Space:  16K
B: RO,  Free Space:  32K
```

The display indicates that Drive A contains a disk with 16 kilobytes of free space remaining available. The current attribute of the disk is Read-Write. Drive B contains a disk with 32 kilobytes of free space; however, the current state of the disk is Read-Only.

```
A>STAT B:
```

If Drive B is set to Read-Only and contains a disk with 98 kilobytes of free storage space, the following message appears on the screen:

```
B: RO, Free Space: 98K
```

Determining the Size and Attributes of Files

Syntax:

STAT *filespec* {SIZE}

Purpose:

This form of the STAT command displays the amount of space (in kilobytes) used by the file specified. It also reports the current attributes of the file.

STAT accepts wildcard characters in this form of the command. When you use wildcards, STAT displays a list of the file characteristics and attributes of all qualifying files from the default or specified drive in alphabetical order.

The STAT command provides information about the attributes of each file. These attributes, described below, determine how access to the file is restricted or allowed.

Table 3-8 File Attributes Displayed by STAT

ATTRIBUTE	DESCRIPTION
RO	Read-Only. Data can be read from the file, but the file cannot be altered.
RW	Read-Write. Data can be read from or written to the file.
SYS	System File. System files do not appear in directory (DIR) listings. System files can be listed using the DIRS command and appear in parentheses during STAT inquiries.
DIR	Directory File. Directory files appear in directory (DIR) listings.

All files have either the RO or RW attribute, and either the SYS or DIR attribute. Unless changed by the STAT command, all files have default attributes of RW and DIR.

This format of the STAT command produces a list of file characteristics under five headings:

Recs	The first column displays the number of records used by the file, where each record is 128 bytes long.
Bytes	The second column displays the number of kilobytes used by the file, where each kilobyte contains 1,024 bytes.
FCBs	The third column displays the number of file control blocks, directory entries, used by the file.
Attributes	The fourth column displays the two attributes currently active for the file.
Name	The fifth column displays the drive specifier, filename, and filetype of each qualifying file listed.

If you include the SIZE option following the filespec in the command line, an additional column labeled Size appears before the Recs column. The Size column displays information about the actual amount of space used by the file. Note that you can include the SIZE parameter in the command line in three ways: 1) precede the option with a dollar sign - $; 2) enclose the option in square brackets - []; 3) enter the option without any delimiters.

Remarks:

Use the SIZE option to compute and display the *virtual file size* of each file. The virtual and real file sizes are identical for sequential files, but can differ for random access files. The value shown in the Size column represents virtual file size: the number of filled and unfilled records allotted to the file.

Whenever you specify a wildcard character in a STAT file inquiry, STAT *.* for example, STAT performs a complete directory verification. In this procedure, STAT checks the directory to ensure that two files do not share the same disk space allocation. If STAT finds a duplicate space allocation, it displays the following message:

> Bad Directory on *d:*
> Space Allocation Conflict:
> User *nn d:*filename.typ

STAT displays the user number and the name of the file containing doubly allocated space. More than one file can appear. Because this message indicates an error condition, the recommended solution is to erase the files listed and press CTRL-C.

Examples:

```
A>STAT MY*.*
```

This command tells STAT to display the characteristics of all files that begin with the letters MY. The following is an example of the type of display that appears.

```
Drive B:                                  User 0
Recs     Bytes     FCBs     Attributes    Name

 16        2K        1       Dir RW     B:MYPROG   .A86
  8        1K        1       Dir RO     B:MYTEST   .DAT
 32       18K        2       Sys RO     B:MYTRAN   .CMD

Total:    21K        4

B: RW, Free Space: 90K
```

```
A>STAT MY*.* SIZE
```

This command displays the same information as for the previous example, but also includes the Size column as shown below.

```
Drive B:                                   User 0
Size  Recs  Bytes  FCBs  Attributes        Name

 16    16    2K     1     Dir RW     B:MYPROG   .A86
  8     8    1K     1     Dir RO     B:MYTEST   .DAT
 21    32   18K     2     Sys RO     B:MYTRAN   .CMD

Total:      21K     4

B: RW, Free Space: 90K
```

Setting File Attributes

Syntax:

STAT *filespec* RO | RW | SYS | DIR

Purpose:

This form of the STAT command allows you to set or reset the access mode attributes of one or more files. Note that you can include the options following *filespec* in the command line in three ways: 1) precede the option with a dollar sign -$; 2) enclose the option in square brackets - [ ]; 3) enter the option without any delimiters, as shown above.

Any of the four attributes — RO, RW, SYS, and DIR — can be set or reset in this way.

Remarks:

If the drive named in the file specification corresponds to an inactive drive, CP/M-86 first places the drive in the online state.

Files can have either the RO or RW attribute, but not both. Similarly, files can be marked either DIR or SYS, but not both.

Examples:

```
A>STAT LETTER.TXT RO
```

This command sets the file named LETTER.TXT on the default drive to Read-Only (RO). The message "LETTER.TXT set to Read Only (RO)" appears to confirm the operation.

```
A>STAT A:*.CMD SYS
```

This command gives all files on Drive A that have a filetype CMD the SYS attribute, making them system files. If the files PIP.CMD, ED.CMD, and ASM86.CMD exist on Drive A, STAT displays the following messages.

```
PIP.CMD set to SYS
ED.CMD set to SYS
ASM86.CMD set to SYS
```

Displaying Disk Status

Syntax:

STAT {*d*:}DSK:

Purpose:

This form of the STAT command displays internal information about your disk system for all online disks. If a drive is specified, information for that drive only is displayed. If the drive specified is not currently online, STAT places the drive online and provides the information.

Remarks:

The information provided by this form of the STAT command is generally useful for more advanced programming functions and is unnecessary for the regular use of CP/M-86.

Examples:

```
A>STAT DSK:
```

This STAT command displays information about Drive A in the following format. The character *n* represents values supplied by STAT.

```
    A:  Drive Characteristics
nnnn:  128 Byte Record Capacity
nnnn:  Kilobyte Drive Capacity
nnnn:  32 Byte Directory Entries
nnnn:  Checked Directory Entries
nnnn:  128 Byte Records/Directory Entry
nnnn:  128 Byte Records/Block
nnnn:  128 Byte Records/Track
nnnn:  Reserved Tracks
```

Information for any drive can be obtained by using the drive specifier in the command line.

Displaying Active User Numbers

Syntax:

STAT {*d*:} USR:

Purpose:

This form of the STAT command displays a list of all user numbers that contain files on the default or specified drive. The command indicates those user numbers currently active on the disk.

Examples:

```
A>STAT USR:
```

This command displays the user numbers on the disk in Drive A that contain active files.

Displaying STAT Commands and Device Names

Syntax:

```
STAT VAL:
```

Purpose:

The STAT VAL: command displays a list of the general form of the STAT commands. It also displays the physical device names that can be assigned to each of the four CP/M-86 logical device names.

Examples:

The STAT VAL: command line entry and display are shown below.

```
A>STAT VAL:
STAT 2.2
Read Only Disk: d:=RO
Set Attribute: d:filename.typ[ro][rw][sys][dir]
Disk Status : DSK: d:DSK:
User Status : USR: d:USR:
Iobyte Assign:
CON: = TTY: CRT: BAT: UC1:
AXI : = TTY: PTR: UR1:  UR2:
AXO: = TTY: PTP: UP1:  UP2:
LST  : = TTY: CRT: LPT:  UL1:
A>
```

Displaying and Setting Physical to Logical Device Assignments

Syntax:

STAT DEV:
STAT *logical device*: = *physical device:*

Purpose:

STAT DEV: displays the current assignments for the four CP/M-86 device names: CON:, RDR:, PUN:, LST:. Use the second form of the STAT command shown above to change the physical to logical device assignments.

The physical device names available are displayed using the STAT VAL: command. Each of the physical device names listed under the heading "Iobyte Assign" can be assigned to a logical device name.

When you assign a physical device to a logical device, STAT assigns a value from 0 to 3 to the logical device name in the area known as the IOBYTE. Thereafter, when a program needs to access a physical device, the logical device assignment in the IOBYTE directs the operation.

Any of the listed physical device names can be assigned to the corresponding logical device names. However, unless the peripheral being identified is properly connected or cabled to the APC, the assignment cannot be properly accessed by CP/M-86.

Remarks:

For additional information on how physical device drivers are defined in the CBIOS (Customized Basic I/O System) and how the IOBYTE is read and interpreted, see the *CP/M-86 System Reference Guide* for the APC.

Examples:

```
A>STAT CON: = CRT:
```

The command above assigns the physical device name CRT: to the logical input device name CON:, which generally refers to the console.

```
A>STAT LST: = LPT:
```

The command above assigns the physical device name LPT: to the logical device name LST:, which generally refers to the printer.

THE SUBMIT (BATCH PROCESSING) COMMAND

Syntax:

SUBMIT *filespec {parameters...}*

Type:

Transient Utility

Purpose:

The SUBMIT Utility allows you to group sets of commands together for automatic processing by CP/M-86.

Normally, you enter one command at a time for CP/M-86 to interpret and process. At times, however, it is advantageous to batch several commands that can be processed sequentially, usually without further operator intervention.

To do this, create a file (with ED or a word-processor) in which the sequence of commands and any parameter values required are listed. The file is identified by its filename, but it must have a filetype of SUB.

When you issue the SUBMIT command, SUBMIT reads the SUB file named by *filespec* and prepares the contents of the file for interpretation by CP/M-86.

SUB files can contain any valid CP/M-86 commands. Although SUB files cannot contain nested SUBMIT commands, the last command in a SUB file can be another SUBMIT command, allowing chaining to other SUB files.

SUBMIT allows you to pass up to nine parameters from the entry at the command line to the operation being performed. However, the SUB file being processed will only accept the parameter values if it contains valid variable names. Each variable name is identified by a dollar sign ($) followed by a number from 1-9. The following are the valid variable names that can be included in a SUB file:

$1	$4	$7
$2	$5	$8
$3	$6	$9

These parameter variable names can be located anywhere within the SUB file. The SUBMIT Utility reads the command line following the filespec and substitutes the actual parameter values you type for the variables in the SUB file. SUBMIT then creates a file named $$$.SUB that contains the command lines that result from the substitutions. After the substitutions occur, SUBMIT sends the SUB and $$$.SUB files to CP/M-86 for processing. Note, however, that parameters cannot be passed to utilities like DDT-86 using SUBMIT.

Remarks:

The parameter values entered in the command tail to be passed to SUBMIT for substitution and processing can be any alphabetic, numeric, or special characters. The items are separated in the command line by one or more blanks.

The first parameter value in the command tail takes the place of $1, the second value replaces $2, and so forth. If you enter more values in the command tail than parameters in the SUB file, the extra parameters are ignored. If you enter fewer values than parameters in the SUB file, the variable name parameter ($1, $2, $3, and so forth) is processed as the actual value.

Batch command processing stops after the last line of the SUB file is processed. PressingCTRL -C stops the SUBMIT process. You can also stop batch processing before the end of the SUB file is reached by pressing any key as CP/M-86 displays the command input prompt, A>.

To include an actual dollar sign ($) in a SUB file, type two dollar signs ($$). The SUBMIT Utility interprets them as (and replaces them with) a single dollar sign when command tail parameter values are substituted.

The $$$.SUB file is automatically deleted after CP/M-86 has processed all command lines.

Examples:

```
A>SUBMIT SUBFILE
```

The above command submits the file named SUBFILE.SUB located on Drive A for batch processing. Assume that SUBFILE.SUB contains the following command lines:

```
DIR *.CMD
ASM86 X $$SB
PIP LST:=X.PRN[T8D80]
```

The SUBMIT command shown above sends these commands to CP/M-86 for processing. CP/M-86 first displays a directory of all files on the default drive having a filetype of CMD. Next, the file named X.A86 is assembled with the output SYM file being sent to Drive B. (Note the two dollar signs used to represent a single dollar sign in the command line.) Finally, CP/M-86 transfers the file named X.PRN to the listing device, expanding the tabs and deleting characters beyond column 80.

```
A>SUBMIT B:ASMCOM X 8 D80 SZ
```

The SUBMIT Utility first assigns the command tail values to variable names as follows and stores them in the temporary file named $$$.SUB.

```
X    =  $1
8    =  $2
D80  =  $3
SZ   =  $4
```

SUBMIT then locates the file ASCOM.SUB located on Drive B. Assume the file contains the following command lines:

```
ERA $1.BAK
ASM86 $1 $$$4
PIP LST:=  $1.PRN[t$2 $3 $5]
```

The SUBMIT Utility reads the file and processes each command line using the substituted values as follows:

```
ERA X.BAK
ASM86 X $SZ
PIP LST:= X.PRN[T8 D80]
```

Each of these commands is processed sequentially by CP/M-86.

THE TIMEROFF COMMAND

Syntax:

```
TIMEROFF
```

Type:

Transient Utility

Purpose:

The TIMEROFF Utility turns off the dynamic update of the time and date in the APC status line at the top of the screen. It also disables the keyboard repeat function. The TIMEROFF program can be run prior to programs in which the interrupt caused by the update of the status line display results in significant delays in processing. If you are using the RS-232C interface of the APC for serial communications, run TIMEROFF prior to those communications where timing below approximately 200 milliseconds occurs. Always use TIMEROFF prior to running any communications software that supports interrupts. Although the status line display is not updated following TIMEROFF, the internal clock/calendar of the APC continues to correctly store the time and date. The dynamic display of the correct time and date and operation of keyboard repeat are restored by the TIMERON Utility.

THE TIMERON COMMAND

Syntax:

```
TIMERON
```

Type:

Transient Utility

Purpose:

The TIMERON Utility restores the dynamic display of the time and date in the APC status line. TIMERON counteracts the effect of the TIMEROFF Utility.

Remarks:

When TIMERON is invoked at the command line as shown above, the correct time and date appear on the APC status line, regardless of the interval between running TIMEROFF and TIMERON.

THE TOD (SET TIME OF DAY) COMMAND

Syntax:

```
TOD
```

Type:

Transient Utility

Purpose:

The TOD Utility allows you to set the system time and date consisting of the following: day of the week, month, day of the month, year, hours, minutes, and seconds. All of these values are displayed in the APC status line at the top of the screen.

Remarks:

The day of the week is set by entering a number from 1, representing Sunday, to 7, representing Saturday. The number entered is interpreted internally and the appropriate weekday name appears in the status line. Time is represented as a 24-hour clock with the values for morning hours ranging from 00 to 11, and for afternoon hours ranging from 12 to 23. The values for month (1-12) and day (1-31) are standard. Enter two digits only for the year.

Parameters must be entered as shown below with the appropriate symbols inserted where shown. Commas must separate the three groups.

```
day of week, month/day/year, hour:minute:second
```

Example:

To invoke the TOD Utility, enter the following at the command line:

```
A>TOD
```

The following display appears, prompting you for date/time entry.

```
ENTER ---- # (day of week),mm/dd/yy,hh:mm:ss
```

Assume that the date is Wednesday, August 25, 1982 and that the time is 1:50 PM. The following represents the values entered to set the time and date.

```
4, 08/25/82, 13:50:00
```

If the parameters are correctly entered, the TOD Utility ends, the system prompt appears, and the APC status line reflects the entries.

THE TYPE (DISPLAY FILE) BUILT-IN

Syntax:

```
TYPE {d:}filename{.typ}
```

Type:

Built-in

Purpose:

The TYPE Built-in command displays the contents of a character file on the screen.

Remarks:

TYPE should only be used to display files containing character data. No wildcard characters can be included in the file specification. Tab characters occurring in the file named by the file specification are automatically expanded to every eighth position on the screen.

To end the file listing, press any key on the keyboard. The system prompt for the default disk reappears.

If the file named by the file specification is not present on an online disk, TYPE displays the following message:

```
NO FILE
```

To simultaneously list the file on the printer as well as on the screen, press CTRL-P before entering the TYPE command line. To disable the printer echo of the display, press CTRL-P again at the system prompt.

Examples:

```
A>TYPE MYPROG.A86
```

This command displays the contents of the file MYPROG.A86 on the screen.

```
A>TYPE B:THISFILE
```

This command displays the contents of the character file named THISFILE located on Drive B.

THE USER (DISPLAY AND SET USER NUMBER) BUILT-IN

Syntax:

```
USER {n}
```

Type:

Built-in

Purpose:

Each disk directory can be divided into as many as 16 distinct groups known as user numbers. Files can be assigned to particular user numbers for the purpose of security or organization. The USER Built-in command displays and changes the currently active user number.

Remarks:

When CP/M-86 is first loaded, the active user number is 0. Any files you create under this user number are not generally accessible under any other user numbers. However, you can gain access to files in user number 0 through the [G*n*] option of the PIP command or by using the system file (SYS) attribute assigned with the STAT command.

To display the currently active user number, enter:

```
USER
```

To change the currently active user number, enter:

```
USER n
```

where *n* is a number in the range 0-15.

To display a list of all user numbers on disk that have files associated with them, enter:

```
STAT USR:
```

Examples:

```
A> USER
0
```

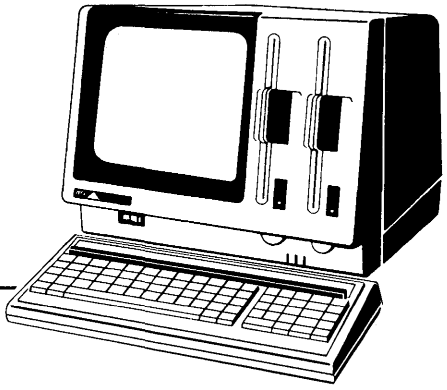
This command displays the current user number, in this case, 0.

```
A> USER 3
```

This command changes the current user number to 3. All files associated with that number can then be accessed for processing.

Chapter 4

ED, The CP/M-86 Editor



INTRODUCTION TO ED

To do almost anything with a computer you need some way to enter data, some way to give the computer the information you want it to process. The programs most commonly used for this task are called editors. Editors transfer what you enter at the keyboard to a disk file. CP/M-86's editor is named ED. Using ED, you can easily create and alter CP/M-86 text files.

The correct command syntax for invoking the CP/M-86 editor is given in the next section. After starting ED, you issue commands that transfer text from file to memory for editing. The section titled ED Operation details this operation and describes the basic text transfer commands that allow you to easily enter and exit the editor.

The Basic Editing Commands section details the commands that edit a file. Combining ED Commands describes how to combine the basic commands to edit more efficiently. Although you can edit any file with the basic ED commands, ED provides several more commands that perform more complicated editing functions, as described in Advanced ED Commands.

During an editing session, ED may return two types of error messages. ED Error Messages lists these messages and provides examples of how to recover from common editing error conditions.

STARTING ED

Syntax:

ED *filespec*
ED *filespec d*:
ED *filespec filespec*

To start ED, enter its name after the CP/M-86 prompt. The command ED must be followed by a file specification containing no wildcard characters, such as:

```
A>ED MYFILE.TEX
```

The file specification, MYFILE.TEX in the above example, names the file to be edited or created. The file specification can be preceded by a drive specifier but a drive specifier is unnecessary if the file to be edited is on your default drive. Optionally, the file specification can be followed by a drive specifier, as shown in the following example.

```
A>ED MYFILE.TEX B:
```

In response to this command, ED opens the file to be edited, MYFILE.TEX, on Drive A, but sends all the edited material to a file with the same name on Drive B.

Optionally, you can send the edited material to a file with a different filename, as shown in the following example.

```
A>ED MYFILE.TEX YOURFILE.TEX
```

If the output file named in the command line already exists, ED prints the following message and terminates.

```
Output File Exists, Erase It
```

After you enter a valid command line, the ED prompt, *, appears on the screen to indicate that ED is ready to accept a command.

If no previous version of the source file exists on the current disk, ED automatically creates a new file and displays the following message:

```
NEW FILE
: *
```

NOTE

Before starting an editing session, use the STAT command to check the amount of free space on your diskette or disk. Make sure that the unused portion of a disk is at least as large as the file you are editing — larger if you plan to add characters to the file. When ED finds that a diskette, a disk, or a directory is full, it has only limited recovery mechanisms. These are explained in the section called ED Error Messages.

ED OPERATION

All text that you enter or change with ED must pass through a memory buffer. When you start ED and the asterisk prompt appears, this memory buffer is empty. Based on the command you then enter, ED reads segments of the source file, for example MYFILE.TEX, into the memory buffer for you to edit. If the file is new, you must insert text into the file before you can edit. During the edit, ED writes the edited text into a temporary work file (for example MYFILE.$$$).

When you end the edit, ED writes the memory buffer contents to the temporary file and appends any text remaining in the source file. ED then changes the name of the source file from MYFILE.TEX to MYFILE.BAK, so you can reclaim the original material from the backup file if necessary. Lastly, ED renames the temporary file, MYFILE.$$$, to MYFILE.TEX, the new edited file.

The following figure illustrates the relationship between the source file, the temporary work file and the new file.

NOTE

When you invoke ED with two filespecs, an input file and an output file, ED does not rename the input file to type .BAK; therefore, the input file can be Read-Only or on a write-protected diskette if the output file is written to another diskette or disk.

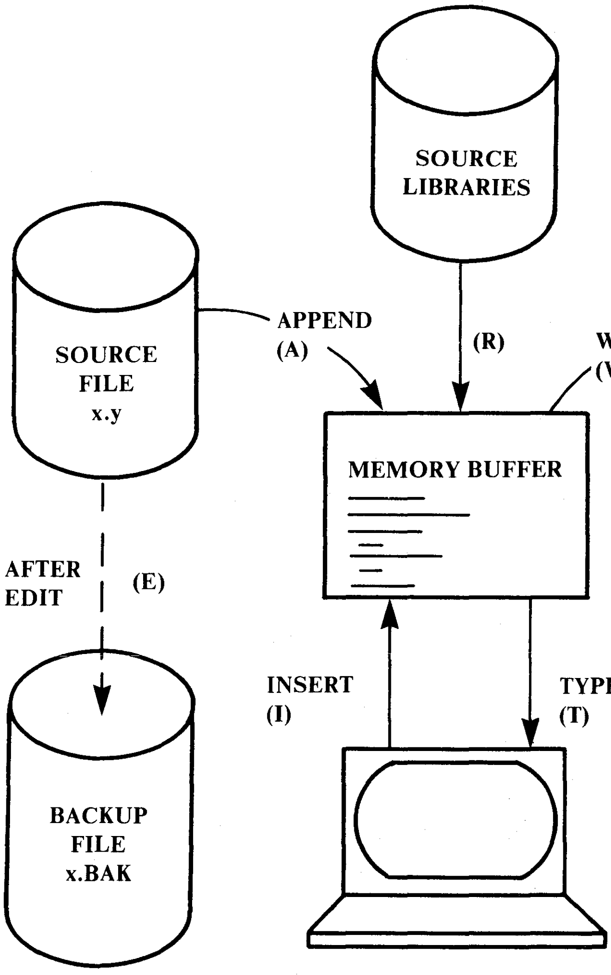


Figure 4-1 Overall ED Operations

In the figure above, the memory buffer is logically between the source file and the temporary work file. ED supports several commands that transfer lines of text between the source file, the memory buffer and the temporary (and eventually final) file. The following table lists the three basic text transfer commands that allow you to enter the editor, write text to the temporary file, and exit the editor.

Table 4-1 Text Transfer Commands

COMMAND	RESULT
*n*A	Append the next *n* unprocessed source lines from the source file to the end of the memory buffer.
*n*W	Write the first *n* lines of the memory buffer to the temporary file free space.
E	End the edit. Copy all buffered text to the temporary file, and copy all unprocessed source lines to the temporary file. Rename files.

Appending Text into the Buffer

When you start ED and the memory buffer is empty, you can use the A (append) command to add text to the memory buffer.

NOTE

ED can number lines of text to help you keep track of data in the memory buffer. The colon that appears when you start ED indicates that line numbering is turned on. Type -V after the ED prompt to turn the line number display off. Line numbers appear on the screen but never become a part of the output file.

THE A (APPEND) COMMAND

The A command appends (copies) lines from an existing source file into the memory buffer. The form of the A command is:

```
nA
```

where *n* is the number of unprocessed source lines to append into the memory buffer. If a pound sign, #, is given in place of *n*, then the integer 65535 is assumed. Because the memory buffer can contain most reasonably sized source files, it is often possible to issue the command #A at the beginning of the edit to read the entire source file into memory.

If *n* is 0, ED appends the unprocessed source lines into the memory buffer until the buffer is approximately half full. If you do not specify *n*, ED appends only one line from the source file into the memory buffer.

Exiting ED

You can use the W (Write) command and the E (Exit) command to save your editing changes. However, the W command writes lines from the memory buffer to the new file without ending the ED session. An E command saves the contents of the buffer and any unprocessed material from the source file and exits ED.

THE W (WRITE) COMMAND

The W command writes lines from the buffer to the new file. The form of the W command is:

```
nW
```

where *n* is the number of lines to be written from the beginning of the buffer to the end of the new file. If *n* is greater than 0, ED writes *n* lines from the beginning of the buffer to the end of the new file. If *n* is 0, ED writes lines until the buffer is half empty. The 0W command is a convenient way of making room in the memory buffer for more lines from the source file. You can determine the number of lines to write out by executing a 0V command to check the amount of free space in the buffer, as shown below:

```
1: *0V
25000/30000
1: *
```

The above display indicates that the total size of the memory buffer is 30,000 bytes and there are 25,000 free bytes in the memory buffer.

NOTE

After a W command is executed, you must enter the H command to reedit any lines previously saved during the current editing session.

THE E (EXIT) COMMAND

An E command performs a normal exit from ED. The form of the E command is:

```
E
```

followed by RETURN.

When you enter an E command, ED first writes all data lines from the buffer and any unprocessed source file lines to the new file. If a .BAK file exists, ED deletes it, then renames the original file with the .BAK filetype. Finally, ED renames the new file from filename.$$$ to the original filetype and returns to the system prompt.

The operation of the E command makes it unwise to edit a backup file. When you edit a BAK file and exit with an E command, ED erases your original file because it has a .BAK filetype. To avoid this, always rename a backup file to some other filetype before editing it with ED.

NOTE

Any command that terminates an ED session must be the only command on the line.

BASIC EDITING COMMANDS

The text transfer commands discussed above allow you to enter and exit the editor. This section describes the basic commands that you use to edit a file.

ED treats a file as a long chain of characters grouped together in lines. ED displays lines and allows you to edit characters and lines in relation to an imaginary device called the character pointer (CP). During an editing session, you issue commands that move the location of the CP in the memory buffer.

The following commands move the character pointer or display text in the vicinity of the CP. These ED commands consist of a numeric argument and a single command letter followed by RETURN. The numeric argument, *n*, determines the number of times ED executes a command; however, there are four special cases to consider regarding the numeric argument:

- If the numeric argument is omitted, ED assumes an argument of 1.
- Use a negative number if the command is to be executed backwards through the memory buffer. (The B command is an exception.)

- If you enter a pound sign, #, in place of a number, ED uses the value 65535 as the argument. A pound sign argument can be preceded by a minus sign to cause the command to execute backwards through the memory buffer (-#).
- ED accepts 0 as a numeric argument only in certain commands. In some cases, 0 causes the command to be executed approximately half the possible number of times, while in other cases it prevents the movement of the CP.

The following table lists the basic editing commands in alphabetical order.

Table 4-2 Basic Editing Commands

COMMAND	ACTION
B, -B	Move CP to the beginning (B) or end (-B) of the memory buffer.
*n*C, -*n*C	Move CP *n* characters forward (*n*C) or backward (-*n*C) through the memory buffer.
*n*D, -*n*D	Delete *n* characters before (-*n*D) or after (*n*D) the CP.
I	Enter insert mode.
I*string*CTRL-Z	Insert a string of characters (terminated by CTRL-Z).
*n*K, -*n*K	Delete (kill) *n* lines before the CP (-*n*K) or after the CP (*n*K).
*n*L, -*n*L	Move the CP *n* lines forward (*n*L) or backward (-*n*L) through the memory buffer.
*n*T, -*n*T	Type *n* lines before the CP (-*n*T) or after the CP (*n*T).
n, -,*n*	Move the CP *n* lines before the CP (-*n*) of after the CP (*n*) and display the destination line.

The following sections discuss these basic editing commands in more detail. The examples illustrate how the commands affect the position of the character pointer in the memory buffer. The imaginary character pointer is represented by the symbol ↑ in the examples. This character does not actually appear in the buffer.

Moving the Character Pointer

This section describes commands that move the character pointer in useful increments but do not display the destination line. Although ED is used primarily to create and edit program source files, the following sections present a simple text as an example to make ED easier to learn and understand.

THE B (BEGINNING/BOTTOM) COMMAND

The B command moves the CP to the beginning or bottom of the memory buffer. The following forms of the B command are allowed:

```
B, -B
```

-B moves the CP to the end or bottom of the memory buffer; B moves the CP to the beginning of the buffer.

THE C (CHARACTER) COMMAND

The C command moves the CP forward or backward the number of characters specified. The following forms of the C command are allowed:

*n*C, -*n*C

where *n* is the number of characters the CP is to be moved. A positive number moves the CP toward the end of the line and the bottom of the buffer. A negative number moves the CP toward the beginning of the line and the top of the buffer. You can enter an *n* large enough to move the CP to a different line. However, each line is separated from the next by two invisible characters: a carriage return and a line feed represented by <cr><lf>. You must compensate for their presence.

For example, assume the character pointer is at the first line of the buffer as shown below.

```
↑Emily Dickinson said,<cr><lf>
"I find ecstasy in living -<cr><lf>
```

The command 30C moves the CP to the next line, counting the <cr> and <lf> each as characters.

```
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 an L command, the CP always points to the beginning of a line. The following forms of the L command are allowed:

*n*L, -*n*L

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

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

The command -L moves the CP to the beginning of the previous line, even if the CP originally points to a character in the middle of the line. Use the special character 0 to move the CP to the beginning of the current line.

THE *n* (NUMBER) COMMAND

The *n* command moves the CP and displays the destination line. The following forms of the *n* command are allowed:

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 prints only the destination line. For example, assume the CP is positioned at the beginning of the second line as shown.

```
   Emily Dickinson said,<cr><lf>
 ↑ "I find ecstasy in living -<cr><lf>
```

Entering the number 2 would result in the following:

```
 ↑ is joy enough."
```

The CP is positioned at the beginning of the destination line, which is displayed.

A further abbreviation of this command is to enter no number at all. In response to RETURN, ED assumes an *n* command of 1 and moves the CP down to the next line and prints it. Similarly, a minus sign (-) entered without a number moves the CP back one line.

Displaying Memory Buffer Contents

ED does not display the contents of the memory 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 types a specified number of lines from the CP at the screen. The forms of the T command are:

*n*T, -*n*T

where *n* specifies the number of lines to be displayed. If you enter a negative number, ED displays *n* lines before the CP. A positive number displays *n* lines after the CP. If no number is specified, ED types from the character pointer to the end of the line. The CP remains in its original position no matter how many lines are typed. For example, if the character pointer is at the beginning of the memory buffer, and you enter the command 4T, four lines are displayed on the screen, but the CP stays at the beginning of line 1.

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

If the CP is between two characters in the middle of the line, entering the T command with no number specified types only the characters between the CP and the end of the line. However, the character pointer stays in the same position, as shown in the memory buffer example below.

```
"I find ec↑stasy in living -
```

Whenever ED is displaying text in response to the T command, you can enter a CTRL-S to stop the display, then a CTRL-Q when you're ready to continue scrolling. Enter a CTRL-C to abort long listings.

Deleting Characters

THE D (DELETE) COMMAND

The D command deletes a specified number of characters and has the 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, towards the bottom of the file. A negative number deletes characters to the left of the CP, towards the top of the file. If the character pointer is positioned in the memory buffer as shown below:

```
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 the six characters after the CP. The resulting memory buffer looks 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 a D command to delete the <cr><lf> between two lines to join them together. Remember that the <cr> and <lf> are two characters.

THE K (KILL) COMMAND

The K command deletes (kills) whole lines from the memory buffer. The following forms are allowed:

> *n*K, -*n*K

where *n* is the number of lines to be deleted. A positive number kills lines after the CP. A negative number kills lines before the CP. When no number is specified, ED kills the current line.

For example, assume that the character pointer is at the beginning of the second line in the following:

```
 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 -K deletes the previous line and the memory buffer changes 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 before the CP with the next line. A -K command deletes all the characters between the beginning of the previous line and the CP. A 0K command deletes the characters on the line up to the CP.

You can use the special # character to delete all the text from the CP to the beginning or end of the buffer. Be careful when using #K because you cannot reclaim lines after they are removed from the memory buffer.

Inserting Characters into the Memory Buffer

THE I (INSERT) COMMAND

To insert characters into the memory buffer by typing them at the keyboard, use the I command. The I command takes the following forms:

> I
> I*string* CTRL-Z

When you type the first command, ED enters insert mode. In this mode, all keystrokes are added directly to the memory buffer. ED enters characters in lines and does not start a new line until you press RETURN.

```
A>ED B: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: CTRL-Z
     : *
```

NOTE

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

In command mode, you can use CP/M-86 line editing control characters to edit your input. The table below lists these control characters.

Table 4-3. CP/M-86 Line Editing Controls

COMMAND	RESULT
CTRL-C	Abort the editor and return to the CP/M-86 system.
CTRL-E	Return carriage for long lines without transmitting command line to the buffer.
CTRL-H	Delete the last character typed on the current line.
CTRL-U	Delete the entire line currently being typed.
CTRL-X	Delete the entire line currently being typed. Same as CTRL-U.
CTRL-Y	Abort an ED command in progress without returning to the system prompt.

NOTE

In insert mode, you can use all of the same line editing controls except for CTRL-C and CTRL-E.

ED provides two ways to translate your alphabetic input to uppercase without affecting numbers. The first is to enter the insert command letter in uppercase: I. All alphabetic characters entered following this command, either in insert mode or as a string, are translated to uppercase. Otherwise, if you enter the insert command letter in lowercase, all alphabetic characters are inserted as typed. The second method is to enter a U command before inserting text. Uppercase translation remains in effect until you enter a -U command.

THE I*STRING* CTRL-Z (INSERT STRING) COMMAND

The second form of the I command does not enter insert mode. It inserts a character string into the memory buffer and returns immediately to the ED prompt. You can use CP/M-86's line editing control characters to edit the command string.

To insert a string, first use one of the commands that position the CP. You must move the CP to the location where you want to insert a 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. With the CP positioned correctly, enter a string to be inserted, as shown below:

```
iIn 1870, CTRL-Z
```

This inserts the phrase "In 1870, " at the beginning of the first line, and returns immediately to the ED asterisk (*) prompt. In the memory buffer, the CP appears after the inserted string, as shown below:

```
In 1870,↑Emily Dickinson said,<cr><lf>
```

Replacing Characters

THE S (SUBSTITUTE) COMMAND

The S command not only searches the memory buffer for a specified search string, but when it finds it, automatically replaces it with a new string as specified. The S command takes the following form:

*n*S*search string* CTRL-Z*new string* {CTRL-Z}

where ***n*** is the number of substitutions to make. If no number is specified, ED searches for the next occurrence of the search string in the memory buffer. For example, the command:

```
sEmily Dickinson CTRL-ZThe poet
```

searches for the first occurrence of "Emily Dickinson" and replaces it with "The poet." In the memory buffer, the CP appears after the substituted phrase, as shown below:

```
The poet ↑said,<cr><lf>
```

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

COMBINING ED COMMANDS

You can save keystrokes and editing time by combining the editing and display commands. This feature allows you to type any number of ED commands on the same line. ED executes the command string only after you press RETURN. Use CP/M-86's line editing controls to manipulate ED command strings.

When you combine several commands on a line, ED executes them in the same order they are entered, from left to right. The following restrictions apply to combining ED commands:

- The combined commands must not exceed CP/M-86's 128 character maximum.
- If the combined commands contain a character string, the line must not exceed 100 characters.
- Commands that terminate an editing session must not appear in combined commands.
- Commands, such as the I, S, J, X and R commands, that require character strings or filespecs must be either the last command on a line or must be terminated with a CTRL-Z or ESC character, even if no character string or filespec is given.

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

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, for example, 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 more characters to the left. You can use this command string to make a change at the end of the line without having to calculate the number of characters before the change. The following is an example of this command.

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

Displaying Text

A T command displays characters from the CP to the end of the line. To verify that an ED command has moved the CP correctly, you can combine the particular command with the T command to display the line. The following example combines the C command with the T command.

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

The following example combines the T and B# commands. This combination moves the CP to the top of the memory buffer then displays the contents of the entire buffer.

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

To see the entire line, you can combine an L command and a T command. Enter 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 displays the characters from the CP to the end of the 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.

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: *
```

Combine the edit command K with other ED commands to delete entire lines and verify the correction quickly, as shown below.

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

The abbreviated form of the I (insert) command makes simple textual changes. To enter and verify these changes, combine the I command string with the C command and the 0LT command string as shown below. Remember that the insert string must be terminated by a CTRL-Z.

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

ADVANCED ED COMMANDS

The basic editing commands discussed above allow you to use ED for all your editing. The following ED commands, however, enhance the CP/M-86 editor's usefulness.

Moving the CP and Displaying Text

THE P (PAGE) COMMAND

Although you can display any amount of text on the screen with a T command, it is sometimes more convenient to move through the buffer by the page, viewing whole screens of data and moving the CP to the top of each new screen at the same time. To do this, use the 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 next 23 lines following the CP and 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 movement of the CP. If you specify a negative number for *n*, P pages backwards toward 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 following form is accepted for the line number command:

```
n:
```

where *n* is the number of the destination line. The colon must follow the line number entered. 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. Take these changes into account when using the *n* command to go to a particular line.

THE :*n* (THROUGH LINE NUMBER) COMMAND

The inverse of the line number command specifies that a particular command will execute through a certain line number. You can only use this command with three ED commands: the T (type) command, the L (line) command, and the K (kill) command.

The :*n* command takes the following form:

```
:ncommand
```

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, but the command that follows it can.

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 types 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: *
```

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 specify. The N command allows ED to search through the entire source file instead of just the buffer. The J command searches for and then juxtaposes character strings.

THE F (FIND) COMMAND

The F command performs the simplest find function. Enter the F command as follows:

*n*F*string* {CTRL-Z}

where *n* is the occurrence of the string to be found. Any number you enter for *n* must be positive because ED can only search from the CP to the bottom of the buffer. If you do not enter a number, ED searches only for the next occurrence of the string in the file. In the following example, the second occurrence of the word "living" is found.

```
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 F command with a T (type) command. Note that if you follow an F command with another ED command on the same line, you must terminate the string with a CTRL-Z, as shown below.

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

It also 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/m-86 searches for CP/M-86 but fCp/m-86 searches for Cp/m-86, and will not find either CP/M-86 or cp/m-86.

If ED does not find a match for the string in the memory buffer, it issues the following message:

```
BREAK "#" AT
```

where the symbol # indicates that the search failed during the 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, the CP points to the first character after the search string. The form of the N command is:

*n*N*string*{CTRL-Z}

where *n* is the occurrence of the string to be found. If no number is entered, ED searches only for the next occurrence of the string in the file. Uppercase and lowercase entries of the N command have the same effect as for an F command. Note that if you follow an N command with another ED command, you must terminate the string with a CTRL-Z

When an N command is executed, ED searches the memory buffer for the specified string. If the search string is not found, however, no error message is issued. Instead, ED automatically empties the memory buffer by writing the searched data from the buffer into the new file. Then ED automatically performs a 0A command to fill the buffer with new, unsearched data from the source file. ED continues to search the buffer, write out data, and append new data until it either finds the string or reaches the end of the source file. If ED reaches the end of the source file, the following message is displayed.

```
BREAK "#" AT
```

Because ED writes the searched data to the new file before looking for more data in the source file, ED usually writes the contents of the buffer to the new file before finding the end of the source file and issuing the error message.

NOTE

You enter the H command to continue an editing session after the source file is exhausted and the memory buffer is emptied.

THE J (JUXTAPOSE) COMMAND

The J command searches for a string, inserts a string after it, then deletes any characters between the end of the inserted string and the beginning of a third "delete-to" string. This places the insert string directly between the search and delete-to strings with no intervening characters. The following is the form of the J command:

*nJsearch string*CTRL-Z*insert string*CTRL-Z*delete-to-string* {CTRL-Z}

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 memory buffer. In the following example, ED searches for the word "Dickinson", 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: *j DickinsonCTRL-Z told a friendCTRL-Z,
1: *01t
1:  Emily Dickinson told a friend,
1: *
```

If you combine this command with other commands, you must terminate the delete-to string with a CTRL-Z shown in the following example. If an uppercase J command letter is specified, 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 below.

```
236: SORT     LXI     H, SW     ;ADDRESS TOGGLE SWITCH
236: *j;CTRL-ZADDRESS SWITCH TOGGLE CTRL-Z CTRL-L CTRL-Z 0LT
236: SORT     LXI     H, SW     ;ADDRESS SWITCH TOGGLE
236: *
```

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

NOTE

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

THE M (MACRO) COMMAND

The 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 following shows the form of the M command:

*n*M*command string* {CTRL-Z}

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 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 memory buffer:

```
1: *mflivingCTRL-Z-6diLivingCTRL-Z0lt
2:  "I find ecstasy in Living -
3:  the mere sense of Living
BREAK "#" ATCTRL-Z
3: *
```

Because the terminator for an M command is a carriage return, an M command must be the last command on the line. Also, all character strings that appear in a macro must be terminated by CTRL-Z. If a character string ends the combined commands, it must be terminated by CTRL-Z, followed by a RETURN to end the M command.

The execution of a macro command always ends in a BREAK "#" message. This message appears 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 at the keyboard.

Moving Text Blocks

Moving blocks of text is a process requiring three steps:

1. transfer the defined lines to a temporary file;
2. delete the lines from their original location;
3. read the lines from the temporary file to their new location.

To move a group of lines from one area of your data to a temporary file, use the X command described below. The temporary file has a filetype of LIB. To delete the lines from their original location, use the K (kill) command described previously in the chapter. Finally, use the R command described below to read the block into its new location.

THE X (XFER) COMMAND

The X command takes the following forms:

*n*X
*n*X *filespec*CTRL-Z

where *n* is the number of lines from the CP toward the bottom of the buffer that are to be transferred to a temporary file. The value entered for *n* must always be a positive number. If no filename is specified, X$$$$$$$ is assumed. If no filetype is specified, LIB is assumed. 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 already created the library file, ED appends the specified lines to the end of the existing file.

Use the special character 0 as the value for *n* 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. Use the R command to retrieve the transferred lines. The R command takes the following forms:

```
R
R*filespec*{CTRL-Z}
```

If no filename is specified, X$$$$$$$ is assumed. Likewise, if no filetype is specified, LIB is assumed. R inserts the contents of 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 it 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 filename from subsequent command letters with CTRL-Z as shown in the following example.

```
1: *41
 : *RCTRL-ZB#T
1:  "I find ecstasy in living -
2:  the mere sense of living
3:  is joy enough."
4:  Emily Dickinson said,
1: *
```

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 memory buffer without ending the ED session. The H command saves the current changes with the remaining lines from the source file, then lets you reedit the file without exiting ED. The following is the form of the H command:

```
H
```

followed by 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 source file, and renames the original source file with the filetype BAK. ED then renames the new file (which had the filetype .$$$) with the original file specification. Finally, ED opens the newly renamed file as the new source file for a new edit, and new .$$$ file. When ED returns the * prompt, the CP is at the beginning of an empty memory buffer.

If you elected to send the edited material to a file other than the original using the following command line:

A>ED *filespec differentfilespec*

the H command renames the file differentfilename.$$$ to differentfilename.BAK and creates a new file of differentfilespec when you finish editing.

THE O (ORIGINAL) COMMAND

An O command abandons all changes made since the beginning of the edit and, without ending the ED session, allows you to return to the original source file for reediting. The following is the form of the O command:

```
O
```

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

```
O (Y/N)?
```

You must respond with either a Y or an N. If you press any other key, ED repeats the question. When you enter Y, ED erases the temporary file and the contents of the memory buffer. When the * prompt returns, the character pointer is at the beginning of an empty memory 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 following is the form of the Q command:

```
Q
```

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

```
Q (Y/N)?
```

You must respond with either a Y or an N. If you press any other key, ED repeats the question. When you enter Y, ED erases the temporary file, closes the source file, and returns control to CP/M-86.

NOTE

You can press CTRL-C to immediately return control to CP/M-86. This does not give ED a chance to close the source or new files, but it prevents ED from deleting any temporary files.

ED ERROR MESSAGES

ED displays one of two types of error messages: 1) an ED error message if ED cannot execute an edit command, or 2) a CP/M-86 error message if ED cannot read or write to the specified file.

The following is the form of an ED error message:

BREAK *x* AT *c*

where *x* is one of the symbols defined in the following table and *c* is the command letter at which the error occurred.

Table 4-4 ED Error Symbols

SYMBOL	MEANING
#	Search failure. ED cannot find the string specified in an F, S, or N command.
?*c*	Unrecognized command letter *c*. ED either does not recognize the indicated command letter, or an E, H, Q, or O command is not alone on its command line.

Table 4-4 ED Error Symbols (cont'd)

SYMBOL	MEANING
0	No .LIB file. ED did not find the .LIB file specified in an R command.
>	Buffer full. ED either cannot put any more characters in the memory buffer, or a string specified in an F, N, or S command is too long.
E	Command aborted. A keystroke at the keyboard aborted command execution.
F	File error. Followed by either DISK FULL or DIRECTORY FULL.

The following examples show how to recover from common editing error conditions. For example:

```
BREAK ">" AT A
```

means that ED filled the memory buffer before completing the execution of an A command. When this occurs, the character pointer is at the end of the buffer and no editing is possible. Use the 0W command to write out half the buffer or use an O or H command and reedit the file.

```
BREAK "#" AT F
```

means that ED reached the end of the memory buffer without matching the string in an F command. At this point, the character pointer is at the end of the buffer. Move the CP with a B or *n:* line number command to resume editing.

```
BREAK "F" AT F
DISK FULL
```

Use the 0X command to erase an unnecessary file on the disk or a B#Xd:buffer.sav command to write the contents of the memory buffer onto another disk.

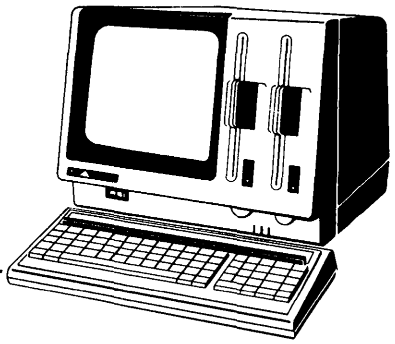
```
BREAK "F" AT n
DIRECTORY FULL
```

Use the same commands described in the previous message to recover from this file error.

The following table defines the disk file error messages ED returns when it cannot read or write a file.

Table 4-5 ED Disk File Error Messages

MESSAGE	MEANING
BDOS ERR ON *d:* RO	Disk *d:* has Read-Only attribute. For Drives A through D, this occurs is a different diskette has been inserted in the drive since the last cold or warm boot.
FILE IS READ ONLY	The file specified in the command to invoke ED has the RO attribute. ED can read the file so that you can examine it, but ED cannot change a Read-Only file.



Appendix A

ASCII and Hexadecimal Conversions

ASCII stands for American Standard Code for Information Interchange. The code contains 96 printable and 32 nonprintable characters used to store data on a disk medium (floppy diskette or hard disk). Table A-1 defines the ASCII symbols. Table A-2 lists ASCII and hexidecimal conversions. The table includes binary, decimal, hexadecimal, and ASCII conversions.

Table A-1 ASCII Symbols

SYMBOL	MEANING	SYMBOL	MEANING
ACK	acknowledge	FS	file separator
BEL	bell	GS	group separator
BS	backspace	HT	horizontal tabulation
CAN	cancel	LF	line-feed
CR	carriage return	NAK	negative acknowledge
DC	device control	NUL	null
DEL	delete	RS	record separator
DLE	data link escape	SI	shift in
EM	end of medium	SO	shift out
ENQ	enquiry	SOH	start of heading
EOT	end of transmission	SP	space
ESC	escape	STX	start of text
ETB	end of transmission	SUB	substitute
ETX	end of text	SYN	synchronous idle
FF	form-feed	US	unit separator
		VT	vertical tabulation

Table A-2 ASCII Conversion Table

BINARY	DECIMAL	HEXADECIMAL	ASCII
0000000	0	0	NUL
0000001	1	1	SOH (CTRL-A)
0000010	2	2	STX (CTRL-B)
0000011	3	3	ETX (CTRL-C)
0000100	4	4	EOT (CTRL-D)
0000101	5	5	ENQ (CTRL-E)
0000110	6	6	ACK (CTRL-F)
0000111	7	7	BEL (CTRL-G)
0001000	8	8	BS (CTRL-H)
0001001	9	9	HT (CTRL-I)
0001010	10	A	LF (CTRL-J)
0001011	11	B	VT (CTRL-K)
0001100	12	C	FF (CTRL-L)
0001101	13	D	CR (CTRL-M)
0001110	14	E	SO (CTRL-N)
0001111	15	F	SI (CTRL-O)
0010000	16	10	DLE (CTRL-P)
0010001	17	11	DC1 (CTRL-Q)
0010010	18	12	DC2 (CTRL-R)
0010011	19	13	DC3 (CTRL-S)
0010100	20	14	DC4 (CTRL-T)
0010101	21	15	NAK (CTRL-U)
0010110	22	16	SYN (CTRL-V)
0010111	23	17	ETB (CTRL-W)
0011000	24	18	CAN (CTRL-X)
0011001	25	19	EM (CTRL-Y)
0011010	26	1A	SUB (CTRL-Z)
0011011	27	1B	ESC (CTRL-[)
0011100	28	1C	FS (CTRL-\)
0011101	29	1D	GS (CTRL-])
0011110	30	1E	RS (CTRL-^)
0011111	31	1F	US (CTRL-_)
0100000	32	20	(SPACE)
0100001	33	21	!
0100010	34	22	″
0100011	35	23	#
0100100	36	24	$

Table A-2 ASCII Conversion Table (cont'd)

BINARY	DECIMAL	HEXADECIMAL	ASCII
0100101	37	25	%
0100110	38	26	&
0100111	39	27	'
0101000	40	28	(
0101001	41	29	)
0101010	42	2A	*
0101011	43	2B	+
0101100	44	2C	,
0101101	45	2D	-
0101110	46	2E	.
0101111	47	2F	/
0110000	48	30	0
0110001	49	31	1
0110010	50	32	2
0110011	51	33	3
0110100	52	34	4
0110101	53	35	5
0110110	54	36	6
0110111	55	37	7
0111000	56	38	8
0111001	57	39	9
0111010	58	3A	:
0111011	59	3B	;
0111100	60	3C	<
0111101	61	3D	=
0111110	62	3E	>
0111111	63	3F	?
1000000	64	40	@
1000001	65	41	A
1000010	66	42	B
1000011	67	43	C
1000100	68	44	D
1000101	69	45	E
1000110	70	46	F
1000111	71	47	G
1001000	72	48	H
1001001	73	49	I

Table A-2 ASCII Conversion Table (cont'd)

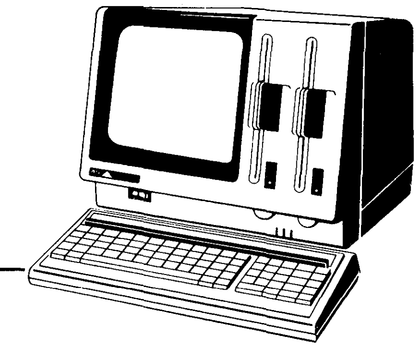
BINARY	DECIMAL	HEXADECIMAL	ASCII
1001010	74	4A	J
1001011	75	4B	K
1001100	76	4C	L
1001101	77	4D	M
1001110	78	4E	N
1001111	79	4F	O
1010000	80	50	P
1010001	81	51	Q
1010010	82	52	R
1010011	83	53	S
1010100	84	54	T
1010101	85	55	U
1010110	86	56	V
1010111	87	57	W
1011000	88	58	X
1011001	89	59	Y
1011010	90	5A	Z
1011011	91	5B	[
1011100	92	5C	\
1011101	93	5D	]
1011110	94	5E	∧
1011111	95	5F	—
1100000	96	60	'
1100001	97	61	a
1100010	98	62	b
1100011	99	63	c
1100100	100	64	d
1100101	101	65	e
1100110	102	66	f
1100111	103	67	g
1101000	104	68	h
1101001	105	69	i
1101010	106	6A	j
1101011	107	6B	k
1101100	108	6C	l
1101101	109	6D	m
1101110	110	6E	n

Table A-2 ASCII Conversion Table (cont'd)

BINARY	DECIMAL	HEXADECIMAL	ASCII
1101111	111	6F	o
1110000	112	70	p
1110001	113	71	q
1110010	114	72	r
1110011	115	73	s
1110100	116	74	t
1110101	117	75	u
1110110	118	76	v
1110111	119	77	w
1111000	120	78	x
1111001	121	79	y
1111010	122	7A	z
1111011	123	7B	{
1111100	124	7C	|
1111101	125	7D	}
1111110	126	7E	~
1111111	127	7F	DEL

Appendix B

CP/M-86 File Types



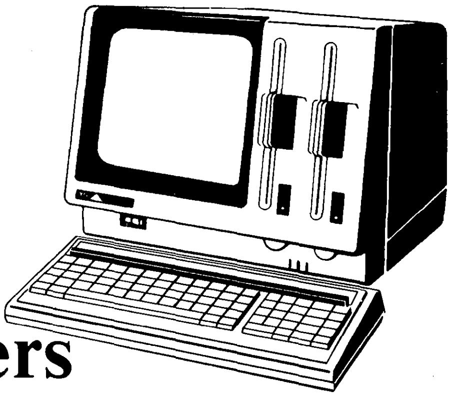
CP/M-86 identifies every file by a unique file specification, consisting of a drive specifier, a filename, and a filetype. The filetype is an optional three character ending separated from the filename by a period. The filetype generally indicates a special kind of file. The following table lists common filetypes and their meanings.

Table B-1 Filetypes

FILETYPE	INDICATION
A86	Assembly language source file; the CP/M-86 assembler, ASM-86, assembles or translates a file with the A86 filetype into machine language.
BAK	Backup file created by text editor; an editor renames the source file with this filetype to indicate that the original file has been processed. The original file stays on the disk as the backup file, so you refer to it by the original file name with the filetype as BAK.
BAS	CBASIC/86 program source file.
CHR	Auxiliary character file created using the APC Auxiliary Character Generator.
CMD	Command file that contains instructions in 8086 or 8088 machine language code.
H86	Assembled ASM-86 program file in hexadecimal format.
INT	CBASIC/86 program intermediate language file.

Table B-1 Filetypes (cont'd)

FILETYPE	INDICATION
KEY	Data file containing APC function key assignments.
LST	Printable file that can be displayed on the console or listed on the printer.
PRN	Printable file that can be displayed on the console or listed on the printer.
SAV	Filetype that identifies system backup file.
SUB	Filetype required for SUBMIT input file containing one or more CP/M-86 commands. The SUBMIT program executes the commands in a SUB file providing a batch mode for CP/M-86.
SYM	Symbol table file.
SYS	CP/M-86 system file.
$$$	Temporary file created with PIP or PIP1.



Appendix C
CP/M-86 Control Characters

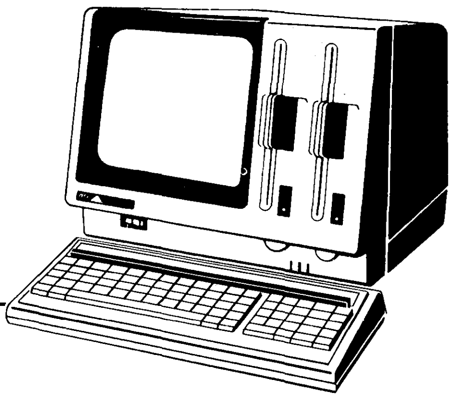
The following table describes the CP/M-86 control characters and any keys on the APC keyboard that perform the same function.

Table C-1 CP/M-86 Control Characters

KEYSTROKE	ACTION	EQUIVALENT APC KEY
CTRL-C	Aborts the program currently running.	SHIFT BREAK/STOP
CTRL-E	Moves the cursor to the beginning of the following line without erasing any previous input.	
CTRL-H	Moves the cursor left one character position and deletes the character.	BACK SPACE
CTRL-I	Moves the cursor to the next tab stop, where tab stops are automatically placed at every eighth column.	TAB
CTRL-J	Moves the cursor to the left of the current line and sends the command line and line feed character to CP/M-86.	
CTRL-M	Moves the cursor to the left of the current line and sends the command line to CP/M-86.	RETURN
CTRL-P	Echoes all console activity at the printer. Pressing again ends printer echo.	PRINT

Table C-1 CP/M-86 Control Characters (cont'd)

KEYSTROKE	ACTION	EQUIVALENT APC KEY
CTRL-R	Types a # at the current cursor location, moves the cursor to the next line, and retypes any partial command typed so far.	
CTRL-S	Temporarily stops console listing. Pressing again resumes the listing.	BREAK STOP
CTRL-U	Discards all characters in the command typed so far, types a # at the current cursor location, and moves the cursor to the next command line.	
CTRL-X	Discards all characters in the command line typed so far and moves the cursor back to the beginning of the current line.	DEL
CTRL-Z	Separates fields or strings.	



Appendix D

CP/M-86 System Command Messages

Appendix D presents the command and error messages generated by the transient utility programs that comprise the CP/M-86 operating system. The messages are organized alphabetically within the section for each of the following programs.

ASM-86
CP/M-86
COPYDISK
DDT-86
GRAPHICS
HDBACKUP
HDFORMAT
PIP
PIP1
STAT

Each message is followed by its meaning and, where applicable, suggested corrections.

ASM-86 COMMAND MESSAGES

Cannot close

An output file cannot be closed. This is a fatal error that terminates ASM-86 execution. Check to see that the correct disk is in the drive and that the disk is not write protected.

Directory full

There is not enough directory space for the output files. You should either erase some unnecessary files or use another disk with more directory space and execute ASM-86 again.

Disk full

There is not enough disk space for the output files (LST, H86 and SYM). Erase some unnecessary files or get another disk with more space and execute ASM-86 again.

Disk read error

A source or include file could not be read properly. This is usually the result of an unexpected end of file. Correct the problem in your source file.

Double defined label

An identifier used as a label is used elsewhere in the program as a label or variable name. Example:

```
LAB3:    MOV     BX,5
LAB3:    CALL    MOVE
```

Double defined variable

An identifier used as the name of a variable is used elsewhere in the program as the name of a variable or label. Example:

```
X    DB    5
X    DB    123H
```

Double defined symbol - treated as undefined

The identifier used as the name of an EQU directive is used as a name elsewhere in the program.

Error in codemacro building

Either a codemacro contains invalid statements, or a codemacro directive was encountered outside a codemacro.

File name syntax error

The filename in an INCLUDE directive is improperly formed. Example:

```
INCLUDE FILE.A86X
```

Garbage at end of line - ignored

Additional items were encountered on a line when ASM-86 was expecting an end of line. Examples:

```
NOLIST 4
MOV     AX,4     RET
```

Illegal expression element

An expression is improperly formed. Examples:

```
X    DB    12X
DW     (4*)
```

Illegal first item

The first item on a source line is not a valid identifier, directive or mnemonic. Example:

1234H

Illegal "IF" operand - "IF" ignored

Either the expression in an IF statement is not numeric, or it contains a forward reference.

Illegal pseudo instruction

Either a required identifier in front of a pseudo instruction is missing, or an identifier appears before a pseudo instruction that doesn't allow an identifier.

Illegal pseudo operand

The operand in a directive is invalid. Examples:

```
X    EQU    0AGH
     TITLE UNQUOTED STRING
```

Instruction not in code segment

An instruction appears in a segment other than a CSEG.

Label out of range

The label referred to in a call, jump, or loop instruction is out of range. The label can be defined in a segment other than the segment containing the instruction. In the case of short instructions (JMPS, conditional jumps and loops), the label is more than 128 bytes from the location of the following instruction.

Missing instruction

A prefix on a source line is not followed by an instruction. Example:

```
REPNZ
```

Missing pseudo instruction

The first item on a source line is a valid identifier and the second item is not a valid directive that can be preceded by an identifier. Example:

THIS IS A MISTAKE

Missing segment information in operand

The operand in a CALLF or JMPF instruction (or an expression in a DD directive) does not contain segment information. The required segment information can be supplied by including a numeric field in the segment directive as follows:

```
   CSEG     1000H
X:
   . . .
   JMPF  X
   DD    X
```

Missing type information in operand(s)

Neither instruction operand contains sufficient type information. Example:

```
MOV     [BX] ,10
```

Nested "IF" illegal - "IF" ignored

The maximum nesting level for IF statements has been exceeded.

Nested INCLUDE not allowed

An INCLUDE directive was encountered within a file already being included.

No file

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

No matching "IF" for "ENDIF"

An ENDIF statement was encountered without a matching IF statement.

Operand(s) mismatch instruction

Either an instruction has the wrong number of operands or the types of the operands do not match. Examples:

```
     MOV    CX,1,2
  X  DB     0
     MOV    AX,X
```

Parameter error

A parameter in the command tail of the ASM-86 command was specified incorrectly. Example:

```
ASM86 TEST $S;
```

Symbol illegally forward referenced - neglected

The indicated symbol was illegally forward referenced in an ORG, RS, EQU or IF statement.

Symbol table overflow

There is not enough memory for the symbol table. Either reduce the length and/or number of symbols, or reassemble on a system with more memory available.

Undefined element of expression

An identifier used as an operand is not defined or has been illegally forward referenced. Examples:

```
       JMP    X
  A    EQU    B
  B    EQU    5
       MOV    AL,B
```

Undefined instruction

The item following a label on a source line is not a valid instruction. Example:

```
DONE:    BAD    INSTR
```

COPYDISK COMMAND MESSAGE

Is this what you want to do (Y/N)?

If the displayed COPYDISK function is what you want performed, press Y.

CP/M-86 COMMAND MESSAGES

BDOS err on *d*:

CP/M-86 replaces *d*: with the drive specifier of the drive where the error occurred. This message appears when CP/M-86 finds no disk in the drive, when the disk is improperly formatted, when the drive latch is open, or when power to the drive is off. Check for one of these situations and retry.

BDOS err on *d*: bad sector

This could indicate a hardware problem or a worn or improperly formatted disk. Press CTRL-C to terminate the program and return to CP/M-86, or press RETURN to ignore the error.

BDOS err on *d*: select

CP/M-86 has received a request specifying a nonexistent drive, or a disk in a drive is improperly formatted. CP/M-86 terminates the current program as soon as you press any key.

BDOS err or *d*: RO

Drive has been assigned Read-Only status with a STAT command, or the diskette in the drive has been changed without being initialized with a CTRL-C. CP/M-86 terminates the current the current program as soon as you press any key.

Command name?

If CP/M-86 cannot find the command you specified, it returns the command name you entered followed by a question mark. Check that you have typed the command name correctly, and that the command you requested exists as a CMD file on the default or specified disk.

File exists

You have asked CP/M-86 to create a new file using a file specification that is already assigned to another file. Either delete the existing file or use another file specification.

File not found

CP/M-86 could not find the specified file. Check that you have entered the correct drive specification and that you have the correct disk in the drive.

No file

CP/M-86 cannot find the specified file, or no files exist.

DDT-86 COMMAND MESSAGES

Ambiguous operand

An attempt was made to assemble a command with an ambiguous operand. Precede the operand with the prefix "BYTE" or "WORD".

Cannot close

The disk file written by a W command cannot be closed. This is a fatal error that terminates DDT-86 execution. Check to see that the correct disk is in the drive and that the disk is not write protected.

Disk read error

The disk file specified in an R command could not be read properly. This is usually the result of an unexpected end of file. Correct the problem in your file.

Disk write error

A disk write operation could not be successfully performed during a W command, probably due to a full disk. Erase some unnecessary files or use another disk with more space and execute ASM-86 again.

Insufficient memory

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

Memory request denied

A request for memory during an R command could not be fulfilled. Up to eight blocks of memory can be allocated at a given time.

No file

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

No space

There is no space in the directory for the file being written by a W command.

Verify error at s:o

The value placed in memory by a Fill, Set, Move, or Assemble command cannot be read back correctly, indicating bad user memory, an attempt to write to ROM, or nonexistent memory at the indicated location.

GRAPHICS COMMAND MESSAGES

d:ASSIGN   .SYS close error

The system could not find the file to close it. Try again using correct disk.

d:ASSIGN   .SYS not found

The ASSIGN.SYS file is not on the drive specified in the message. Enter DIR to see which drive ASSIGN.SYS is on. Try the GRAPHICS command again.

d:ASSIGN   .SYS syntax error

The first line in the ASSIGN.SYS file does not follow the syntax rule. Edit the file, making necessary corrections. Try the GRAPHICS command again.

d:ffffffff.xxx close error

An error occurred when the system was closing the specified file. Try again with the correct disk.

d:ffffffff.xxx contains absolute segment

The specified file is invalid. Use another driver. You may have to modify the Assignment Table in ASSIGN.SYS.

d:ffffffff.xxx empty

The specified file was found, but it had no data in it. Delete the empty file. Copy the file from the distribution diskette again.

d:ffffffff.xxx load error

An error occurred while the system was reading the driver file. This probably happened if you swapped diskettes. Try again with the correct disk.

d:ffffffff.xxx not found

One of the device driver files specified in the Assignment Table in ASSIGN.SYS was not found on the drive specified in the message. Enter DIR to find out which drive the file is on. The ASSIGN.SYS file and the device driver files must be on the same drive.

Not enough memory for GSX-88

There is not enough memory for GSX-86 and the default device driver. You need more memory.

HDBACKUP COMMAND MESSAGES

FATAL ERROR. ZERO CYLINDER IS BAD
ABNORMAL END

The diskette or the hard disk drive contains a defective cylinder 0. Initiate HDBACKUP again and retry the operation.

FLOPPY DISK *d*: IS NOT READY

The diskette drive name was specified before the diskette was inserted or properly placed in the drive.

FLOPPY DISK H/W ERROR
CYLINDER [*ccc*]
TRACK [*tt*]
PLEASE CHANGE FLOPPY DISK

A defective area of the diskette was encountered during a SAVE operation. Replace the diskette.

HARD DISK IS NOT AVAILABLE

The hard disk is not properly attached, power to the disk is off, or improper version of CP/M-86 is used. Check that the power is on and that all cables are securely attached.

PLEASE CHANGE FLOPPY DISK
HARD DISK H/W ERROR
CYLINDER [*ccc*]
TRACK [*tt*]

The error map of the hard disk is unreadable or there is an error not listed in the hard disk error map. Press R to retry. If the message reappears, press A. Restart HDBACKUP and try the operation again. Pressing I may result in copying defective data to the hard disk.

SIZE CHECK ERROR

The amount of data to be recovered exceeds the size of the copying hard disk drive. Press R. If the message reappears, check that the hard disk drive specified is the one from which the data was originally saved. If not, press A, then begin again with the correct drive specifier.

THIS FLOPPY HAS ALREADY BEEN LOADED

During a load operation, indicates that the data on the diskette in the drive already has been restored to the hard disk. Remove the diskette and insert one that has not been restored to the hard disk.

HDFORMAT COMMAND MESSAGES

DISK HW ERROR IN [*ccc*] CYLINDER ON (*drive or unit number*)
[R:RETRY A:ABORT] :

An error occurred during cylinder [*ccc*] access. Press R to retry. If the error message reappears, press A and restart HDFORMAT from the beginning.

DISK HW ERROR IN ERROR MAP WRITING ON (*drive or unit number*)
R:RETRY A:ABORT :

An error occurred when the error map was being written. Press R to write to the error map again. If the message reappears, press A and restart HDFORMAT from the beginning.

DISK HW ERROR ? ON (*drive or unit number*)
R:RETRY A:ABORT :

An undefined error occurred. Make sure that power to the hard disk unit is on and all cables are connected. Press R.

DRIVE *d*: NOT READY
THIS DRIVE EXISTS IN HARD DISK UNIT *u*:
PLEASE SET HARD DISK UNIT TO READY CONDITION
[R:RETRY A:ABORT] :

The specified logical drive unit, *d*:, is not ready or not attached. Check the cables and the power switch to be sure the drive unit, *u*:, is ready. Press R to retry or A to abort.

ERROR MAP READ ERROR ON DRIVE *d*:

Format the entire disk unit in which the specified drive (*d*:) is located.

FATAL ERROR
ERROR MAP TRACK IS BAD ON HARD DISK UNIT *u*:
ABNORMAL END
A:AGAIN E:END :

The track on which the error map is located (cylinder 0, track 2) is bad. Try reformatting the disk.

HARD DISK UNIT *u*: NOT READY
PLEASE SET HARD DISK UNIT TO READY CONDITION
[R:RETRY A:ABORT] :

The disk unit you specified is not ready. Check the cables and power switch. Press R to retry or A to abort.

NEED FULL FORMATTING HARD DISK UNIT *u*:

Format the entire disk unit in which the disk unit specified (*u*:) is located.

NOW FORMATTING DISK ON DRIVE *d*:

Format the entire disk unit in which the drive specified (*d*:) is located.

NO HARD DISK SYSTEM
END OF FORMAT PROGRAM

The drive or unit number is not defined in the CP/M-86 BIOS. Check that you have the appropriate version of CP/M-86.

NOT CP/M FORMAT ON DRIVE *d*:

Format the entire disk unit in which the drive specified (*d*:) is located.

OUT OF CYLINDER
PLEASE CHECK HARD DISK DEFINITION TABLE

No more tracks are available for bad track reallocation. Try reformatting the drive.

PIP COMMAND MESSAGES

DESTINATION IS R/O, DELETE (Y/N)?

The destination file specified in a PIP command already exists and has the Read-Only attribute. If you press Y, the destination file is deleted before the file copy is done.

ERROR: BAD PARAMETER

An illegal parameter was entered in a PIP command. Retype the entry correctly.

ERROR: CLOSE FILE {*filespec*}

An output file cannot be closed. Check to see that the correct disk is in the drive and that the disk is not write protected.

ERROR: DISK READ - {*filespec*}

The input disk file specified in a PIP command cannot be read properly. This is usually the result of an unexpected end of file. Correct the problem in your file.

ERROR: DISK WRITE - {*filespec*}

A disk write operation cannot be successfully performed during a PIP command, probably due to a full disk. Erase some unnecessary files or use another disk with more space and execute PIP again.

ERROR: FILE NOT FOUND -{*filespec*}

An input file that you have specified does not exist.

ERROR: HEX RECORD CHECKSUM -{*filespec*}

A hex record checksum was encountered during the transfer of a hex file. The hex file with the chekcsum error should be corrected, probably by recreating the hex file.

ERROR: INVALID DESTINATION

The destination specified in your PIP command is illegal. You have probably specified an input device as a destination.

ERROR: INVALID FORMAT

The format of your PIP command is illegal. See the description of the PIP command.

ERROR: INVALID HEX DIGIT -{*filespec*}

An invalid hex digit has been encountered while reading a hex file. The hex file with the invalid hex digit should be corrected, probably by recreating the hex file.

ERROR: INVALID SEPARATOR

You used an invalid character for a separator between two input filenames.

ERROR: INVALID SOURCE

The source specified in your PIP command is illegal. You have probably specified an output device as a source.

ERROR: INVALID USER NUMBER

You have specified a User Number greater than 15. User Numbers are in the range 0 to 15.

ERROR: NO DIRECTORY SPACE -{*filespec*}

There is not enough directory space for the output file. Erase some unnecessary files or use another disk with more directory space and execute PIP again.

ERROR: QUIT NOT FOUND

The string argument to a Q parameter was not found in your input file.

ERROR: START NOT FOUND

The string argument to an S parameter cannot be found in the source file.

ERROR: UNEXPECTED END OF HEX FILE - {*filespec*}

An end of file was encountered prior to a termination hex record. The hex file without a termination record should be corrected, probably by recreating the hex file.

ERROR: USER ABORTED

The user has aborted a PIP operation by pressing a key.

ERROR: VERIFY - {*filespec*}

When copying with the V option, PIP found a difference when rereading the data just written and comparing it to the data in its memory buffer. Usually this indicates a failure of either the destination disk or drive.

PIP1 COMMAND MESSAGES

A (ABORT) or R (RETRY)

If you have kept track of source and destination diskettes, this message will occur only when there is a physical problem with the information on the diskette.

Do not press A unless there is no way to recover by pressing R. If you press A, the system returns to the A> prompt and the SSSSSSSS.SSS file is not erased.

If the file transfer was not completed, the DDDDDDDD.DDD file is not erased on the destination diskette. The last file transferred will be complete and correct. However, any *.$$$ files must be deleted when you press A since they will be missing extents.

DESTINATION FILE NOT FOUND

Usually appears when the incorrect destination diskette was labelled with the correct identification. Check that the diskette is not worn. Insert the proper destination diskette and press R. If the destination file is not found, there is a physical problem with the disk medium. Press A to abort.

INCORRECT DESTINATION DISKETTE

A diskette without the DDDDDDDD.DDD file was inserted. Insert the correct destination diskette and continue.

INCORRECT SOURCE DISKETTE

A diskette without the SSSSSSSS.SSS file was inserted. Insert the correct source diskette and continue.

NOT ENOUGH SPACE

Either all possible directory entries are taken or all extents on the diskette have been allocated to existing files. The system returns to the A> prompt. Only the last extent of the last file transferred is lost. ERASE the SSSSSSSS.SSS, DDDDDDDD.DDD and *.$$$ files from the source and destination diskettes before attempting another transfer.

SOURCE FILE NOT FOUND

Incorrect source diskette was inserted. Check that the diskette is not worn. Insert the correct diskette and press R to retry. If the source file is missing and there is no physical problem with the medium, press A to abort.

SYNTAX ERROR IN COMMAND LINE

The command line was typed incorrectly. Check for illegal characters. Then check that the destination file and source file specifications match.

STAT COMMAND MESSAGES

Bad Directory on d:
Space Allocation Conflict:
User *n d:filename.typ*

STAT has detected a space allocation conflict in which one data block is assigned to more than one file. One or more filenames can be listed. Each of the files listed contains a data block already allocated to another file on the disk. Correct the problem by erasing the files listed. After erasing the conflicting file or files, press CTRL-C to regenerate the allocation vector. If you do not, the error might repeat itself.

Invalid Assignment

An invalid device was specified in a STAT device assignment. Use the STAT VAL: command to display the valid assignments for each of the four logical STAT devices: CON:, RDR:, PUN: and LST:.

Too Many Files

A STAT wildcard command matched more files in the directory than STAT can sort. STAT can sort a maximum of 512 files.

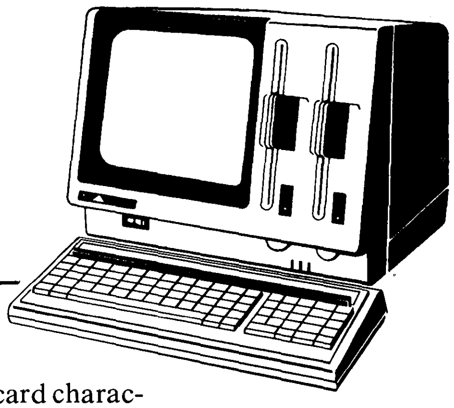
Use: [*size*][*ro*][*rw*][*sys*] or [*dir*]

This message results from an invalid set file attributes command. These are the only options valid in a STAT filespec [*option*] command.

Use: STAT *d*:=RO

An invalid STAT drive command was given. The only valid drive assignment in STAT is STAT *d*:=RO.

Glossary



ambiguous filename: Filename that contains either of the CP/M-86 wildcard characters, ? or *, in the primary filename, the filetype or both. When you replace characters in a filename with these wildcard characters, you create an ambiguous filename and can easily reference more than one CP/M-86 file in a single command line. See Chapter 2 of this manual.

applications program: Program that needs an operating system to provide an environment in which to execute. Typical application programs are business accounting packages, word processing (editing) programs, and mailing list programs.

argument: Symbol, usually a letter, indicating a place into which you can substitute a number, letter, or name to give an appropriate meaning to the formula in question.

ASCII: The American Standard Code for Information Interchange is a standard code for representation of numbers, letters, and symbols. An ASCII text file is a file that can be intelligibly displayed on the video screen or printed on paper. See Appendix A.

attribute: File characteristic that can be set to on or off.

backup: Copy of a diskette, disk, or file made for safekeeping.

bit: "Switch" in memory that can be set to on (1) or off (0). Bits are grouped into bytes.

block: Area of disk reserved for a specific use.

bootstrap: Process of loading an operating system into memory. Bootstrap procedures vary from system to system. The boot for an operating system must be customized for the memory size and hardware environment that the operating system manages. Typically, the boot is loaded automatically and executed at power up or when the computer is reset. Sometimes called a cold start.

buffer: Area of memory that temporarily stores data during the transfer of information.

built-in commands: Commands that permanently reside in memory. They respond quickly because they are not accessed from a disk. See Chapter 1.

byte: Unit of memory or disk storage containing eight bits.

command: Elements of a CP/M-86 command line. In general, a CP/M-86 command has three parts: the command keyword, the command tail, and a carriage return.

command file; Series of coded machine-executable instructions stored on a disk medium as a program file, invoked in CP/M-86 by typing the command keyword at the system prompt on the console. CP/M-86 command files generally have the filetype CMD.

command keyword: Name that identifies a CP/M-86 command, usually the primary filename of a file with CMD filetype, or a built-in command. The command keyword precedes the command tail and carriage return in the command line.

command syntax: Statement that defines the correct way to enter a command. The correct structure generally includes the command keyword, the command tail, and a carriage return. A syntax line usually contains symbols that you replace with actual values when you enter the command. In this manual, symbols are printed in lowercase italics.

command tail: Part of a command that follows the command keyword in the command line. The command tail can include a drive specification, a filename and/or filetype, and options or parameters. Some commands do not require a command tail.

concatenate: To link together. The term describes one of PIP's operations that copies two or more separate files into one new file in the specified sequence.

console: Primary input/output device. The console consists of a listing device such as a screen and a keyboard through which the user communicates with the operating system or applications program.

control character: Nonprinting character combination that sends a simple command to CP/M-86. Some control characters perform line editing functions. To enter a control character, simultaneously press the CTRL (control) key on the keyboard and the character key specified. See Appendix C.

cursor: Symbol that appears on the console screen to indicate the position where the next keystroke at the console will have an effect.

cylinder: The corresponding tracks on each side of a dual-sided diskette or disk. On a single-sided diskette, each track is also called a cylinder. A diskette has 77 cylinders; a hard disk has 181 cylinders. See **track**.

data file: Nonexecutable collection of similar information that generally requires a command file to manipulate it.

default: Value assumed or supplied automatically by the system. Used in CP/M-86 to refer to the currently selected diskette or disk drive and user number. Any command that does not specify a drive or a user number references the default drive and user number. When CP/M-86 is first invoked, the default drive is Drive A and the default user number is 0. The default drive can be changed by entering a new drive specifier (A-H) at the "A>" prompt. The user number can be changed with the USER command.

delimiter: Special character or characters used to separate different items in a command line. For example, in CP/M-86, a colon separates the drive specification from the filename. A period separates the filename from the filetype. Brackets separate any options from their command or file specification. Commas separate one item in an option list from another. All of the above special characters are delimiters.

directory: Portion of a diskette or disk that contains entries for each file on the disk medium. In response to the DIR command, CP/M-86 displays the filenames stored in the directory.

DIR attribute: File attribute. A file with the DIR attribute can be displayed by a DIR command. The file can be accessed from the default user number and drive only.

disk, diskette: Magnetic medium used to store information. Programs and data are recorded on the disk in the same way that music is recorded on a cassette tape. A hard disk is a high-capacity storage device that is permanently mounted in the drive. A diskette is removable and flexible. It has less storage capacity and slower access speed than a hard disk.

disk drive: Peripheral device that reads and writes on hard disks or diskettes. CP/M-86 assigns a letter to each drive under its control. For example, CP/M-86 with the APC refers to the diskette drives in a four-drive system as A, B, C, and D. The hard disk drives are labeled E, F, G, and H.

editor: Utility program that creates and modifies text files. An editor can be used for creation of documents or creation of code for computer programs. The CP/M-86 editor is invoked by typing the command ED at the system prompt on the console. (See Chapter 4 of this manual).

executable: Ready to be run by the computer. Executable code is a series of machine language instructions that can be carried out by the computer. For example, the computer cannot execute names and addresses, but it can execute a program that reads names and addresses from a data file and prints them on mailing labels.

execute a program: To start running a program. When a program is running, the computer is executing a sequence of instructions.

extent: A contiguous area of a disk or other physical storage medium. Usually refers to some group of sectors or blocks.

FCB: File Control Block.

file: Collection of characters, instructions or data stored on a disk. The user can create files on a disk.

File Control Block: Structure used for accessing files on diskette or disk. Contains the drive, filename, filetype and other information describing a file to be accessed or created.

filename: Name assigned to a file. A filename can include a primary filename of 1-8 characters and a filetype of 0-3 characters. A period separates the primary filename from the filetype.

file specification: Unique file identifier. A complete CP/M-86 file specification, also called a filespec, includes a disk drive specification followed by a colon (*d*:), a primary filename of 1 to 8 characters, a period and a filetype of 0 to 3 characters. For example, b:example.tex is a complete CP/M-86 file specification.

filetype: Extension to a filename. A filetype can be from 0 to 3 characters and must be separated from the primary filename by a period. A filetype can tell something about the file. Certain programs require that files to be processed have certain filetypes (see Appendix B).

hardware: Physical components of a computer.

hex file: ASCII-printable representation of a command (machine language) file.

hexadecimal notation: Notation for the base 16 number system using the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F to represent the sixteen digits. Machine code is often converted to hexadecimal notation because it can be easily represented by ASCII characters and therefore displayed on the console screen or printed on paper (see Appendix A).

input: Data going into the computer, usually from an operator typing at the keyboard, a program reading from the disk, or a data communications interface.

interface: Object that allows two independent systems to communicate with each other, as an interface between hardware and software in a microcomputer.

I/O: Abbreviation for input/output.

keyword: See **command keyword.**

kilobyte: 1024 bytes denoted as 1K. 32 kilobytes equal 32K. 1024 kilobytes equal one megabyte, or over one million bytes.

list device: Device such as a printer at which data can be listed or printed.

logged in: Made known to the operating system, in reference to drives. A drive is logged in when it is selected by the user or an executing process. It remains selected or logged in until you change diskettes in a diskette drive or enter CTRL-C at the command level.

logical: Representation of something that may or may not be the same in its actual physical form. The term usually refers to how the computer undertstands and controls data flow regardless of the actual hardware. For example, CP/M-86 defines several logical device names that you can arbitrarily reassign to a variety of physical input/output devices. The computer then recognizes the physical device you are using by the logical name you assign.

megabyte: Over one million bytes; 1024 kilobytes (see **byte, kilobyte**).

microprocessor: Silicon chip that is the Central Processing Unit (CPU) of the microcomputer.

operating system: Collection of programs that supervises the running of other programs and the management of computer resources. An operating system provides an orderly input/output environment between the computer and its peripheral devices. CP/M-86 is an operating system for the APC.

option: One of many parameters that can be part of a command tail. Use options to specifiy additional conditions for the execution of a command.

output: Data that the processor sends to a peripheral device such as the console, printer, or disk.

parameter: Value in the command tail that provides additional information for the command. Technically, a parameter is a required element of a command.

peripheral devices: Devices external to the CPU. For example, consoles, printers, and disk drives are common peripheral devices that are not part of the processor, but are used in conjunction with it.

physical: A term that usually refers to the actual hardware of a computer. The physical environment varies from computer to computer.

primary filename: First 8 characters of a filename. The primary filename is a unique name that helps the user identify the file contents. A primary filename can include any letter or number and some special characters. The primary filename follows the optional drive specification and precedes the optional filetype.

program: Series of specially coded instructions that performs specific tasks when executed by a computer.

prompt: Any characters displayed on the screen to help the user decide what the next appropriate action is. A system prompt is a special prompt displayed by the operating system. The system prompt indicates to the user that the operating system is ready to accept input. The CP/M-86 system prompt is an alphabetic character followed by an angle bracket (>). The alphabetic character indicates the default drive. Some applications programs have their own special "system" prompts.

Read-Only: Attribute that can be assigned to a diskette or disk file or a drive. When assigned to a file, the Read-Only attribute allows you to read from that file but not write any changes to it. When assigned to a drive, the Read-Only attribute allows you to read any file on the disk, but prevents you from adding a new file, erasing or changing a file, renaming a file, or writing on the disk medium. The STAT command can set a file or a drive to Read-Only. Every file and drive is either Read-Only or Read-Write. The default setting for drives and files is Read-Write, but an error in resetting a diskette or changing media automatically sets the drive to Read-Only until the error is corrected.

Read-Write: Attribute that can be assigned to a diskette or disk file or to a drive. The Read-Write attribute allows you to read from and write to a specific Read-Write file or to any file on a diskette or disk that is in a drive set to Read-Write. A file or drive can be set to either Read-Only or Read-Write.

record: Collection of data. A file consists of one or more records stored on diskette or disk.

RO: Abbreviation for Read-Only.

RW: Abbreviation for Read-Write.

sector: Portion of a disk track. There are 26 sectors on each track of both floppy and hard disks.

software: Specially coded programs that transmit machine readable instructions to the computer. Software is differentiated from hardware, which is the actual physical components of a computer.

source file: ASCII text file that is an input file for a processing program, such as an editor, text formatter, or assembler.

syntax: Format for entering a given command.

system attribute: A file attribute. You can give a file the system attribute by using the SYS option in the STAT command. A file with the SYS attribute is not displayed in response to a DIR command; you must use the DIRS command (see Chapter 2). If you give a file with user number 0 the SYS attribute, you can read and execute that file from any user number on the same drive. Use this feature to make your commonly used programs available under any user number.

system prompt: Symbol displayed by the operating system indicating that the system is ready to receive input. See **prompt.**

terminal: See **console.**

timer: An electronic timer which monitors and controls input/output servicing by the operating system. When the TIMEROFF command is in effect, the keyboard repeat function is disabled on the APC running under CP/M-86.

track: Concentric rings dividing a disk. There are 2 tracks per cylinder an an eight-inch diskette and 8 tracks per cylinder on a hard disk.

user number: Number assigned to files in the disk directory so that different users need only deal with their own files in their own logical directories even though they are all working from the same disk. In CP/M-86, files can be divided into 16 user groups.

utility: Program that enables the user to perform certain common operations, such as copying files, erasing files, and editing files. Utilities are created for the convenience of programmers and users.

wildcard characters: Special characters that match certain specified items. In CP/M-86 there are two wildcard characters, ? and *. The ? can be substituted for any single character in a filename, and the * can be substituted for the primary filiename or the filetype or both. By placing wildcard characters in filenames, the user creates an ambiguous filename and can quickly reference one or more files.

Index