Section 1
CP/M Features and Facilities

Table of Contents

Figures

1.1 Introduction

CP/M is a monitor control program for microcomputer system development that uses floppy disks or

Winchester hard disks for backup storage. Using a computer system based on the Intel 8080 microcomputer, CP/M provides an environment for program construction, storage, and editing, along with assembly and program checkout facilities. CP/M can be easily altered to execute with any computer configuration that uses a Zilog Z80 or an Intel 8080 Central Processing Unit (CPU) and has at least 20K bytes of main memory with up to 16 disk drives. A detailed discussion of the modifications required for any particular hardware environment is given in Section 6. Although the standard Digital Research version operates on a single-density Intel MDS 800, several different hardware manufacturers support their own input-output (I/O) drivers for CP/M.

The CP/M monitor provides rapid access to programs through a comprehensive file management package. The file subsystem supports a named file structure, allowing dynamic allocation of file space as well as sequential and random file access. Using this file system, a large number of programs can be stored in both source and machine executable form.

CP/M 2 is a high-performance, single console operating system that uses table-driven techniques to allow field reconfiguration to match a wide variety of disk capacities. All fundamental file restrictions are removed, maintaining upward compatibility from previous versions of release 1.

Features of CP/M 2 include field specification of one to sixteen logical drives, each containing up to eight megabytes. Any particular file can reach the full drive size with the capability of expanding to thirty-two megabytes in future releases. The directory size can be field-configured to contain any reasonable number of entries, and each file is optionally tagged with Read-Only and system attributes. Users of CP/M 2 are physically separated by user numbers, with facilities for file copy operations from one user area to another. Powerful relative-record random access functions are present in CP/M 2 that provide direct access to any of the 65536 records of an eight-megabyte file.

CP/M also supports ED, a powerful context editor, ASM, an Intel-compatible assembler, and DDT, debugger subsystems. Optional software includes a powerful Intel-compatible macro assembler, symbolic debugger, along with various high-level languages. When coupled with CP/M's Console Command Processor (CCP), the resulting facilities equal or exceed similar large computer facilities.

CP/M is logically divided into several distinct parts:

- BIOS (Basic I/O System), hardware-dependent
- BDOS (Basic Disk Operating System)
- CCP (Console Command Processor)
- TPA (Transient Program Area)

The BIOS provides the primitive operations necessary to access the disk drives and to interface standard peripherals: teletype, CRT, paper tape reader/punch, and user-defined peripherals. You can tailor peripherals for any particular hardware environment by patching this portion of CP/M. The BDOS provides disk management by controlling one or more disk drives containing independent file directories. The BDOS implements disk allocation strategies that provide fully dynamic file construction while minimizing head movement across the disk during access. The BDOS has entry points that include the following primitive operations, which the program accesses:

- SEARCH looks for a particular disk file by name.
- OPEN opens a file for further operations.
- CLOSE closes a file after processing.
- RENAME changes the name of a particular file.
- READ reads a record from a particular file.
- WRITE writes a record to a particular file.
- SELECT selects a particular disk drive for further operations.

The CCP provides a symbolic interface between your console and the remainder of the CP/M system. The CCP reads the console device and processes commands, which include listing the file directory, printing the contents of files, and controlling the operation of transient programs, such as assemblers, editors, and debuggers. The standard commands that are available in the CCP are listed in Section 1.2.1.

The last segment of CP/M is the area called the Transient Program Area (TPA). The TPA holds programs that are loaded from the disk under command of the CCP. During program editing, for example, the TPA holds the CP/M text editor machine code and data areas. Similarly, programs created under CP/M can be checked out by loading and executing these programs in the TPA.

Any or all of the CP/M component subsystems can be overlaid by an executing program. That is, once a user's program is loaded into the TPA, the CCP, BDOS, and BIOS areas can be used as the program's data area. A bootstrap loader is programmatically accessible whenever the BIOS portion is not overlaid; thus, the user program need only branch to the bootstrap loader at the end of execution and the complete CP/M monitor is reloaded from disk.

The CP/M operating system is partitioned into distinct modules, including the BIOS portion that defines the hardware environment in which CP/M is executing. Thus, the standard system is easily modified to any nonstandard environment by changing the peripheral drivers to handle the custom system.

1.2 Functional Description

You interact with CP/M primarily through the CCP, which reads and interprets commands entered through the console. In general, the CCP addresses one of several disks that are on-line. The standard system addresses up to sixteen different disk drives. These disk drives are labeled A through P. A disk is logged-in if the CCP is currently addressing the disk. To clearly indicate which disk is the currently logged disk, the CCP always prompts the operator with the disk name followed by the symbol >, indicating that the CCP is ready for another command. Upon initial start-up, the CP/M system is loaded from disk A, and the CCP displays the following message:

```
CP/M VER x.x
```

where x.x is the CP/M version number. All CP/M systems are initially set to operate in a 20K memory space, but can be easily reconfigured to fit any memory size on the host system (see Section 1.6.9). Following system sign-on, CP/M automatically logs in disk A, prompts you with the symbol A>, indicating that CP/M is currently addressing disk A, and waits for a command. The commands are implemented at two levels: built-in commands and transient commands.

1.2.1 General Command Structure

Built-in commands are a part of the CCP program, while transient commands are loaded into the TPA from disk and executed. The following are built-in commands:

- ERA erases specified files.
- DIR lists filenames in the directory.
- REN renames the specified file.
- SAVE saves memory contents in a file.
- TYPE types the contents of a file on the logged disk.

Most of the commands reference a particular file or group of files. The form of a file reference is

specified in Section 1.2.2.

1.2.2 File References

A file reference identifies a particular file or group of files on a particular disk attached to CP/M. These file references are either unambiguous (ufn) or ambiguous (afn). An unambiguous file reference uniquely identifies a single file, while an ambiguous file reference is satisfied by a number of different files.

File references consist of two parts: the primary filename and the filetype. Although the filetype is optional, it usually is generic. For example, the filetype ASM is used to denote that the file is an assembly language source file, while the primary filename distinguishes each particular source file. The two names are separated by a period, as shown in the following example:

```
filename.typ
```

In this example, filename is the primary filename of eight characters or less, and typ is the filetype of no more than three characters. As mentioned above, the name

```
filename
```

is also allowed and is equivalent to a filetype consisting of three blanks. The characters used in specifying an unambiguous file reference cannot contain any of the following special characters:

```
< > . , ; : = ? * [ ] % | ( ) / \
```

while all alphanumerics and remaining special characters are allowed.

An ambiguous file reference is used for directory search and pattern matching. The form of an ambiguous file reference is similar to an unambiguous reference, except the symbol ? can be interspersed throughout the primary and secondary names. In various commands throughout CP/M, the ? symbol matches any character of a filename in the ? position. Thus, the ambiguous reference

```
X?Z.C?M
```

matches the following unambiguous filenames

```
XYZ.COM
```

and

```
X3Z.CAM
```

The wildcard character can also be used in an ambiguous file reference. The * character replaces all or part of a filename or filetype. Note that

```
*.*
```

equals the ambiguous file reference

```
????????.???
```

while

```
filename.*
```

and

```
*.typ
```

are abbreviations for

```
filename.???
```

and

```
????????.typ
```

respectively. As an example,

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

is interpreted by the CCP as a command to list the names of all disk files in the directory. The following example searches only for a file by the name X.Y:

```
A>DIR X.Y
```

Similarly, the command

```
A>DIR X?Y.C?M
```

causes a search for all unambiguous filenames on the disk that satisfy this ambiguous reference.

The following file references are valid unambiguous file references:

```
X
X.Y
XYZ
XYZ.COM
GAMMA
GAMMA.1
```

As an added convenience, the programmer can generally specify the disk drive name along with the filename. In this case, the drive name is given as a letter A through P followed by a colon (:). The specified drive is then logged-in before the file operation occurs. Thus, the following are valid file references with disk name prefixes:

```
A:X.Y
P:XYZ.COM
B:XYZ
B:X.A?M
C:GAMMA
C:*.ASM
```

All alphabetic lower-case letters in file and drive names are translated to upper-case when they are processed by the CCP.

1.3 Switching Disks

The operator can switch the currently logged disk by typing the disk drive name, A through P, followed by a colon when the CCP is waiting for console input. The following sequence of prompts and commands can occur after the CP/M system is loaded from disk A:

CP/M VER 2.2 A>DIR A:SAMPLE ASM SAMPLE PRN	List all files on disk A.
A>B:	Switch to disk B.
B>DIR *.ASM B:DUMP ASM FILES ASM	List all ASM files on B.
B>A:	Switch back to A.

1.4 Built-in Commands

The file and device reference forms described can now be used to fully specify the structure of the built-in commands. Assume the following abbreviations in the description below:

ufn unambiguous file reference

afn ambiguous file reference

Recall that the CCP always translates lower-case characters to upper-case characters internally. Thus, lower-case alphabetics are treated as if they are upper-case in command names and file references.

1.4.1 ERA Command

Syntax:
ERA afn

The ERA (erase) command removes files from the currently logged- in disk, for example, the disk name currently prompted by CP/M preceding the >. The files that are erased are those that satisfy the ambiguous file reference afn. The following examples illustrate the use of ERA:

ERA X.Y	The file named X.Y on the currently logged disk is removed from the disk directory and the space is returned.
ERA X.*	All files with primary name X are removed from the current disk.
ERA *.ASM	All files with secondary name ASM are removed from the current disk.
ERA X?Y.C?M	All files on the current disk that satisfy the ambiguous reference X?Y.C?M are deleted.
ERA *.*	Erase all files on the current disk. In this case, the CCP prompts the console with the message `ALL FILES (Y/N)?` which requires a Y response before files are actually removed.
ERA	All files on drive B that satisfy the ambiguous reference ????????.PRN are deleted,

B:*.PRN	independently of the currently logged disk.

1.4.2 DIR Command

Syntax:
DIR afn

The DIR (directory) command causes the names of all files that satisfy the ambiguous filename afn to be listed at the console device. As a special case, the command

```
DIR
```

lists the files on the currently logged disk (the command DIR is equivalent to the command DIR *.*). The following are valid DIR commands:

```
DIR X.Y
DIR X?Y.C?M
DIR ??.Y
```

Similar to other CCP commands, the afn can be preceded by a drive name. The following DIR commands cause the selected drive to be addressed before the directory search takes place:

```
DIR B:
DIR B:X.Y
DIR B:*.A?M
```

If no files on the selected disk satisfy the directory request, the message NO FILE appears at the console.

1.4.3 REN Command

Syntax:
REN ufn1=ufn2

The REN (rename) command allows you to change the names of files on disk. The file satisfying ufn2 is changed to ufn1. The currently logged disk is assumed to contain the file to rename (ufn2). You can also type a left-directed arrow instead of the equal sign if the console supports this graphic character. The following are examples of the REN command:

REN X.Y=Q.R	The file Q.R is changed to X.Y.
REN XYZ.COM=XYZ.XXX	The file XYZ.COM is changed to XYZ.XXX.

The operator precedes either ufn1 or ufn2 (or both) by an optional drive address. If ufn1 is preceded by a drive name, then ufn2 is assumed to exist on the same drive. Similarly, if ufn2 is preceded by a drive name, then ufn1 is assumed to exist on the drive as well. The same drive must be specified in both cases if both ufn1 and ufn2 are preceded by drive names. The following REN commands illustrate this format:

REN A:X.ASM=Y.ASM	The file Y.ASM is changed to X.ASM on drive A.
REN B:ZAP.BAS=ZOT.BAS	The file ZOT.BAS is changed to ZAP.BAS on drive B.
REN B:A.ASM=B:A.BAK	The file A.BAK is renamed to A.ASM on drive B.

If ufn1 is already present, the REN command responds with the error FILE EXISTS and not perform the change. If ufn2 does not exist on the specified disk, the message NO FILE is printed at the console.

1.4.4 SAVE Command

Syntax:
SAVE n ufn

The SAVE command places n pages (256-byte blocks) onto disk from the TPA and names this file ufn. In the CP/M distribution system, the TPA starts at 100H (hexadecimal) which is the second page of memory. The SAVE command must specify 2 pages of memory if the user's program occupies the area from 100H through 2FFH. The machine code file can be subsequently loaded and executed. The following are examples of the SAVE command:

SAVE 3 X.COM	Copies 100H through 3FFH to X.COM.
SAVE 40 Q	Copies 100H through 28FFH to Q. Note that 28 is the page count in 28FFH, and that 28H = 2 * 16 + 8 = 40 decimal.
SAVE 4 X.Y	Copies 100H through 4FFH to X.Y.

The SAVE command can also specify a disk drive in the ufn portion of the command, as shown in the following example:

SAVE 10 B:ZOT.COM	Copies 10 pages, 100H through 0AFFH, to the file ZOT.COM on drive B.

1.4.5 TYPE Command

Syntax:
TYPE ufn

The TYPE command displays the content of the ASCII source file ufn on the currently logged disk at the console device. The following are valid TYPE commands:

```
TYPE X.Y
TYPE X.PLM
TYPE XXX
```

The TYPE command expands tabs, CTRL-I characters, assuming tab positions are set at every eighth column. The ufn can also reference a drive name.

TYPE B:X.PRN	The file X.PRN from drive B is displayed.

1.4.6 USER Command

Syntax:
USER n

The USER command allows maintenance of separate files in the same directory. In the syntax line, n is an Integer value in the range 0 to 15. On cold start, the operator is automatically logged into user area number 0, which is compatible with standard CP/M 1 directories. You can issue the USER command at any time to move to another logical area within the same directory. Drives that are logged-in while addressing one user number are automatically active when the operator moves to another. A user number is simply a prefix that accesses particular directory entries on the active disks.

The active user number is maintained until changed by a subsequent USER command, or until a cold start when user 0 is again assumed.

1.5 Line Editing and Output Control

The CCP allows certain line-editing functions while typing command lines. The CTRL-key sequences are obtained by pressing the control and letter keys simultaneously. Further, CCP command lines are generally up to 255 characters in length; they are not acted upon until the carriage return key is pressed.

Character	Meaning
CTRL-C	Reboots CP/M system when pressed at start of line.
CTRL-E	Physical end of line; carriage is returned, but line is not sent until the carriage return key is pressed.
CTRL-H	Backspaces one character position.
CTRL-I	Terminates current input (line-feed).
CTRL-M	Terminates current input (carriage return).
CTRL-P	Copies all subsequent console output to the currently assigned list device (see Section 1.6.1). Output is sent to the list device and the console device until the next CTRL-P is pressed.
CTRL-R	Retypes current command line; types a clean line following character deletion with rubouts.
CTRL-S	Stops the console output temporarily. Program execution and output continue when you press any character at the console, for example another CTRL-S. This feature stops output on high speed consoles, such as CRTs, in order to view a segment of output before continuing.
CTRL-U	Deletes the entire line typed at the console.
CTRL-X	Same as CTRL-U.
CTRL-Z	Ends input from the console (used in PIP and ED).
rub/del	Deletes and echoes the last character typed at the console.

Table 1-1. Line-editing Control Characters

1.6 Transient Commands

Transient commands are loaded from the currently logged disk and executed in the TPA. The transient commands for execution under the CCP are below. Additional functions are easily defined by the user (see Section 1.6.3).

Command	Function
STAT	Lists the number of bytes of storage remaining on the currently logged disk, provides statistical information about particular files, and displays or alters device assignment.
ASM	Loads the CP/M assembler and assembles the specified program from disk.
LOAD	Loads the file in Intel HEX machine code format and produces a file in machine executable form which can be loaded into the TPA. This loaded program becomes a new command under the CCP.
DDT	Loads the CP/M debugger into TPA and starts execution.
PIP	Loads the Peripheral Interchange Program for subsequent disk file and peripheral transfer operations.
ED	Loads and executes the CP/M text editor program.
SYSGEN	Creates a new CP/M system disk.
SUBMIT	Submits a file of commands for batch processing.
DUMP	Dumps the contents of a file in hex.
MOVCPM	Regenerates the CP/M system for a particular memory size.

Table 1-2. CP/M Transient Commands

Transient commands are specified in the same manner as built-in commands, and additional commands are easily defined by the user. For convenience, the transient command can be preceded by a drive name which causes the transient to be loaded from the specified drive into the TPA for execution. Thus, the command

```
B:STAT
```

causes CP/M to temporarily log in drive B for the source of the STAT transient, and then return to the original logged disk for subsequent processing.

1.6.1 STAT Command

Syntax:
STAT
STAT "command line"

The STAT command provides general statistical information about file storage and device assignment. Special forms of the command line allow the current device assignment to be examined and altered. The various command lines that can be specified are shown with an explanation of each form to the right.

STAT	If you type an empty command line, the STAT transient calculates the storage remaining on all active drives, and prints one of the following messages: d: R/W, SPACE: nnnK d: R/O, SPACE: nnnK for each active drive d:, where R/W indicates the drive can be read or written, and R/O indicates the drive is Read-Only (a drive becomes R/O by explicitly setting it to Read- Only, as shown below, or by inadvertently changing disks without performing a warm start). The space remaining on the disk in drive d: is given in kilobytes by nnn.
STAT d:	If a drive name is given, then the drive is selected before the storage is computed. Thus, the command STAT B: could be issued while logged into drive A, resulting in the message BYTES REMAINING ON B: nnnK

STAT afn	The command line can also specify a set of files to be scanned by STAT. The files that satisfy afn are listed in alphabetical order, with storage requirements for each file under the heading:
	
	`RECS BYTES EXT D:FILENAME.TYP`
	`rrrr bbbk ee d:filename.typ`
	
	where rrrr is the number of 128-byte records allocated to the file, bbb is the number of kilobytes allocated to the file (bbb=rrrr*128/1024), ee is the number of 16K extensions (ee=bbb/16), d is the drive name containing the file (A ... P), filename is the eight-character primary filename, and typ is the three-character filetype. After listing the individual files, the storage usage is summarized.

STAT d:afn	The drive name can be given ahead of the afn. The specified drive is first selected, and the form STAT afn is executed.
STAT d:=R/O	This form sets the drive given by d to Read- Only, remaining in effect until the next warm or cold start takes place. When a disk is Read-Only, the message BDOS ERR ON d: Read-Only appears if there is an attempt to write to the Read-Only disk. CP/M waits until a key is pressed before performing an automatic warm start, at which time the disk becomes R/W.

The STAT command allows you to control the physical-to-logical device assignment. See the IOBYTE function described in Sections 5 and 6. There are four logical peripheral devices that are, at any particular instant, each assigned one of several physical peripheral devices. The following is a list of the four logical devices:

- CON: is the system console device, used by CCP for communication with the operator.
- RDR: is the paper tape reader device.
- PUN: is the paper tape punch device.
- LST: is the output list device.

The actual devices attached to any particular computer system are driven by subroutines in the BIOS portion of CP/M. Thus, the logical RDR: device, for example, could actually be a high speed reader,

teletype reader, or cassette tape. To allow some flexibility in device naming and assignment, several physical devices are defined in Table 1-3.

Device	Meaning
TTY:	Teletype device (slow speed console)
CRT:	Cathode ray tube device (high speed console)
BAT:	Batch processing (console is current RDR:, output goes to current LST: device)
UC1:	User-defined console
PTR:	Paper tape reader (high speed reader)
UR1:	User-defined reader #1
UR2:	User-defined reader #2
PTP:	Paper tape punch (high speed punch)
UP1:	User-defined punch #1
UP2:	User-defined punch #2
LPT:	Line printer
UL1:	User-defined list device #1

Table 1-3. Physical Devices

It is emphasized that the physical device names might not actually correspond to devices that the names imply. That is, you can implement the PTP: device as a cassette write operation. The exact correspondence and driving subroutine is defined in the BIOS portion of CP/M. In the standard distribution version of CP/M, these devices correspond to their names on the MDS 800 development system.

The command,

```
STAT VAL:
```

produces a summary of the available status commands, resulting in the output:

```
Temp R/O Disk d:$R/O
Set Indicator: filename.typ $R/O $R/W $SYS $DIR
Disk Status: DSK: d:DSK
Iobyte Assign:
```

which gives an instant summary of the possible STAT commands and shows the permissible logical-to-physical device assignments:

```
 CON:=TTY:CRT:BAT:UCI:
 RDR:=TTY:PTR:URI:UR2:
 PUN:=TTY:PTP:UP1:UP2:
 LST:=TTY:CRT:LPT:ULI:
```

The logical device to the left takes any of the four physical assignments shown to the right. The current logical-to-physical mapping is displayed by typing the command:

```
STAT DEV:
```

This command produces a list of each logical device to the left and the current corresponding physical device to the right. For example, the list might appear as follows:

```
CON:=CRT:
RDR:=URl:
PUN:=PTP:
LST:=TTY:
```

The current logical-to-physical device assignment is changed by typing a STAT command of the form:

```
STAT ld1=pd1,ld2=pd2,...,ldn=pdn
```

where ld1 through ldn are logical device names and pd1 through pdn are compatible physical device names. For example, ld1 and pd1 appear on the same line in the VAL: command shown above. The following example shows valid STAT commands that change the current logical-to-physical device assignments:

```
STAT CON:=CRT:
STAT PUN:=TTY:,LST:=LPT:,RDR:=TTY
```

The command form,

```
STAT d:filename.typ $S
```

where d: is an optional drive name and filename.typ is an unambiguous or ambiguous filename, produces the following output display format:

```
 Size      Recs      Bytes          Ext Acc

   48        48         6K     1 R/O A:ED.COM
   55        55        12K     1 R/O (A:PIP.COM)
65536       128        16K     2 R/W A:X.DAT
```

where the $S parameter causes the Size field to be displayed. Without the $S, the Size field is skipped, but the remaining fields are displayed. The Size field lists the virtual file size in records, while the Recs field sums the number of virtual records in each extent. For files constructed sequentially, the Size and Recs fields are identical. The Bytes field lists the actual number of bytes allocated to the corresponding file. The minimum allocation unit is determined at configuration time; thus, the number of bytes corresponds to the record count plus the remaining unused space in the last allocated block for sequential files. Random access files are given data areas only when written, so the Bytes field contains the only accurate allocation figure. In the case of random access, the Size field gives the logical end-of-file record position and the Recs field counts the logical records of each extent. Each of these extents, however, can contain unallocated holes even though they are added into the record count.

The Ext field counts the number of physical extents allocated to the file. The Ext count corresponds to the number of directory entries given to the file. Depending on allocation size, there can be up to 128K bytes (8 logical extents) directly addressed by a single directory entry. In a special case, there are actually 256K bytes that can be directly addressed by a physical extent.

The Acc field gives the R/O or R/W file indicator, which you can change using the commands shown. The four command forms,

```
STAT d:filename.typ $R/O
STAT d:filename.typ $R/W
STAT d:filename.typ $SYS
STAT d:filename.typ $DIR
```

set or reset various permanent file indicators. The R/O indicator places the file, or set of files, in a Read-Only status until changed by a subsequent STAT command. The R/O status is recorded in the directory with the file so that it remains R/O through intervening cold start operations. The R/W indicator places the file in a permanent Read-Write status. The SYS indicator attaches the system indicator to the file, while the DIR command removes the system indicator. The filename.typ may be ambiguous or unambiguous, but files whose attributes are changed are listed at the console when the change occurs. The drive name denoted by d: is optional.

When a file is marked R/O, subsequent attempts to erase or write into the file produce the following BDOS message at your screen:

```
BDOS Err on d: File R/O
```

lists the drive characteristics of the disk named by d: that is in the range A:, B:,...,P:. The drive characteristics are listed in the following format:

```
    d: Drive Characteristics
65536: 128 Byte Record Capacity
 8192: Kilobyte Drive Capacity
  128: 32 Byte Directory Eritries
    0: Checked Directory Eritries
 1024: Records/Extent
  128: Records/BlocK
   58: Sectors/TracK
    2: Reserved TracKs
```

where d: is the selected drive, followed by the total record capacity (65536 is an eight-megabyte drive), followed by the total capacity listed in kilobytes. The directory size is listed next, followed by the checked entries. The number of checked entries is usualiv identical to the directory size for removable media, because this mechanism is used to detect changed media during CP/M operation without an intervening warm start. For fixed media, the number is usually zero, because the media are not changed without at least a cold or warm start.

The number of records per extent determines the addressing capacity of each directory entry (1024 times 128 bytes, or 128K in the previous example). The number of records per block shows the basic allocation size (in the example, 128 records/block times 128 bytes per record, or 16K bytes per block). The listing is then followed by the number of physical sectors per track and the number of reserved tracks.

For logical drives that share the same physical disk, the number of reserved tracks can be quite large because this mechanism is used to skip lower-numbered disk areas allocated to other logical disks. The command form

```
STAT DSK:
```

produces a drive characteristics table for all currently active drives. The final STAT command form is

```
STAT USR:
```

which produces a list of the user numbers that have files on the currently addressed disk. The display format is

```
Active User: 0
Active Files: 0 1 3
```

where the first line lists the currently addressed user number, as set by the last CCP USER command, followed by a list of user numbers scanned from the current directory. In this case, the active user number is 0 (default at cold start) with three user numbers that have active files on the current disk. The operator can subsequently examine the directories of the other user numbers by logging in with USER 1 or USER 3 commands, followed by a DIR command at the CCP level.

1.6.2 ASM Command

Syntax:
ASM ufn

The ASM command loads and executes the CP/M 8080 assembler. The ufn specifies a source file containing assembly language statements, where the filetype is assumed to be ASM and is not specified. The following ASM commands are valid:

```
ASM
ASM GAMMA
```

The two-pass assembler is automatically executed. Assembly errors that occur during the second pass are printed at the console.

The assembler produces a file:

```
X.PRN
```

where X is the primary name specified in the ASM command. The PRN file contains a listing of the source program with embedded tab characters if present in the source program, along with the machine code generated for each statement and diagnostic error messages, if any. The PRN file is listed at the console using the TYPE command, or sent to a peripheral device using PIP (see Section 1.6.4). Note that the PRN file contains the original source program, augmented by miscellaneous assembly information in the leftmost 16 columns; for example, program addresses and hexadecimal machine code. The PRN file serves as a backup for the original source file. If the source file is accidentally removed or destroyed, the PRN file can be edited by removing the leftmost 16 characters of each line (see Section 2). This is done by issuing a single editor macro command. The resulting file is identical to the original source file and can be renamed from PRN to ASM for subsequent editing and assembly. The file

```
A.HEX
```

is also produced, which contains 8080 machine language in Intel HEX format suitable for subsequent loading and execution (see Section 1.6.3). For complete details of CP/M's assembly language program, see Section 3.

The source file for assembly is taken from an alternate disk by prefixing the assembly language filename by a disk drive name. The command

```
ASM B:ALPHA
```

loads the assembler from the currently logged drive and processes the source program ALPHA.ASM on drive B. The HEX and PRN files are also placed on drive B in this case.

1.6.3 LOAD Command

Syntax:
LOAD ufn

The LOAD command reads the file ufn, which is assumed to contain HEX format machine code, and produces a memory image file that can subsequently be executed. The filename ufn is assumed to be of the form:

```
X.HEX
```

and only the filename X need be specified in the command. The LOAD command creates a file named

```
X.COM
```

that marks it as containing machine executable code. The file is actually loaded into memory and executed when the user types the filename X immediately after the prompting character > printed by the CCP.

Generally, the CCP reads the filename X following the prompting character and looks for a built-in function name. If no function name is found, the CCP searches the system disk directory for a file by the name

```
X.COM
```

If found, the machine code is loaded into the TPA, and the program executes. Thus, the user need only LOAD a hex file once; it can be subsequently executed any number of times by typing the primary name. This way, you can invent new commands in the CCP. Initialized disks contain the transient commands as COM files, which are optionally deleted. The operation takes place on an alternate drive if the filename is prefixed by a drive name. Thus,

```
LOAD B:BETA
```

brings the LOAD program into the TPA from the currently logged disk and operates on drive B after execution begins.

Note: the BETA.HEX file must contain valid Intel format hexadecimal machine code records (as produced by the ASM program, for example) that begin at 100H of the TPA. The addresses in the hex records must be in ascending order; gaps in unfilled memory regions are filled with zeroes by the LOAD command as the hex records are read. Thus, LOAD must be used only for creating CP/M standard COM files that operate in the TPA. Programs that occupy regions of memory other than the TPA are loaded under DDT.

1.6.4 PIP

Syntax:
PIP
PlP destination=source#1,source#2,...,source#n

PIP is the CP/M Peripheral Interchange Program that implements the basic media conversion operations necessary to load, print, punch, copy, and combine disk files. The PIP program is initiated by typing one of the following forms:

```
PIP
```

```
PIP command line
```

In both cases PIP is loaded into the TPA and executed. In the first form, PIP reads command lines directly from the console, prompted with the * character, until an empty command line is typed (for example, a single carriage return is issued by the operator). Each successive command line causes some media conversion to take place according to the rules shown below.

In the second form, the PIP command is equivalent to the first, except that the single command line given with the PIP command is automatically executed, and PIP terminates immediately with no further prompting of the console for input command lines. The form of each command line is

```
destination=source#1,source#2,...,source#n
```

where destination is the file or peripheral device to receive the data, and source#1,...,source#n is a series of one or more files or devices that are copied from left to right to the destination.

When multiple files are given in the command line (for example, n>1), the individual files are assumed to contain ASCII characters, with an assumed CP/M end-of-file character (CTRL-Z) at the end of each file (see the O parameter to override this assumption). Lower-case ASCII alphabetics are internally translated to upper-case to be consistent with CP/M file and device name conventions. Finally, the total command line length cannot exceed 255 characters. CTRL-E can be used to force a physical carriage return for lines that exceed the console width.

The destination and source elements are unambiguous references to CP/M source files with or without a preceding disk drive name. That is, any file can be referenced with a preceding drive name (A: through P:) that defines the particular drive where the file can be obtained or stored. When the drive name is not included, the currently logged disk is assumed. The destination file can also appear as one or more of the source files; in which case the source file is not altered until the entire concatenation is complete. If it already exists, the destination file is removed if the command line is properly formed. It is not removed if an error condition arises. The following command lines, with explanations to the right, are valid as input to PIP:

X=Y	Copies to file X from file Y, where X and Y are unambiguous filenames; Y remains unchanged.
X=Y,Z	Concatenates files Y and Z and copies to file X, with Y and Z unchanged.
X.ASM=Y.ASM,Z.ASM	Creates the file X.ASM from the concatenation of the Y and Z.ASM files.
NEW.ZOT=B:OLD.ZAP	Moves a copy of OLD.ZAP from drive B to the currently logged disk; names the file NEW.ZOT.
B:A.U=B:B.V,A:C.W,D.X	Concatenates file B.Y from drive B with C.W from drive A and D.X from the logged disk; creates the file A.U on drive B.

For convenience, PIP allows abbreviated commands for transferring files between disk drives. The abbreviated PIP forms are

```
PIP d:=afn
PIP d1:=d2:afn
PIP ufn=d2:
PlP d1:ufn=d2:
```

The first form copies all files from the currently logged disk that satisfy the afn to the same files on

drive d, where d = A...P. The second form is equivalent to the first, where the source for the copy is drive d2 where d2 = A ... P. The third form is equivalent to the command PIP d1:ufn=d2:ufn which copies the file given by ufn from drive d2 to the file ufn on drive d1. The fourth form is equivalent to the third, where the source disk is explicitly given by d2.

The source and destination disks must be different in all of these cases. If an afn is specfied, PIP lists each ufn that satisfies the afn as it is being copied. If a file exists by the same name as the destination file, it is removed after successful completion of the copy and replaced by the copied file.

The following PIP commands give examples of valid disk-to-disk copy operations:

B=*.COM	Copies all files that have the secondary name COM to drive B from the current drive.
A:=B:ZAP.*	Copies all files that have the primary name ZAP to drive A from drive B.
ZAP.ASM=B:	Same as ZAP.ASM=B:ZAP.ASM
B:ZOT.COM=A:	Same as B:ZOT.COM=A:ZOT.COM
B:=GAMMA.BAS	Same as B:GAMMA.BAS=GAMMA.BAS
B:=A:GAMMA.BAS	Same as B:GAMMA.BAS=A:GAMMA.BAS

PIP allows reference to physical and logical devices that are attached to the CP/M svstem. The device names are the same as given under the STAT command, along with a number of specially named devices. The following is a list of logical devices given in the STAT command

```
CON: (console)
RDR: (reader)
PUN: (punch)
LST: (list)
```

while the physical devices are

```
TTY: (console , reader, punch, or list)
CRT: (console, or list), UC1: (console)
PTR: (reader), URI: (reader), UR2: (reader)
PTP: (punch), UPI: (punch), UP2: (punch)
LPT: (list), ULI: (list)
```

The BAT: physical device is not included, because this assignment is used only to indicate that the RDR: and LST: devices are used for console input/output.

The RDR, LST, PUN, and CON devices are all defined within the BIOS portion of CP/M, and are easily altered for any particular I/O system. The current physical device mapping is defined by IOBYTE; see Section 6 for a discussion of this function. The destination device must be capable of receiving data, for example, data cannot be sent to the punch, and the source devices must be capable of generating data, for example, the LST: device cannot be read.

The following list describes additional device names that can be used in PIP commands.

- NUL: sends 40 nulls (ASCII 0s) to the device. This can be issued at the end of punched output.
- EOF: sends a CP/M end-of-file (ASCII CTRL-Z) to the destination device (sent automatically at the end of all ASCII data transfers through PIP).
- INP: is a special PIP input source that can be patched into the PIP program. PIP gets the input data character-by-character, by CALLing location 103H, with data returned in location 109H

(parity bit must be zero).

- OUT: is a special PIP output destination that can be patched into the PIP program. PIP CALLs location 106H with data in register C for each character to transmit. Note that locations 109H through 1FFH of the PIP memory image are not used and can be replaced by special purpose drivers using DDT (see Section 4).
- PRN: is the same as LST:, except that tabs are expanded at every eighth character position, lines are numbered, and page ejects are inserted every 60 lines with an initial eject (same as using PIP options [t8np]).

File and device names can be interspersed in the PIP commands. In each case, the specific device is read until end-of-file (CTRL-Z for ASCII files, and end-of-data for non-ASCII disk files). Data from each device or file are concatenated from left to right until the last data source has been read.

The destination device or file is written using the data from the source files, and an end-of-file character, CTRL-Z, is appended to the result for ASCII files. If the destination is a disk file, a temporary file is created ($$$ secondary name) that is changed to the actual filename only on successful completion of the copy. Files with the extension COM are always assumed to be non-ASCII.

The copy operation can be aborted at any time by pressing any key on the keyboard. PIP responds with the message ABORTED to indicate that the operation has not been completed. If any operation is aborted, or if an error occurs during processing, PIP removes any pending commands that were set up while using the SUBMIT command.

PIP performs a special function if the destination is a disk file with type HEX (an Intel hex-formatted machine code file), and the source is an external peripheral device, such as a paper tape reader. In this case, the PIP program checks to ensure that the source file contains a properly formed hex file, with legal hexadecimal values and checksum records.

When an invalid input record is found, PIP reports an error message at the console and waits for corrective action. Usually, you can open the reader and rerun a section of the tape (pull the tape back about 20 inches). When the tape is ready for the reread, a single carriage return is typed at the console, and PIP attempts another read. If the tape position cannot be properly read, continue the read by typing a return following the error message, and enter the record manually with the ED program after the disk file is constructed.

PIP allows the end-of-file to be entered from the console if the source file is an RDR: device. In this case, the PIP program reads the device and monitors the keyboard. If CTRL-Z is typed at the keyboard, the read operation is terminated normally.

The following are valid PIP commands:

PIP LST:=X.PRN	Copies X.PRN to the LST device and terminates the PIP program.
PIP	Starts PIP for a sequence of commands. PIP prompts with *.
*CON:=X.ASM,Y.ASM,Z.ASM	Concatenates three ASM files and copies to the CON device.
*X.HEX=CON:,Y.HEX,PTR:	Creates a HEX file by reading the CON until a CTRL-Z is typed, followed by data from Y.HEX and PTR until a CTRL-Z is encountered.
	Sends 40 nulls to the punch device; copies the X.ASM file

PIP PUN:=NUL:,X.ASM,EOF:,NUL:	to the punch, followed by an end-of-file, CTRL-Z, and 40 more null characters.
(carriage return)	A single carriage return stops PIP.

You can also specify one or more PIP parameters, enclosed in left and right square brackets, separated by zero or more blanks. Each parameter affects the copy operation, and the enclosed list of parameters must immediately follow the affected file or device. Generally, each parameter can be followed by an optional decimal integer value (the S and Q parameters are exceptions). Table 1-4 describes valid PIP parameters.

Parameter	Meaning
B	Blocks mode transfer. Data are buffered by PIP until an ASCII x-off character, CTRL-S, is received from the source device. This allows transfer of data to a disk file from a continuous reading device, such as a cassette reader. Upon receipt of the x-off, PIP clears the disk buffers and returns for more input data. The amount of data that can be buffered depends on the memory size of the host system. PIP issues an error message if the buffers overflow.
Dn	Deletes characters that extend past column n in the transfer of data to the destination from the character source. This parameter is generally used to truncate long lines that are sent to a narrow printer or console device.
E	Echoes all transfer operations to the console as they are being performed.
F	Filters form-feeds from the file. All embedded form-feeds are removed. The P parameter can be used simultaneously to insert new form-feeds.
Gn	Gets file from user number n (n in the range 0-15).
H	Transfers HEX data. All data are checked for proper Intel hex file format. Nonessential characters between hex records are removed during the copy operation. The console is prompted for corrective action in case errors occur.
I	Ignores :00 records in the transfer of Intel hex format file. The I parameter automatically sets the H parameter.
L	Translates upper-case alphabetics to lower-case.
N	Adds line numbers to each line transferred to the destination, starting at one and incrementing by 1. Leading zeroes are suppressed, and the number is followed by a colon. If N2 is specified, leading zeroes are included and a tab is inserted following the number. The tab is expanded if T is set.
O	Transfers non-ASCII object files. The normal CP/M end-of-file is ignored.
Pn	Includes page ejects at every n lines with an initial page eject. If n = 1 or is excluded altogether, page ejects occur every 60 lines. If the F parameter is used, form-feed suppression takes place before the new page ejects are inserted.
QS^Z	Quits copying from the source device or file when the string S, terminated by CTRL-Z, is encountered.
R	Reads system files.
Ss^Z	Start copying from the source device when the string s, terminated by CTRL-Z, is encountered. The S and Q parameters can be used to abstract a particular section of a file, such as a subroutine. The start and quit strings are always included in the copy operation. If you specify a command line after the PIP command keyword, the CCP translates

	strings following the S and Q parameters to uppercase. If you do not specify a command line, PIP does not perform the automatic upper-case translation.
Tn	Expands tabs, CTRL-I characters, to every nth column during the transfer of characters to the destination from the source.
U	Translates lower-case alphabetics to upper-case during the copy operation.
V	Verifies that data have been copied correctly by rereading after the write operation (the destination must be a disk file).
W	Writes over R/O files without console interrogation.
Z	Zeros the parity bit on input for each ASCII character.

Table 1-4. PIP Parameters

The following examples show valid PIP commands that specify parameters in the file transfer.

PIP X.ASM=B:[V]
: Copies X.ASM from drive B to the current drive and verifies that the data were properly copied.

PIP LPT:=X.ASM[NT8U]
: Copies X.ASM to the LPT: device; numbers each line, expands tabs to every eighth column, and translates lower-case alphabetics to upper-case.

PIP PUN:=X.HEX[I],Y.ZOT[H]
: First copies X.HEX to the PUN: device and ignores the trailing :00 record in X.HEX; continues the transfer of data by reading Y.ZOT, which contains HEX records, including any :00 records it contains.

PIP X.LIB=Y.ASM[sSUBR1:^zqJMP L3^z]
: Copies from the file Y.ASM into the file X.LIB. The command starts the copy when the string SUBR1: has been found, and quits copying after the string JMP L3 is encountered.

PIP PRN:=X.ASM[p50]
: Sends X.ASM to the LST: device with line numbers, expands tabs to every eighth column, and elects pages at every 50th line. The assumed parameter list for a PRN file is nt8p60; p50 overrides the default value.

Under normal operation, PIP does not overwrite a file that is set to a permanent R/O status. If an attempt is made to overwrite an R/O file, the following prompt appears:

```
DESTINATION FILE IS R/O, DELETE (Y/N)?
```

If you type Y, the file is overwritten. Otherwise, the following response appears:

```
** NOT DELETED **
```

The file transfer is skipped, and PIP continues with the next operation in sequence. To avoid the prompt and response in the case of R/O file overwrite, the command line can include the W parameter, as shown in this example:

```
PIP A:=B:*.COM[W]
```

The W parameter copies all nonsystem files to the A drive from the B drive and overwrites any R/O files in the process. If the operation involves several concatenated files, the W parameter need only be included with the last file in the list, as in this example:

```
PIP A.DAT=B.DAT,F:NEW.DAT,G:OLD.DAT[W]
```

Files with the system attribute can be included in PIP transfers if the R parameter is included; otherwise, system files are not recognized. For example, the command line:

```
PIP ED.COM=B:ED.COM[R]
```

reads the ED.COM file from the B drive, even if it has been marked as an R/O and system file. The system file attributes are copied, if present.

Downward compatibility with previous versions of CP/M is only maintained if the file does not exceed one megabyte, no file attributes are set, and the file is created by user 0. If compatibility is required with nonstandard, for example, double-density versions of 1.4, it might be necessary to select 1.4 compatibility mode when constructing the internal disk parameter block. See Section 6 and refer to Section 6.10, which describes BIOS differences.

Note:

To copy files into another user area, PIP.COM must be located in that user area. Use the following procedure to make a copy of PIP.COM in another user area.

USER 0	Log in user 0.
DDT PIP.COM	(note PIP size s) Load PIP to memory.
GO	Return to CCP.
USER 3	Log in user 3.
SAVE s PIP.COM	

In this procedure, s is the integral number of memory pages, 256- byte segments, occupied by PIP. The number s can be determined when PIP.COM is loaded under DDT, by referring to the value under the NEXT display. If, for example, the next available address is 1D00, then PIP.COM requires 1C hexadecimal pages, or 1 times 16 + 12 = 28 pages, and the value of s is 28 in the subsequent save. Once PIP is copied in this manner, it can be copied to another disk belonging to the same user number through normal PIP transfers.

1.6.5 ED Command

Syntax:
ED ufn

The ED program is the CP/M system context editor that allows creation and alteration of ASCII files in the CP/M environment. Complete details of operation are given in Section 2. ED allows the operator to create and operate upon source files that are organized as a sequence of ASCII characters, separated by end-of-line characters (a carriage return/line-feed sequence). There is no practical restriction on line length (no single line can exceed the size of the working memory) that is defined by the number of characters typed between carriage returns.

The ED program has a number of commands for character string searching, replacement, and insertion that are useful for creating and correcting programs or text files under CP/M. Although the CP/M has a limited memory work space area (approximately 5000 characters in a 20K CP/M system), the file size that can be edited is not limited, since data are easily paged through this work area.

If it does not exist, ED creates the specified source file and opens the file for access. If the source file does exist, the programmer appends data for editing (see the A command). The appended data can then be displayed, altered, and written from the work area back to the disk (see the W command). Particular points in the program can be automatically paged and located by context, allowing easy access to particular portions of a large file (see the N command).

If you type the following command line:

```
ED X.ASM
```

the ED program creates an intermediate work file with the name

```
X.$$$
```

to hold the edited data during the ED run. Upon completion of ED, the X.ASM file (original file) is renamed to X.BAK, and the edited work file is renamed to X.ASM. Thus, the X.BAK file contains the original unedited file, and the X.ASM file contains the newly edited file. The operator can always return to the previous version of a file by removing the most recent version and renaming the previous version. If the current X.ASM file has been improperly edited, the following sequence of commands reclaim the backup file.

DIR X.*	Checks to see that BAK file is available.
ERA X.ASM	Erases most recent version.
REN X.ASM=X.BAK	Renames the BAK file to ASM.

You can abort the edit at any point (reboot, power failure, CTRL-C, or CTRL-Q command) without destroying the original file. In this case, the BAK file is not created and the original file is always intact.

The ED program allows the user to edit the source on one disk and create the back-up file on another disk. This form of the ED command is

```
ED ufn d:
```

where ufn is the name of the file to edit on the currently logged disk and d is the name of an alternate drive. The ED program reads and processes the source file and writes the new file to drive d using the name ufn. After processing, the original file becomes the back-up file. If the operator is addressing disk A, the following command is valid.

```
ED X.ASM B:
```

This edits the file X.ASM on drive A, creating the new file X.$$$ on drive B. After a successfuI edit, A:X.ASM is renamed to A:X.BAK, and B:X.$$$ is renamed to B:X.ASM. For convenience, the currently logged disk becomes drive B at the end of the edit. Note that if a file named B:X.ASM exists before the editing begins, the following message appears on the screen:

```
FILE EXISTS
```

This message is a precaution against accidentally destroying a source file. You should first erase the existing file and then restart the edit operation.

Similar to other transient commands, editing can take place on a drive different from the currently

logged disk by preceding the source filename by a drive name. The following are examples of valid edit requests:

ED A:X.ASM	Edits the file X ASM on drive A, with new file and back-up on drive A.
ED B:X.ASM A:	Edits the file X.ASM on drive B to the temporary file X.$$$ on drive A. After editing, this command changes X.ASM on drive B to X.BAK and changes X.$$$ on drive A to X.ASM

1.6.6 SYSGEN Command

Syntax:
SYSGEN

The SYSGEN transient command allows generation of an initialized disk containing the CP/M operating system. The SYSGEN program prompts the console for commands by interacting as shown.

SYSGEN
: Initiates the SYSGEN program.

SYSGEN VERSION x.x
: SYSGEN sign-on message.

SOURCE DRIVE NAME
(OR RETURN TO SKIP)
: Respond with the drive name (one of the letters A, B, C, or D) of the disk containing a CP/M system, usually A. If a copy of CP/M already exists in memory due to a MOVCPM command, press only a carriage return. Typing a drive name d causes the response:

SOURCE ON d THEN TYPE RETURN
: Place a disk containing the CP/M operating system on drive d (d is one of A, B, C, or D). Answer by pressing a carriage return when ready.

FUNCTION COMPLETE
: System is copied to memory. SYSGEN then prompts with the following:

DESTINATION DRIVE NAME
(OR RETURN TO REBOOT)
: If a disk is being initialized, place the new disk into a drive and answer with the drive name. Otherwise, press a carriage return and the system reboots from drive A. Typing drive name d causes SYSGEN to prompt with the following message:

DESTINATION ON d
THEN TYPE RETURN
: Place new disk into drive d; press return when ready.

FUNCTION COMPLETE
: New disk is initialized in drive d.

The DESTINATION prompt is repeated until a single carriage return is pressed at the console, so that more than one disk can be initialized.

Upon completion of a successful system generation, the new disk contains the operating system, and only the built-in commands are available. An IBM-compatible disk appears to CP/M as a disk with an empty directory; therefore, the operator must copy the appropriate COM files from an existing CP/M disk to the newly constructed disk using the PIP transient.

You can copy all files from an existing disk by typing the following PIP command:

```
PIP B:=A:*.*[v]
```

This command copies all files from disk drive A to disk drive B and verifies that each file has been copied correctly. The name of each file is displayed at the console as the copy operation proceeds.

Note that a SYSGEN does not destroy the files that already exist on a disk; it only constructs a new operating system. If a disk is being used only on drives B through P and will never be the source of a bootstrap operation on drive A, the SYSGEN need not take place.

1.6.7 SUBMIT Command

Syntax:
SUBMIT ufn parm#l ... parm#n

The SUBMIT command allows CP/M commands to be batched for automatic processing. The ufn given in the SUBMIT command must be the filename of a file that exists on the currently logged disk, with an assumed file type of SUB. The SUB file contains CP/M prototype commands with possible parameter substitution. The actual parameters parm#1 ... parm#n are substituted into the prototype commands, and, if no errors occur, the file of substituted commands are processed sequentially by CP/M.

The prototype command file is created using the ED program, with interspersed $ parameters of the form:

```
$1 $2 $3 ... $n
```

corresponding to the number of actual parameters that will be included when the file is submitted for execution. When the SUBMIT transient is executed, the actual parameters parm#1 ... parm#n are paired with the formal parameters $1 ... $n in the prototype commands. If the numbers of formal and actual parameters do not correspond, the SUBMIT function is aborted with an error message at the console. The SUBMIT function creates a file of substituted commands with the name

```
$$$.SUB
```

on the logged disk. When the system reboots, at the termination of the SUBMIT, this command file is read by the CCP as a source of input rather than the console. If the SUBMIT function is performed on any disk other than drive A, the commands are not processed until the disk is inserted into drive A and the system reboots. You can abort command processing at any time by pressing the rubout key when the command is read and echoed. In this case, the $$$.SUB file is removed and the subsequent commands come from the console. Command processing is also aborted if the CCP detects an error in any of the commands. Programs that execute under CP/M can abort processing of command files when error conditions occur by erasing any existing $$$.SUB file.

To introduce dollar signs into a SUBMIT file, you can type a $$ which reduces to a single $ within the command file. A caret, ^, precedes an alphabetic character s, which produces a single CTRL-X character within the file.

The last command in a SUB file can initiate another SUB file, allowing chained batch commands.

Suppose the file ASMBL.SUB exists on disk and contains the prototype commands:

```
ASM $1
DIR $1.*
```

```
ERA *.BAK
PIP $2:=$1.PRN
ERA $1.PRN
```

then, you issue the following command:

```
SUBMIT ASMBL X PRN
```

The SUBMIT program reads the ASMBL.SUB file, substituting X for all occurrences of $1 and PRN for all occurrences of $2. This results in a $$$.SUB file containing the commands:

```
ASM X
DIR X.*
ERA *.BAK
PIP PRN:=X.PRN
ERA X.PRN
```

which are executed in sequence by the CCP.

The SUBMIT function can access a SUB file on an alternate drive by preceding the filename by a drive name. Submitted files are only acted upon when they appear on drive A. Thus, it is possible to create a submitted file on drive B that is executed at a later time when inserted in drive A.

An additional utility program called XSUB extends the power of the SUBMIT facility to include line input to programs as well as the CCP. The XSUB command is included as the first line of the SUBMIT file. When it is executed, XSUB self-relocates directly below the CCP. All subsequent SUBMIT command lines are processed by XSUB so that programs that read buffered console input, BDOS Function 10, receive their input directly from the SUBMIT file. For example, the file SAVER.SUB can contain the following SUBMIT lines:

```
XSUB
DDT
I $1.COM
R
G0
SAVE 1 $2.COM
```

a subsequent SUBMIT command, such as

```
A:SUBMIT SAVER PIP Y
```

substitutes PIP for $1 and Y for $2 in the command stream. The XSUB program loads, followed by DDT, which is sent to the command lines PIP.COM, R, and G0, thus returning to the CCP. The final command SAVE 1 Y.COM is processed by the CCP.

The XSUB program remains in memory and prints the message

```
(xsub active)
```

on each warm start operation to indicate its presence. Subsequent SUBMIT command streams do not require the XSUB, unless an intervening cold start occurs. Note that XSUB must be loaded after the optional CP/M DESPOOL utility, if both are to run simultaneously.

1.6.8 DUMP Command

Syntax:
DUMP ufn

The DUMP program types the contents of the disk file (ufn) at the console in hexadecimal form. The file contents are listed sixteen bytes at a time, with the absolute byte address listed to the left of each line in hexadecimal. Long typeouts can be aborted by pressing the rubout key during printout. The source listing of the DUMP program is given in Section 5 as an example of a program written for the CP/M environment.

1.6.9 MOVCPM Command

Syntax:
MOVCPM

The MOVCPM program allows you to reconfigure the CP/M system for any particular memory size. Two optional parameters can be used to indicate the desired size of the new system and the disposition of the new system at program termination. If the first parameter is omitted or an * is given, the MOVCPM program reconfigures the system to its maximum size, based upon the kilobytes of contigous RAM in the host system (starting at 0000H). If the second parameter is omitted, the system is executed, but not permanently recorded; if * is given, the system is left in memory, ready for a SYSGEN operation. The MOVCPM program relocates a memory image of CP/M and places this image in memory in preparation for a system generation operation. The following is a list of MOVCPM command forms:

MOYCPM	Relocates and executes CP/M for management of the current memory configuration (memory is examined for contiguous RAM, starting at 100H). On completion of the relocation, the new system is executed but not permanently recorded on the disk. The system that is constructed contains a BIOS for the Intel MDS 800.
MOVCPM n	Creates a relocated CP/M system for management of an n kilobyte system (n must be in the range of 20 to 64), and executes the system as described.
MOYCPM * *	Constructs a relocated memory image for the current memory configuration, but leaves the memory image in memory in preparation for a SYSGEN operation.
MOYCPM n *	Constructs a relocated memory image for an n kilobyte memory system, and leaves the memory image in preparation for a SYSGEN operation.

For example, the command,

```
MOYCPM * *
```

constructs a new version of the CP/M system and leaves it in memory, ready for a SYSGEN operation. The message

```
READY FOR 'SYSGEN' OR
'SAYE 34 CPMxx.COM'
```

appears at the console upon completion, where xx is the current memory size in kilobytes. You can then type the following sequence:

SYSGEN
This starts the system generation.

SOURCE DRIVE NAME
(OR RETURN TO SKIP)
: Respond with a carriage return to skip the CP/M read operation, because the system is already in memory as a result of the previous MOVCPM operation.

DESTINATION DRIVE NAME
OR RETURN TO REBOOT)
: Respond with B to write new system to the disk in drive B. SYSGEN prompts with the following message:

DESTINATION ON B,
THEN TYPE RETURN
: Place the new disk on drive B and press the RETURN key when ready.

If you respond with A rather than B above, the system is written to drive A rather than B. SYSGEN continues to print this prompt:

```
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)
```

until you respond with a single carriage return, which stops the SYSGEN program with a system reboot.

You can then go through the reboot process with the old or new disk. Instead of performing the SYSGEN operation, you can type a command of the form:

```
SAVE 34 CPMxx.COM
```

at the completion of the MOVCPM function, where xx is the value indicated in the SYSGEN message. The CP/M memory image on the currently logged disk is in a form that can be patched. This is necessary when operating in a nonstandard environment where the BIOS must be altered for a particular peripheral device configuration, as described in Section 6.

The following are valid MOVCPM commands:

MOVCPM 48	Constructs a 48K version of CP/M and starts execution.
MOVCPM 48 *	Constructs a 48K version of CP/M in preparation for permanent recording; the response is READY FOR 'SYSGEN' OR 'SAVE 34 CPM48.COM'
MOVCPM	Constructs a maximum memory version of CP/M and starts execution.

The newly created system is serialized with the number attached to the original disk and is subject to the conditions of the Digital Research Software Licensing Agreement.

1.7 BDOS Error Messages

There are three error situations that the Basic Disk Operating System intercepts during file processing. When one of these conditions is detected, the BDOS prints the message:

```
BDOS ERR ON d: error
```

where d is the drive name and error is one of the three error messages:

```
BAD SECTOR
SELECT
READ ONLY
```

The BAD SECTOR message indicates that the disk controller electronics has detected an error condition in reading or writing the disk. This condition is generally caused by a malfunctioning disk controller or an extremely worn disk. If you find that CP/M reports this error more than once a month, the state of the controller electronics and the condition of the media should be checked.

You can also encounter this condition in reading files generated by a controller produced by a different manufacturer. Even though controllers claim to be IBM compatible, one often finds small differences in recording formats. The MDS-800 controller, for example, requires two bytes of ones following the data CRC byte, which is not required in the IBM format. As a result, disks generated by the Intel MDS can be read by almost all other IBM- compatible systems, while disk files generated on other manufacturers' equipment produce the BAD SECTOR message when read by the MDS. To recover from this condition, press a CTRL-C to reboot (the safest course), or a return, which ignores the bad sector in the file operation.

Note: pressing a return might destroy disk integrity if the operation is a directory write. Be sure you have adequate back-ups in this case.

The SELECT error occurs when there is an attempt to address a drive beyond the range supported by the BIOS. In this case, the value of d in the error message gives the selected drive. The system reboots following any input from the console.

The READ ONLY message occurs when there is an attempt to write to a disk or file that has been designated as Read-Only in a STAT command or has been set to Read-Only by the BDOS. Reboot CP/M by using the warm start procedure, CTRL-C, or by performing a cold start whenever the disks are changed. If a changed disk is to be read but not written, BDOS allows the disk to be changed without the warm or cold start, but internally marks the drive as Read- Only. The status of the drive is subsequently changed to Read- Write if a warm or cold start occurs. On issuing this message, CP/M waits for input from the console. An automatic warm start takes place following any input.

1.8 Operation of CP/M on the MDS

This section gives operating procedures for using CP/M on the Intel MDS microcomputer development system. Basic knowledge of the MDS hardware and software systems is assumed.

CP/M is initiated in essentially the same manner as the Intel ISIS operating system. The disk drives are labeled 0 through 3 on the MDS, corresponding to CP/M drives A through D, respectively. The CP/M system disk is inserted into drive 0, and the BOOT and RESET switches are pressed in sequence. The interrupt 2 light should go on at this point. The space bar is then pressed on the system console, and the light should go out. If it does not, the user should check connections and baud rates. The BOOT switch is turned off, and the CP/M sign-on message should appear at the selected console device, followed by the A> system prompt. You can then issue the various resident and transient commands.

The CP/M system can be restarted (warm start) at any time by pushing the INT 0 switch on the front panel. The built-in Intel ROM monitor can be initiated by pushing the INT 7 switch, which generates an RST 7, except when operating under DDT, in which case the DDT program gets control instead.

Diskettes can be removed from the drives at any time, and the system can be shut down during operation without affecting data integrity. Do not remove a disk and replace it with another without rebooting the system (cold or warm start) unless the inserted disk is Read-Only.

As a result of hardware hang-ups or malfunctions, CP/M might print the following message:

```
BDOS ERR ON d: BAD SECTOR
```

where d is the drive that has a permanent error. This error can occur when drive doors are opened and closed randomly, followed by disk operations, or can be caused by a disk, drive, or controller failure. You can optionally elect to ignore the error by pressing a single return at the console. The error might produce a bad data record, requiring reinitialization of up to 128 bytes of data. You can reboot the CP/M system and try the operation again.

Termination of a CP/M session requires no special action, except that it is necessary to remove the disks before turning the power off to avoid random transients that often make their way to the drive electronics.

You should use IBM-compatible disks rather than disks that have previously been used with any ISIS version. In particular, the ISIS FORMAT operation produces nonstandard sector numbering throughout the disk. This nonstandard numbering seriously degrades the performance of CP/M, and causes CP/M to operate noticeably slower than the distribution version. If it becomes necessary to reformat a disk, which should not be the case for standard disks, a program can be written under CP/M that causes the MDS 800 controller to reformat with sequential sector numbering (1-26) on each track.

Generally, IBM-compatible 8-inch disks do not need to be formatted. However, 5 1/4-inch disks need to be formatted.

Back to title page