DIGITAL RESEARCH

Post Office Box 579, Pacific Grove, California 93950, (408) 373-3403

An Introduction to CP/M Features and Facilities

Version 1.3

Table of Contents

Section Page

An Introduction to CP/M Features and Facilities

1. GENERAL

CP/M is a monitor control program for microcomputer system development which uses IBM-compatible flexible disks for back-up storage. Using a computer mainframe based upon Intel's 8080 microcomputer, CP/M provides a general environment for program construction, storage, and editing, along with assembly and program check-out facilities.

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 distinct programs can be stored in both source and machine-executable form.

CP/M also supports a powerful context editor, Intel compatible assembler, and debugger subsystems. When coupled with CP/M's console command processor, the resulting facilities equal or excel similar large computer facilities.

CP/M is logically divided into several distinct parts:

BIOS - the basic I/O system

BDOS - the basic disk operating system

CCP - the console command processor

TPA - the transient program area

The BIOS provides the primitive operations necessary to interface standard peripherals (teletype, CRT, Paper Tape Reader/Punch, and user-defined peripherals), and can be tailored by the user 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 which provide fully dynamic file construction while minimizing head movement across the disk during access. Any particular file may contain any number of records, not exceeding the size of any single disk (240 records of 128 bytes each). In a standard CP/M system, each disk can contain up to 64 distinct files. The BDOS has entry points which include the following primitive operations:

SEARCH	look for a particular disk file by name
OPEN	open a file for further operations
CLOSE	close a file after processing
RENAME	change the name of a particular file

READ	read a record from a particular file
WRITE	write a record onto the disk
SELECT	select a particular disk drive for further operations

The CCP provides symbolic interface between the user´s 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 assemblers, editors, and debuggers. The standard commands which are available in the CCP are listed in a following section.

The last segment of CP/M is the area called the TPA. The transient program area holds programs which are loaded from the disk under command by 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.

It should be mentioned that any or all of the CP/M component subsystems can be "overlayed" 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 at all times; 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.

It should be reiterated that the CP/M operating system is partitioned into distinct modules, including the BIOS portion which defines the hardware environment in which CP/M is executing. Thus, the standard system can be easily modified to any nonstandard environment by changing the peripheral drivers to handle the custom system. The standard system is provided with I/O drivers for Intel´s MDS microcomputer development system, along with a general discussion of the modification technique.

2. FUNCTIONAL DESCRIPTION OF CP/M.

The user interacts with CP/M primarily through the CCP which reads and interprets commands input through the console. In general, the CCP addresses one of several disks which are online (the standard system addresses up to two different disk drives). These drives are labelled disk "A", "B", and so-forth. A disk is "logged in" if the CCP is currently addressing the disk. In order 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 brought in from disk A, and the CCP displays the message

```
xxK CP/M VER m.m
```

where xx is the memory size (in kilobytes) which this CP/M system manages, and m.m is the CP/M version number. The CCP then automatically logs-in disk A, and prompts the user 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.

2.1. GENERAL COMMAND STRUCTURE.

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

ERA	remove files from the logged disk
DIR	list names of the files on the logged disk
REN	rename the specified file on the logged disk
SAVE	save the specified file on the logged disk
TYPE	type the contents of a file on the logged disk

Nearly all of the commands reference a particular file or group of files. Thus, the form of a file reference is specified below.

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 can be either "unambiguous" or "ambiguous". An unambiguous file reference uniquely identifies a single file, while an ambiguous file reference may be satisfied by a number of different files.

File references consist of two parts: the primary name and the secondary name. Although the secondary name is optional, it usually is generic; that is, the secondary name "ASM" for example, is used to denote that the file is an assembly language source file, while the primary name distinguishes each particular source file. The two names are separated by a "." as shown below:

```
pppppppp.sss
```

Where pppppppp represents the primary name of eight characters or less, and sss is the secondary name of no more than three characters. As mentioned above, the name

```
pppppppp
```

is also allowed and is equivalent to a secondary name consisting of three blanks. The characters used in specifying an unambiguous file reference cannot contain any of the 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 "?" may be interspersed throughout the primary and secondary names. In various commands throughout CP/M, the "?" symbol indicates that any file name satisfies a match if it matches exactly in all character positions where "?" appears. Thus, the ambiguous reference

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

is satisfied by the unambiguous file names

```
XYZ.COM
              and
X3Z.CAM
```

Note that the ambiguous reference

```
*.*
```

is equivalent to the ambiguous file reference

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

while

```
pppppppp.*
              and
*.sss
```

are abbreviations for

```
pppppppp.???
                   and
????????.sss
```

respectively. As an example,

```
DIR *.*
```

is interpreted by the CCP as a command to list the names of all disk files in the directory, while

```
DIR X.Y
```

searches only for a file by the name X.Y Similarly, the command

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

causes a search for all (unambiguous) file names on the disk which satisfy this ambiguous reference.

The following file names are valid unambiguous file references:

```
X          XYZ           GAMMA

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

3. SWITCHING DISKS.

The operator can switch the currently logged-in disk by typing the disk drive name (A, B, ...) followed by a colon (:) when the CCP is waiting for console input. Thus, the sequence of prompts and commands shown below might occur after the CP/M system is loaded from disk A:

```
16K CP/M VER 1.Ø
A>DIR *.*                list all files on disk A
SAMPLE   ASM
SAMPLE   PRN
A>B:                     switch to disk B
B>DIR *.ASM              list all "ASM" files on B
DUMP     ASM
FILES    ASM
B>A:                     switch back to A
```

4. THE FORM OF BUILT-IN COMMANDS.

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

ufn	-	unambiguous file reference
afn	-	ambiguous file reference
cr	-	carriage return

Further, note 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 file references.

4.1 ERA afn cr

The ERA (erase) command removes files from the currently logged-in disk (i.e., the disk name currently prompted by CP/M preceding the ">"). The files which are erased are those which 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 which satisfy the ambiguous reference X?Y.C?M are deleted

4.2. DIR afn cr

The DIR (directory) command causes the names of all files which satisfy the ambiguous file name afn to be listed at the console device. The command

```
DIR *.*
```

for example, lists the files on the currently logged disk.

Valid DIR commands are:

```
DIR X.Y

DIR X?Z.C?M

DIR ??.Y
```

4.3. REN ufn1=ufn2 cr

The REN (rename) command allows the user 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. The CCP also allows the user to type a left-oriented arrow instead of the equal sign, if the user's console supports this graphic character. Examples of the REN command are:

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

```
REN XYZ.COM=XYZ.XXX      The file XYZ.XXX is changed to XYZ.COM
```

4.4. SAVE n ufn cr

The SAVE command places n pages (256 byte blocks) onto disk from the TPA and names this file ufn. The machine code file can be subsequently loaded and executed. Examples are:

```
SAVE  3  X.COM

SAVE  4Ø  Q

SAVE  4  X.Y
```

(Note that n is a decimal value).

4.5. TYPE ufn cr

The TYPE command displays the contents of the ASCII source file ufn on the currently logged disk at the console device. Valid TYPE commands are

```
TYPE  X.Y

TYPE  X.C

TYPE  XXX
```

The TYPE command expands tabs (clt-I characters), assumming tab positions are set at every eighth column.

5. LINE EDITING.

The CCP allows certain line editing functions while typing the command.

rubout	delete and echo the last character typed at the console
ctl-U	delete the entire line typed at the console
ctl-E	physical end of line, carriage is returned, but line is not sent until the carriage return key is depressed
ctl-C	CP/M system reboot (warm start)
ctl-Z	end-of-input from the console (used in PIP and ED)

Note that the ctl-x sequence shown above denotes that the control key and the key x are depressed simultaneously.

6. TRANSIENT COMMANDS.

Transient commands are loaded from the system disk and executed in the TPA. The transient commands defined with the CCP are:

STAT	List the number of bytes of storage remaining on the currently logged disk
ASM	load the CP/M macro assembler and assemble the specified program from disk
LOAD	load the file in Intel "hex" machine code format and produce a file in machine executable form which can be loaded into the TPA
DDT	load the CP/M debugger into the TPA and start execution
PIP	load the Peripheral Interchange Program for subsequent media conversion operations
ED	load and execute the CP/M text editor program
SYSGEN	Create a new CP/M system diskette
SUBMIT	Submit a file of commands for batch processing
DUMP	Dump the contents of a file in hex

Transient commands are specified in the same manner as built-in commands, and additional commands can be easily defined by the user. The basic transient commands are listed in detail below.

6.1. STAT cr

The STAT transient command examines the storage map for the currenly logged diskette and prints a message in the format

```
xxxK BYTES REMAINING
```

where xxx is the number of kilobytes of storage available

6.2. ASM ufn cr

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

```
ASM X

ASM GAMMA
```

The two pass assembler is automatically executed. If assembly errors occur during pass 2, the errors 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 imbedded 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 can be listed at the console using the TYPE command, or sent to a peripheral device using PIP (see the PIP command structure below). The file

x.HEX

is also produced which contains 8080 machine language in Intel "hex" format suitable for subsequent loading and execution (see the LOAD command). For complete details of CP/M assembly language program, see the "CP/M Assembler Language (ASM) User's Guide."

6.3. LOAD ufn cr

The LOAD command reads the file ufn, which is assumed to contain "hex" format machine code, and produces a memory imaage file which can be subsequently executed. The file name ufn is assumed to be of the form

x.HEX

and thus only the name x need be specified in the command. If a file x.HEX does not exist, the LOAD command reads the current RDR: device instead of a disk file. The LOAD command creates a file named

x.COM

which marks it as a machine executable code. The file is actually loaded into memory and executed when the user types the name x immediately after the prompting character ">" printed by the CCP.

In general, the CCP reads the name 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 simply typing the primary name. In this way, the user can "invent" new commands in the CCP. In fact, initialized disks have the transient commands as COM files, and thus can be deleted at the users option.

6.4. PIP cr

PIP is the CP/M Peripheral Interchange Program which 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 cr
PIP command-line cr
```

In both cases, PIP is loaded into the TPA and executed. In the first case, PIP reads command lines directly from the console, prompted with the character "*" until an empty command line is typed (i.e., 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. The second form of 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 cr
```

where "destination" is the file or peripheral device to receive the data, and "source#1, ..., source#n" represents a series of one or more files or devices which are copied from left to right to the destination. A CP/M end of file mark (ctl-Z) is inserted as the last character if the destination is an ASCII file (all files except ".COM" files are treated as ASCII files in the current CP/M implementation). The equal symbol (=) can be replaced by a left-oriented arrow if your console supports this ASCII character, to improve readability. 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 (the ctl-E control can be used to force a physical carriage return for lines which exceed the console width).

The destination and source elements can be 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:, B:, C:, ...) which defines the particular drive to fetch or store the file. When the drive name is not included, the currently logged disk is assumed. Further, 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 the destination file already exists, it 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 cr	copy to file x from file y, y remains unchanged
x = y,z cr	concatenate files y and z and copy to file x, with y and z unchanged
X.ASM=Y.ASM,Z.ASM,FIN.ASM cr	create the file X.ASM from the concatenation of the Y, Z, and FIN files with type ASM
NEW.ZOT = B:OLD.ZAP cr	move a copy of OLD.ZAP from drive B to the currently logged disk, and name the file NEW.ZOT
B:A.U = B:B.V,A:C.W,D.X cr	concatenate file B.V from drive B with C.W from drive A and D.X from the logged disk, and create the file A.U on drive B

PIP also allows reference to physical and logical devices which are attached to the CP/M system. The device names are three character identifiers, followed by the colon (:) symbol. The device names are

RDR:	Paper tape reader
LST:	Listing device (printer)
PUN:	Paper tape punch
TTY:	Teletype device
CRT:	Cathode ray tube display
ARD:	Addmaster paper tape reader
IRD:	Intel or Icom paper tape reader
PRN:	Tally printer device
CON:	Currently defined console device

The RDR, LST, PUN, and CON devices are all defined within the BIOS portion of CP/M, and thus are easily altered to any particular I/O system. The TTY and CRT devices are present to support the Intel "iobyte" function which allow a simple logical to physical device mapping (see the CP/M Interface Guide for a discussion of the iobyte function). ARD, IRD, and PRN are three popular peripheral devices which have dedicated input/output ports in the CP/M environment. Tab characters (ctl-I) are expanded when the destination device is not the punch. The allowable destination devices are

LST PUN TTY CRT PRN CON

while the allowable source devices are

RDR TTY CRT ARD IRD CON

When devices are used as input, the end of file is indicated by a ctl-Z (the CP/M end of file standard) or, in the case of the ARD and IRD devices, a sequence of 255 rubout characters which is obtained by running the reader with no paper tape.

It should also be noted that 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. It is usually sufficient to open the reader and rerun a section of the tape (pull the tape back about 20 inches). When the tape is ready for the re-read, type a single carriage return at the console, and PIP will attempt another read. If the tape position cannot be properly read, simply 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.

Valid PIP commands are shown below

pip lst: = x.prn cr	copy x.prn to the LST device and terminate the PIP program
pip cr	start PIP for a sequence of commands (PIP prompts with "*")
*con:=x.asm,y.asm,z.asm cr	concatenate three ASM files and copy to the CON device
*x.hex=con:,y.hex,ard: cr	create a HEX file by reading the CON (until a ctl-Z is typed), followed by data from y.hex, followed by data from ARD until a ctl-Z or 255 rubouts are encountered.
*cr	Single carriage return stops PIP

6.6. ED ufn cr

The ED program is the CP/M system context editor, which allows creation and alteration of ASCII files in the CP/M environment. Complete details of operation are given the ED user's manual "ED: a Context Editor for the CP/M Disk System." In general, ED allows the operator to create and operate upon source files which 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) but is instead defined by the number of characters typed between cr's. The ED program has a number of commands for character string searching, replacement, and insertion, which are useful in the creation and correction of programs or text files under CP/M. Although the CP/M has a limited memory work space area (approximately 6000 characters in a 16K CP/M system), the file size which can be edited is not limited, since data is easily "paged" through this work area.

Upon initiation, ED creates the specified source file if it does not exist, and opens the file for access. The programmer then "appends" data from the source file into the work area, if the source file already exists (see the

-mand) for editinq. 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 (see the N command) allowing easy access to particular portions of a large file.

Given that the operator has typed

```
ED X.ASM cr
```

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. Suppose, for example, that the current X.ASM file was improperly edited, the sequence of CCP command shown below would reclaim the backup file

```
DIR X.*                check to see that BAK file
                       is available
ERA X.ASM              erase most recent version
REN X.ASM=X.BAK        rename the BAK file to ASM
```

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

The ED user´s manual should be consulted for complete operating details.

6.6. SYSGEN cr

The SYSGEN transient command allows generation of an initialized diskette containing the CP/M operating system. The SYSGEN program prompts the console for commands, with interaction as shown below

```
SYSGEN cr                     initiate the SYSGEN program
*SYSGEN VERSION m.m           SYSGEN signon message
GET SYSTEM? (Y/N)             If a memory image of the CP/M
                              is not present (see CP/M inter-
                              face guide) type N, otherwise
                              type Y.  Normally type Y.
SOURCE ON B THEN TYPE RETURN  Place a diskette containing the
                              CP/M operating system on drive
                              B (it´s ok to remove the one
                              that you are using on drive A)
                              and follow with a return when
```

	ready.
FUNCTION COMPLETE	System is copied to memory
PUT SYSTEM? (Y/N)	If a new diskette is being built, type Y; otherwise type N. Normally type Y.
DESTINATION ON B THEN TYPE RETURN	Place new diskette into drive B, type return when ready.
FUNCTION COMPLETE	New diskette is initialized in drive B

The SYSGEN program then reboots the system from drive A. Upon completion of a successful system generation, the new diskette contains the operating system, and only the built-in commands are available. A factory-fresh IBM-compatible diskette appears to CP/M as a diskette with an empty directory, and thus the operator must copy the appropriate COM files from an existing CP/M diskette to the newly constructed diskette using the PIP transient.

It should be noted that a SYSGEN does not destroy the files which already exist on a diskette; it results only in construction of a new operating system. Further, if a diskette is being used only on drive B, and will never be the source of a bootstrap operation on drive A, the SYSGEN need not take place, and, in fact, a new diskette needs absolutely no initialization to be used with CP/M.

6.7. SUBMIT ufn parm#1 parm#2 ... parm#n cr

The SUBMIT command allows CP/M commands to be batched together for automatic processing. The ufn given in the SUBMIT command must be the filename of a file which 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 which 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 number of formal and actual parameters does not correspond, then 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. Further, the user can abort command processing at any time by typing a rubout when the command is read and echoed. In this case, the $$$.SUB file is removed, and the subsequent commands come from the console. Command processing is also aborted if the CCP detects an error in any of the commands. Programs which execute under CP/M can abort processing of command files when error conditions occur by simply erasing any existing $$$.SUB file.

The last command in a SUB file can initiate another SUB file, thus 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
```

and the command

```
SUBMIT ASMBL X PRN cr
```

is issued by the operator. The SUBMIT program reads the ASMBL.SUB file, and substitutes "X" for all occurrences of $1, and "PRN" for all occurrences of $2, resulting in a $$$.SUB file containing

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

which are executed in sequence by the CCP.

6.8. DUMP ufn cr

The DUMP program types the contents of the disk file given by ufn at the console in hexadecimal form. The file contents is 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 pushing the rubout key during printout. (The source listing of the DUMP program is given in the CP/M interface guide, as an example of program written for the CP/M environment.)

7. OPERATION OF CP/M ON THE MDS.

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

CP/M is initiated in easentially the same manner as Intel's ISIS operating system. The disk drives are labelled 0 and 1 on the MDS, corresponding to CP/M's drive A and B, respectively. The CP/M system diskette is inserted into drive 0, and the BOOT and RESET switches are depressed in sequence. The interrupt 2 light should go on at this point. The space bar is then depressed on the device which is to be taken as the system console, and the light should go out (if it does not, then check connections and baud rates). The BOOT switch is then turned off, and the CP/M signon message should appear at the selected console device, followed by the "A>" system prompt. The user 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, 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. Note, however, that the user must not remove a diskette and replace it with another without rebooting the system (cold or warm start) unless the inserted diskette is read-only.

Due to hardware hang-ups or malfunctions, CP/M may type the message

```
PERM ERR DISK x
```

where x is the drive name which has the permanent error. This error may occur when drive doors are opened and closed randomly, followed by disk operations, or may be due to a diskette, drive, or controller failure. The user can optionally elect to ignore the error by typing a single return at the console. The error may produce a bad data record, requiring re-initialization of up to 128 bytes of data. The operator can reboot the CP/M system and try the operation again.

Termination of a CP/M session requires no special action, although it is best to remove the diskettes before turning the power off, to avoid random transients which could make their way to the drive electronics.