VECTOR

VECTOR 4 CP/M

Programmer's Guide

VECTOR 4 CP/M

PROGRAMMER'S GUIDE

Version 1.0

Revision A

September 1, 1982

P/N 7100-0003

DISCLAIMER

Vector Graphic makes no representations or warranties with respect to the contents of this manual itself, whether or not the product it describes is covered by a warranty or repair agreement. Further, Vector Graphic reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Vector Graphic to notify any person of such revision or changes, except when an agreement to the contrary exists.

REVISION NUMBERS

The date and revision of each page herein appears at the bottom of each page. The revision letter, such as A or B, changes if the manual has been improved but the product itself has not been significantly modified. The date and revision on the Title Page corresponds to that of the page most recently revised. When the product itself is modified significantly, the product will get a new revision number, as shown on the manual's Title Page, and the manual will revert to revision A, as if it were treating a brand new product. THIS MANUAL SHOULD ONLY BE USED WITH THE PRODUCT(S) IDENTIFIED ON THE TITLE PAGE.

TRADEMARK

CP/M is a registered trademark of **Digital Research.**

FOREWORD

Audience

The Vector 4 CP/M Programmer's Guide is designed for System Programmers.

Scope

The Vector 4 CP/M Programmer's Guide covers all aspects of the Vector 4 CP/M Operating System. It describes the command structure along with the numerous BDOS functions. It also lists all the BIOS Jump Routines.

Organization

The Vector 4 CP/M Programmer's Guide is organized into five sections. The first section gives an overview of Vector 4 CP/M. Sections II and III describe the Resident and Transient Commands. The last two sections give the BDOS system calls and the BIOS Jump Routines.

TABLE OF CONTENTS

LIST OF EXHIBITS

SECTION I - INTRODUCTION TO VECTOR 4 CP/M

1.1 HARDWARE PROVISIONS

A. Disk Drives

There are two types of disk drives that are supported by the VECTOR 4 CP/M Operating System: the 5 1/4-inch floppy, and the 5 1/4-inch hard disk.

B. Diskettes

The Vector 4 uses an industry standard 5 1/4-inch diskette with 16 'hard sectors'. This diskette has 16 sector holes and one additional index hole around the edge of the center hole. They are usually marked with the number '16' on the label. You can purchase them from computer stores or from other computer supply sources.

1.2 GENERAL CONVENTIONS

The following keyboard and entry conventions are used throughout the VECTOR 4 CP/M Programmer's Guide.

Key Strokes

Return key =	**[RETURN]**
Control key =	**[CTRL x]** where x = any of the keys used in a control sequence; both keys are pressed together simultaneously.
Escape key =	**[ESC]**
Tab key =	**[TAB]**
Spacebar =	**[SPACE]**
Backspace =	**[BACKSPACE]**
Delete key =	**[DEL]**
Line Feed =	**[LF]**
Shift =	**[SHIFT]**
Options =	<option(s)> where any option(s) are enclosed in < >.

Entry

A command line within VECTOR 4 CP/M consists of several components:

- Command
- Drive(s)
- Filename(s) and extension(s)
- Option(s), if available
- [RETURN] to process

for instance, the command line to copy a file:

PIP A:=B:MEMORITE.*[V] [RETURN]

which means **Copy** to **Drive A all** files from **Drive B** having a filename of **MEMORITE** with **any extension** and **verify** the transfer of data. [RETURN] represents the key which is pressed to process the command.

Note: All options within PIP, i.e., the 'V' in the above command line, must have brackets around the option; the option to verify, [V] with brackets, must be part of the command line in order to verify the transfer of data (see Paragraph 3.2.7 for more information on the PIP command).

1.3 INSTALLING VECTOR 4 CP/M

VECTOR 4 CP/M is shipped on a System diskette, and, in the case of a hard disk system, on the hard disk as well as on a diskette. It is extremely important to backup the System diskette immediately because it is possible that the hard disk could be damaged and the information contained on it destroyed. It would therefore be impossible to generate another system on the hard disk if the System diskette has been destroyed or damaged, and no backup copy had been made.

A. Diskette Copy

To install or place the VECTOR 4 CP/M operating system onto a blank double-sided diskette enabling you to boot, or bring up, the system from that diskette, simply copy the VECTOR 4 CP/M System diskette to a 'personalized' diskette by using the DISKCOPY program (see Paragraph 3.2.9).

To install the operating system on a diskette already in use, copy the operating system by using the GENSYS program (see Paragraph 3.2.1-A.).

To install the operating system onto a blank diskette which is to be used for data (other than a System diskette), it is necessary to initialize the diskette first (a new diskette must first be 'initialized' by using the FORMAT (see Paragraph 3.2.8-B) program before transferring any program or data files to it). It is then possible to selectively add files to or erase files from that diskette as desired. To transfer or copy the desired files onto the diskette, use the SAVE command, the PIP utility, or a higher level language.

NOTE: Do NOT format the VECTOR 4 CP/M System diskette or any other diskette containing desired information. Formatting completely erases a diskette.

B. Hard Disk Copy

To generate an operating system on the 5 Mbyte hard disk system, see Paragraph 3.2.1-B.

1.4 DISKETTE BACKUP, REPLACEMENT, AND PROTECTION

As with any magnetic storage medium, the recording gradually deteriorates over time. Even if a diskette is not damaged, it will begin producing errors after a very long period of use.

A. Backup

The best method for insuring against loss of diskette-based data is to maintain a backup diskette for each diskette you use. In the business world, this is considered dogma. Copy a diskette onto its backup whenever you cannot afford to lose the information stored since you last backed it up; this goes for programs as well as data. If you are operating business programs such as Inventory or Accounts Receivable, maintain a regular backup schedule, once a week In addition, a transaction journal, i.e., a printed copy of entries made each day into the system, is an excellent idea to build into business software as a 'last resort' backup.

B. Replacement

In addition to being backed up, replace frequently used diskettes by copying each to a fresh diskette every six months. A good suggestion is to use the backup diskette, which is fairly fresh, as the new front-line diskette, and to create a fresh backup. Do not wait until a frequently used diskette fails before you replace it with the backup. To create backup diskettes (i.e., to copy diskettes) use the DISKCOPY utility.

C. Protection

Write-protect tabs come in boxes of new diskettes; if you attach a tab over the write-protect cutout on a 5 1/4-inch diskette, the disk drive will not allow you to erase or change any information on the diskette. The tab may be removed later, if required.

1.5 HANDLING ERRORS

Bad sector disk errors refer to errors reported on the screen (when CP/M is running) as:

ERROR - drive:<error message>

indicating something is wrong with the diskette, the drive, or the disk controller board. There are several error situations which the Basic Disk Operating System (BDOS) intercepts during file processing. When one of these conditions is detected, the BDOS prints the message:

ERROR - drive:<error>

where <error> is the one of the following messages:

DISK READ/WRITE(xxyy)

The DISK READ/WRITE message will occur when the disk controller electronics has detected an error condition in reading or writing to the diskette ('xx' is a hex code denoting the particular physical BIOS error detected by the physical disk driver; 'yy' corresponds to the BDOS physical and logical error codes). This condition is generally due to a malfunctioning in the disk controller or the result of an extremely worn diskette.

If your system reports this error more than once a month, it may require servicing or your media may need replacement. Recovery from this condition may be accomplished by responding to the next prompt:

PRESS [CTRL C] TO REBOOT OR [RETURN] TO CONTINUE

If a [CTRL C] is entered, the current program will be aborted and control returned to the CCP *; if a [RETURN] is entered, the error will be ignored by the program and execution will continue normally. Pressing [RETURN] would only be used when copying a file (using PIP), when it is mandatory that as much of the file is recovered as possible. Note, however, that entering a [RETURN] may destroy diskette integrity if the operation is a directory write; therefore, make sure an adequate backup has been made.

* See Section 1.6.2 for a description of the CCP.

NOTE: The following BDOS errors will display the particular error message and simply wait for user input to reboot; entering any key will cause a warm boot.

DRIVE SELECT/

The DRIVE SELECT message will occur when there is an attempt to access a drive that is not supported by the current hardware/software configuration. In this case, the drive which is out of range is displayed along with the error message. The system reboots following any input from the terminal.

DISK IS READ-ONLY

The DISK READ-ONLY message will occur when there is an attempt to write to a diskette which has been designated as 'READ/ONLY' using the STAT command, or has been set to 'READ-ONLY' by the BDOS. In general, the user should warm boot the system, using [CTRL C], whenever physically changing a diskette. If a changed diskette is to be read but not written to, the system allows the diskette to be changed without requiring a warm boot, but internally marks the drive as Read-Only. The status of the drive is subsequently changed back to 'Read/Write' following a warm boot. Upon displaying the READ-ONLY message, CP/M waits for an input from the terminal; an automatic warm boot will occur following any user input.

FILE IS READ-ONLY

The FILE READ-ONLY message will occur when there is an attempt to write to a file which has been declared as 'R/O' using the STAT command or by an applications program, or if the file is opened in READ-ONLY mode. The file may be returned to READ/WRITE status using the STAT command, if necessary.

FILE CURRENTLY OPEN

The FILE CURRENTLY OPEN message will occur if an attempt is made to access a file which is in use and locked by another user.

ILLEGAL FILE ACCESS

The ILLEGAL FILE ACCESS message will occur when an attempt is made to access a file that is not open.

FILE ALREADY EXISTS

The FILE ALREADY EXISTS message will occur when a user or program is trying to create, rename, or spool a specific file which already exists on disk.

ILLEGAL '?' IN FCB

The ILLEGAL '?' IN FCB message will occur when a question mark '?' is used within the File Control Block (FCB) in any file function other than SEARCH FIRST, SEARCH NEXT, and DELETE FILE.

FILE OPEN LIST FULL

The FILE OPEN LIST FULL message will occur when an attempt is made to OPEN or CREATE a file and the System File List is full (32 files maximum - list is maintained by the operating system of open files in the system, the task(s) which have the files open, and the mode in which the file is open).

RECORD LOCK LIST FULL

The RECORD LOCK LIST FULL message will occur when an attempt is made to LOCK a record and the Record Lock List is full (512 maximum - series of lists linked to the open file entries of the System File List describing which records are locked, and which task has them locked). NOTE: records can only be locked while in unlocked mode.

DRIVE NOT READY

The DRIVE NOT READY message will occur when an attempt is made to read from a drive and the disk is either not properly seated or there is no disk in the drive.

If any of the above errors occur, repeat the procedure that resulted in the error. If the error occurs again, refer to the section below on recovery techniques. If you consistently get such disk errors, switch to another of the suggested brands of diskettes. Try this before obtaining hardware service; if the errors persist, contact your service representative.

If you repeatedly get bad sector disk errors using one particular diskette, then it is probably defective. This will sometimes happen with a new diskette when you are reformatting it or using it to back up another diskette. After several attempts, discard it or return it if possible. Whenever you repeat a disk operation after an error, always unload and reload the diskette, because it may be seated incorrectly.

If an old diskette repeatedly gives bad sector errors, first repeat the operation several times, unloading and reloading the diskette each time. If there is still a problem, check the center hole. If it is wrinkled, straighten it out with your fingers and then try again. If you still get bad sector errors, try copying the diskette to another diskette using the DISKCOPY utility in CP/M. If the error still occurs, try switching source and destination drives. Some combination of drives and repositioning of diskettes within drives will often result in a successful copy. If you cannot copy a diskette at all, then copy it file by file to another initialized diskette using the PIP utility. There may be one or two files which will not copy, but hopefully all the other files will copy over successfully.

When you get a bad sector error message, CP/M allows you to skip over the bad sector and continue the operation. This is not usually desireable because the bad data will cause other problems. However, if there is no other way to get the file off the diskette intact, simply bypass the error message by pressing the [RETURN] key immediately after the bad sector error message occurs. The disk operation will continue, ignoring the fact that a bad sector was transferred.

1.6 DESCRIPTION OF VECTOR 4 CP/M

1.6.1 General

VECTOR 4 CP/M is a high-performance operating system which uses table driven techniques to allow field configuration to match a wide variety of disk capacities.

All of the fundamental file restrictions are removed, while maintaining upward compatibility from previous versions. The directory size can be field configured to contain any reasonable number of entries, and each file is optionally tagged with READ-ONLY and READ/WRITE system attributes. Users of CP/M can physically separate groups of files by user areas, with facilities for file copy operations from one user area to another. Powerful relative-record random access functions are present in CP/M which provide direct access to any of the 65536 records of an eight megabyte file.

In the VECTOR 4 CP/M environment, the user is given a separate bank of memory to use exclusively. Within each drive, the user has access to 16 user areas (Ø through 15). These user areas are accessed in the very same manner as accessing a different drive; i.e., to change drives, the user simply enters which drive is required:

A>B:

which equates to moving from Drive A to Drive B. In the same way, the user can change user areas by entering which user area number is wanted; e.g.:

A>USER 5

which moves the user from the current user area (usually User Ø since CP/M boots into User Ø) to User 5.

Although a computer can only perform a single operation at a time, it operates at such high speed that it can service several tasks at one time (using the background printing feature); this is done by switching between tasks, giving a portion of time (called a time slice) to both tasks. When that task's time slice is used up, the system will suspend the operation of that task and allow the other task to have a time slice. This time slice operation happens constantly between both tasks or tasks on the system; however, because a computer operates at very high speeds, this sharing of time is generally transparent to the user. A task may lose its time slice if, for any reason, it is waiting for an input or output process to complete, or if it is waiting for a shared resource, such as a printer.

The purpose of VECTOR 4 CP/M is to execute commands and allow the user convenient access to all of the hardware and software resources provided by the computer. CP/M provides rapid access to programs through a comprehensive file management package. The file system 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.

The part of the operating system that switches (swaps) tasks is called the scheduler. The scheduler is responsible for saving all the registers, the stack pointer, and program counter of the current task into the Task Control Block (TCB). The scheduler will look for a task that is ready to run and, once found, will reload all the registers with the values that were saved for that task when it was swapped out. If, while the scheduler is looking for a task that is ready to run, it finds a task that is waiting for an event to occur, the scheduler will check to see if the event has occured, and if it has, the task's status will be set to ready.

When the Vector system is turned on, or powered up, the following banner and prompt will display on the screen:

```
|VECTOR 4 SBC EXECUTIVE 1.Ø|

SBC>
```

VECTOR 4 CP/M, once booted, will display the following messages on all terminals:

```
VECTOR 4 CP/M - VERSION 1.Ø
PRINTER 1 LOADED WITH  x
PRINTER 2 LOADED WITH  x

A>
```

where 'x' is equivalent to one of the following printer driver messages, depending on what kind of printer is available and what printer configuration has been enabled (through the CONFIG P or CONFIG program):

```
A DRAFT DRIVER
A WORD PROCESSING DRIVER
NO DRIVER
AN UNKNOWN DRIVER
```

The following is a typical display after enabling the printer drivers:

```
VECTOR 4 CP/M - VERSION 1.Ø
PRINTER 1 LOADED WITH A DRAFT DRIVER
PRINTER 2 LOADED WITH A WORD PROCESSING DRIVER

A>
```

1.6.2 System Division

CP/M is logically divided into four distinct areas, the lower portion of memory is reserved for system information: the Basic I/O System (BIOS), the Basic Disk Operating System (BDOS), the Console Command Processor (CCP), and the Transient Program Area (TPA).
The **CCP** provides the interface between the user's console and the remainder of the CP/M system. The user interacts with CP/M primarily through the CCP, which reads and interprets commands entered through the console. 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 CCP will also close and release any open user files when necessary. It is a distinct program which uses the BDOS and BIOS to provide a human-oriented interface to the information which is cataloged on the backup storage device.

In general, the CCP addresses one of several drives which are on-line (the standard system addresses up to four different disk drives). These disk drives are labeled A, B, C, and D. A drive is 'logged-in' if the CCP is currently addressing that drive. In order to clearly indicate which drive is the currently logged drive, the CCP always prompts the operator with the drive name followed by the symbol '>' (e.g., 'A>') indicating that the CCP is ready for another command. Upon initial start up, the CP/M system is brought in from Drive A and will display the following message:

VECTOR 4 CP/M VERSION 1.0

Following the system sign-on, CP/M automatically logs into Drive A, prompts the user with the 'A>' prompt (indicating that CP/M is currently addressing Drive A), and waits for a command. The commands are implemented at two levels: resident commands and transient commands. Resident commands are a part of the CCP program itself, while transient commands are loaded into memory (the TPA) from disk and executed.

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. Any particular file may contain any number of records, not exceeding the size of any single disk. The BDOS has entry points which include the following primitive operations which can be programmatically accessed:

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 **BIOS** provides the primitive operations necessary to access the disk drives and to interface with standard peripherals (CRTs, Serial Terminals, and Printers). The BIOS is a hardware-dependent module which defines the exact low-level interface to a particular computer system which is necessary for peripheral device I/O; it also provides the user the interface to the multitasking environment within VECTOR 4 CP/M.

The **TPA**, the last segment of CP/M, is the area called the Transient Program Area. The TPA is an area of memory (i.e., the portion which is not used by the BDOS, BIOS, and CCP) where various non-resident operating system commands and user programs are executed. The TPA holds programs which 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. The user communicates with the CCP by entering command lines following each prompt. Each command line takes one of the following forms:

Command
Command file1
Command file1 file2
.Command file1 file2

where 'Command' is either a resident function, such as DIR or TYPE, or the filename of a transient command or program, such as PIP or CONFIG. If the command is a resident function of CP/M, it is executed immediately; otherwise, the CCP searches the currently addressed disk for a file by the name '<Command>.COM'. If the file is found, it is assumed to be a memory image of a program which executes in the TPA. The CCP loads the COM file from the disk into memory. The transient program receives control from the CCP and begins execution; the program is 'called' from the CCP and, therefore, will simply return to the CCP upon completion of its processing.

If the user is in a user area other than User 0, but the program being called is on User Ø, the operating system will pull the program from User Ø into the current user area and begin execution. Data files created or edited within the current user area will be saved on that user area. If, however, the user wants to work within User Ø and is currently in another user area, the 'dot' command can be used to work within User Ø, saving data files within User Ø, while actually residing in the current user area. For example:

A>USER 5 Change from User 0 to User 5

A>DIR Call the directory of Drive A, User 5

A>.DIR Call the directory of Drive A, User Ø

This facility allows the user to work within different user areas and, depending on where the data files are to reside, store files in either area, without ever leaving the current user area (as in the example above, User 5).

It should be reiterated that the VECTOR 4 CP/M operating system is partitioned into distinct modules, including the BIOS portion which defines the hardware environment in which CP/M is executing. Therefore, the system can be easily updated to any standard environment.

Several CP/M modules and utilities have improvements that correspond to the enhanced file system. STAT and PIP both account for file attributes and user areas, while the CCP provides a 'log-in' function to change from one user area to another. The CCP also formats directory displays in a more convenient manner and accounts for both CRT and hard-copy devices in its enhanced line editing functions.

1.6.3 File Description

Nearly all of the commands reference a particular file or group of files. 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 an 'unambiguous filename' (ufn) or an 'ambiguous filename' (afn). An unamabiguous file reference uniquely identifies a single file, while an ambiguous file reference may be satisfied by a number of different files, when using '?' or '*' either in the filename or the extension.

File references consist of two parts: the filename and the extension. Although the extension is optional, it usually is generic; that is, the extension '.ASM' is used to denote that the file is an assembly language source file, while the filename distinguishes each particular source file. The two names are separated by a period (.); e.g., ffffffff.eee where 'f' represents the filename of up to eight characters, and 'e' represents the extension consisting of three characters.

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 symbols '?' and '*' may be interspersed throughout the filename and extension. In various commands throughout VECTOR 4 CP/M, the '?' matches any character of a filename in the '?' position, and the asterisk '*' matches any characters of a filename from that particular position on in the field; e.g., C*.COM would be equivalent to C???????.COM. Therefore, the ambiguous reference 'AB*.C?M' is satisified by the unambiguous filenames 'ABCD.COM' and 'ABCDEFGH.CAM'. Note that the ambiguous reference '*.*' is equivalent to the ambiguous file reference '?????????.???', while 'ffffffff.*' is equivalent to 'ffffffff.??? and '*.eee' is equivalent to '?????????.eee'. However, 'ABC*.ASM' is equivalent to 'ABC?????.ASM' where 'ABC?.ASM' is equivalent to 'ABC?####.ASM', where '#' = a space, not a character.
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 exact 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

As an added convenience, the user can generally specify the disk drive name along with the filename. In this case, the drive name is given as a letter (A through Z) followed by a colon (:); e.g., A>B:<filename.ext>. The specified drive is then 'logged-in' before the file operation occurs. Thus, the following are valid filenames with disk name prefixes:

A:W.Y B:XYZ C:GAMMA Z:XYZ.COM B:X.A?M D:*.ASM

It should also be noted that all alphabetic lower case letters for drives and filenames are always translated to upper case when they are processed by the CCP.

Since it is possible for the user to switch the currently logged disk by entering the disk drive name (A, B, C, or D) followed by a colon (:) when the CCP is waiting for console input, the following are possible examples in a sequence of prompts and commands:

A>DIR (Lists all files on Drive A)

SAMPLE ASM

SAMPLE PRN

A>B: (Switches to Drive B)

B>DIR *.ASM (Lists all '.ASM' files on Drive B)

DUMP ASM

SAMPLE1 ASM

TEST3 ASM

B>A: (Switches back to Drive A)

A>DIR B: (Lists all files on Drive B)

1.6.4 File Construction

CP/M files are constructed from basic building blocks called 'records'. A record is a 128-byte block of data containing either program object code, program data, or text. A file is, therefore, simply a number of these records which are grouped together and given a name. Since a file can have a very large number of these records in it (up to 65,536 records in each file), CP/M must impose a higher level of internal structure to manage all of the records in a file. This is achieved by using a logical data block called an 'extent' consisting of 128 records (16,384 bytes of data). Therefore, any CP/M file is comprised of at least one or more extents; these are used by CP/M to keep track of where all the records of a file are located on a disk.

Since there are many sectors on a disk, CP/M could store a 128-byte record in any number of places on the disk. In order to keep track of where these sectors are, the diskette is broken up into 'allocation blocks', the size depending on the disk configuration (may be from 1K to 16K in length). Then, if a file needs more space to write more records to, a full new allocation block is assigned to it even though only 128 bytes (1 record) may actually be used. Each directory entry can have 8 or 16 allocation blocks assigned to it. The number of each allocation block assigned to a directory entry is stored in one or two bytes within field 7 of the File Control Block (FCB). See Sections 4.2 and 4.3 for a complete description of FCB and allocation blocks.

CP/M keeps a master list in memory of all of the assigned allocation blocks currently assigned to a file along with all the unused allocation blocks. Whenever a new allocation unit is required by a file, CP/M will take the next available allocation unit, delete it from the unused list, and add it to the FCB of the file. These lists are read into the BDOS from the diskette the first time the diskette is accessed after a warm boot. By assigning new allocation blocks to files as they get larger, and reclaiming allocation units as files get smaller or are erased altogether, CP/M can dynamically manage all of the file space on a diskette, and a file's data can be spread across the disk in randomly located sectors.

The final step used by CP/M to keep track of the files on a diskette is the directory. CP/M stores the directory information in the beginning of the data storage area of the diskette. A directory entry may contain multiple extents. This depends on allocation block size and the total number of blocks a drive can accomodate. Therefore, a file with more than one extent may have multiple directory entries. Hence, although a 'DIR' command is entered, only the first directory entry for each file is displayed. The format of the directory entry is the same as the FCB except for the following fields:

Field 1: User number 0-xx. E5 means file is deleted
Fields 9-11: e1', e2', e3' are set according to file type.
Field 12: Gives maximum extent number.
Field 13: Mod number equals Ø.
Fields 32-35: Not used.

Therefore, whenever a file is accessed through CP/M, its FCB is brought into the TPA so that it can be accessed quickly and conveniently. Any changes or updates to a file by the user will automatically be updated to the FCB by CP/M.

Random access facilities are present in CP/M which allow immediate reference to any record of an eight megabyte file. Using CP/M's unique data organization, data blocks are only allocated when actually required and movement to a record position requires little search time. Sequential file access is upward compatible from earlier versions to the full eight megabytes, while random access compatibility stops at 512 Kbyte files. Due to CP/M's simpler and faster random access, application programmers are encouraged to alter their program to take full advantage of the facilities.

SECTION II - RESIDENT COMMANDS

2.1 INTRODUCTION

VECTOR 4 CP/M will execute two types of commands: resident and transient. Resident commands are those commands whose instructions are permanently resident in the CCP and may be used at any time. Transient commands are those commands whose programs are stored on a disk, are then loaded into the TPA from the disk and executed. CP/M provides the user with several resident commands which may be interpreted and immediately executed by the CCP when invoked by the user.

2.2 CCP LINE EDITING AND OUTPUT CONTROL

The CCP allows certain line editing and output functions while entering command lines.

A. Line Editing

[BACKSPACE] or
[DEL] Deletes the last character entered at the terminal.

[CTRL U] Ignores the entire line entered at the terminal, keyboard entry continues on a clean command line.

[CTRL X] Deletes the entire line enterered at the terminal, keybaord entry continues on same line.

[CTRL R] Redisplays current command line, keyboard entry continues at the end of the redisplayed command line.

[CTRL C] CP/M system reboot (warm start).

[CTRL D] Clears the screen before executing any CCP command; must be entered before the command.

[CTRL Z] Ends input from the terminal (used in PIP).

B. OUTPUT CONTROL

[CTRL P] Copies all subsequent console output to the currently assigned list device (see the STAT command). Output is sent to both the list device and the terminal until the next [CTRL P] is entered.

[CTRL K] Sends a form feed to the currently assigned list device; if no device is assigned, the [CTRL K] will be ignored.

[CTRL S] Stops the terminal output temporarily; program execution and output continues when any key, except [CTRL C], is entered at the terminal. This feature is used to stop output on high speed terminals, such as CRTs, in order to view a segment of output before continuing.

Note that the [CTRL]-key sequences are executed by depressing the Control ([CTRL]) key and the desired letter key simultaneously. Further, CCP command lines can generally consist of up to 127 characters in length; they are not acted upon until a [RETURN] is entered.

2.3 RESIDENT COMMANDS

Resident commands are a part of the CCP program itself, while transient commands are loaded into the TPA from disk and executed. The resident commands consist of the following:

ERA Erases specified files.

DIR Lists file names in the directory.

REN Renames the specified file.

SAVE Saves memory contents in a file.

TYPE Types or displays the contents of a file to the terminal.

TPA Calls address 1ØØH in user's memory.

BSPL	Begins Spooler.
ESPL	Ends Spooler.
DSPL	Activates Despooler to print a specified spooled file.
KSPL	Aborts the Despooler.
BYE	Performs a cold boot process.

The following are Primitive Resident Commands. Originally they were available through the use of the system monitor program. They are now accessible through the CCP.

FILL	Fills memory with constant
FIND	Finds a code sequence in memory.
DISP	Displays memory in hex and ASCII.
GOTO	Starts execution at given address.
PROG	Programs memory.
INP	Inputs from CPU port.
OUT	Outputs to port.
COMP	Compares two blocks of memory.
MOVE	Moves a block of memory.
CHEK	Computes checksum of memory.
XCOM	Mindless Terminal Emulator/loop back tester.

The file and device references described before can now be used to fully specify the structure or format of the resident commands.

2.3.1 ERA Command

The ERA (erase) command will remove files from the currently logged disk or another specified disk. The files which are erased are those which satisfy the ambiguous filename (afn). The following examples illustrate the use of the ERA command:

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 a filename 'X' are removed from the current disk.

ERA *.ASM All files with the extension of '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 erased.

ERA *.* All files on the current disk are erased (in this case, the CCP prompts the user with the message 'ALL FILES (Y/N)?' which requires a 'Y' response before files are actually erased).

ERA B:*.PRN All files on Drive B which have an extension of '.PRN' are erased, independently of the currently logged disk.

2.3.2 DIR Command

The DIR (directory) command will cause the names of all files on the currently logged disk which satisfy the ambiguous filename (afn) '*.*' to be listed at the terminal (the command 'DIR' is equivalent to the command 'DIR *.*'). Following is a list of valid DIR commands:

DIR X.Y Lists the unambiguous file 'X.Y'.

DIR X?Z.C?M Lists all ambiguous files matching the specified reference.

DIR *.Y Lists all ambiguous files with an extension of '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: Lists all files on Drive B.

DIR B:X.Y Lists the file 'X.Y' on Drive B.

DIR B:*.A?M Lists all ambiguous files matching the specified reference on Drive B.

If no files can be found on the selected disk which satisfy the directory request, the message **'NOT FOUND'** is displayed on the terminal.

2.3.3 REN Command

The REN (rename) command will allow the user to change the names of files on disk. The command format is **'REN unf1=unf2'** where unf1 is the name of the new file and unf2 is the name of the old file. Following are examples of the REN command:

REN X.Y=Q.R Renames the file 'Q.R' to 'X.Y'.

REN Y.COM=Y.MEM Renames the file 'Y.MEM' to 'Y.COM'.

The user can precede either ufn1 or ufn2 (or both) by an optional drive name. When one of the files is preceded by a drive name, then the other is assumed to exist on the same drive. If both files are preceded by drive names, the same drive must be specified in both cases. Following are examples of the REN command:

REN A:X.ASM=Y.ASM Renames the file 'Y.ASM' to 'X.ASM' on Drive A.

REN B:X.BAS=Z.BAS Renames the file 'X.BAS' to 'Z.BAS' on Drive B.

REN A:A.ASM=B:A.BAK Renames the file 'A.BAK' on Drive B to 'A.ASM' on Drive A.

If the ufn1 already exists, the message **'FILE EXISTS'** will display and the change will not occur. If ufn2 does not exist on the specified disk, the message **'NOT FOUND'** will display on the terminal.

2.3.4 SAVE Command

The SAVE command places 'n' pages (256-byte blocks) onto disk from the TPA and names this file with a specified ufn. In the CP/M distribution system, the TPA starts at 1ØØH (hexadecimal), which is the second page of memory; therefore, if the user's program occupies the area from 1ØØH through 2FFH, the SAVE command must specify two (2) pages of memory. The machine code file can be subsequently loaded and executed. The SAVE command can also specify a disk drive in the afn portion of the command. Following are examples of the SAVE command:

SAVE 3 X.COM	Copies 1ØØH through 3FFH to a file named 'X.COM'.
SAVE 4Ø C	Copies 1ØØH through 28FFH to a file named 'C' (note that 28 is the page count in 28FFH and that 28H = 2x16+8 = 4Ø decimal).
SAVE 4 X.Y	Copies 1ØØH through 4FFH to a file named 'X.Y'.
SAVE 1Ø B:ZOT.COM	Copies 10 pages (1ØØH through ØAFFH) to a file named 'ZOT.COM' on Drive B.

2.3.5 TYPE Command

The TYPE command displays the contents of a specified ASCII source file (ufn) on the currently logged disk at the terminal. This command expands tabs ([CTRL L] characters) assuming tab positions are set at every eighth column. The command can also reference a drive name before the ufn. Following are examples of the TYPE command:

TYPE XYZ.MEM
TYPE ABC.EPL > Types the specified file on the terminal.
TYPE XXX.ASM

TYPE B:X.PRN Types the specified file on the terminal from Drive B.

2.3.6 TPA Command

The TPA (Transient Program Area) command will 'jump' the user to address 1ØØH in the user's memory and execute the program at that location; i.e., to go back into memory if, for instance, the user forgot to save the current file or unintentionally exited from a program. The command line format consists of either of the following:

TPA

or

TPA drive:filename

WARNING: Not all programs may be re-entered successfully after exiting; consult your program User's Manual or Manufacturer for more information.

2.3.7 BSPL Command

The BSPL command will begin the spooler function and send all subsequent printer characters to a specified file. Following is the command format:

BSPL drive:filename (must be ufn)

NOTE: Characters sent to printer via the BIOS will not be spooled; only characters sent through BDOS function 5 will be spooled (most programs use function 5 for printer output). BSPL will detach any attached list device before spooling.

After the command is entered, one of the following messages will display on the terminal giving the status of the spooler:

SPOOLER ACTIVATED Spooler has been successfully activated.

SPOOLER ALREADY ACTIVE Spooler has already been activated and is ready.

FILE EXISTS	Spooler is refusing to work with an existing spooled file.
DIR FULL	No room left in directory.
ILLEGAL FILENAME	Filename specified either does not exist or has illegal characters in the entry.
d:DRIVE SELECT	Drive specified is bad, where 'd:' is the specified drive in the command line.

2.3.8 ESPL Command

The ESPL command will close the spooler file and deactivate the spooler. The spooler will not close after a warm boot, it will remain open; therefore, this command must be executed when the user no longer wants the printer information to go to the spooler file. After the command is entered, one of the following messages will display on the terminal giving the status of the end-spooler attempt:

SPOOLER CLOSED	Spooler has been closed successfully.
SPOOLER NOT ACTIVE	Spooler has already been deactivated, or if the spooler gets an I/O error of some sort.
BAD WRITE	Spooler encounters a disk I/O error when trying to write the last sector.
CLOSE ERROR	Spooler encounters a problem writing to the directory or trying to spool to a deleted file.

2.3.9 DSPL Command

The DSPL command will despool a specified file to a specific print device. The format of the command line is as follows:

DSPL n< drive:filename

where 'n' is the print device (either '1' or '2'), '<' will optionally turn on the 'auto-paging' function, and the filename must be unambiguous. The despooler runs in 'background', i.e., the despooler may be printing to the list device and the user can continue with some other task, such as editing another file.

After the command is entered, one of the following messages will display on the terminal giving the status of the despooler attempt:

DESPOOL ACTIVATED	Despooler has been activated successfully.
DESPOOL ALREADY ACTIVE	Despooler has been asked to work on more than one file at a time.
NO FILE	Despooler cannot find the specified file on that disk, or the file has not yet been spooled.
BAD FILE NAME	File is either blank or has an afn specification; file must be an ufn.
INVALID PRINT DEVICE	Printer number is invalid, or the printer is busy.
d:DRIVE SELECT	Drive specified is bad, where 'd:' is the specified drive in the command line.
PRINTER BUSY	Desired list device is in use by another task (user or despooler).

2.3.10 KSPL Command

The KSPL command will abort, or prematurely stop, the operation of the despooler; the despooler will turn off normally once the specified file has been printed. After the command is entered, one of the following messages will display on the terminal giving the status of the abort attempt:

DESPOOL ABORTED	Despooler has been aborted successfully.
DESPOOL NOT ACTIVE	Despooler is not active so cannot be aborted.

NOTE: The Despooler will not expand tabs or initialize pagination control; it will send characters to the printer exactly as they are in the file.

2.3.11 BYE Command

The BYE command will perform a cold boot procedure executing functions beyond a warm boot ([CTRL C]) procedure. Following is a list of both warm boot and cold boot functions performed by the BYE command:

Warm Boot functions:

- Release File and Record Resources (function 255)
- Reset Keyboard buffering and conversions (function 241, 242)
- Set Error mode to default (function 226 or 45)
- Execute warm boot Auto-Command

Cold Boot functions provided by BYE command:

- Reset BDOS base (function 246)
- Clear Cross-Bank calls (function 245)
- Clear Auto-Paging (function 243)
- Release (detach) List Device (function 159)
- Execute Cold Boot Auto-Command
- Set Default Key Conversion (function 219)
- Enable [CTRL C] Warm Boot (function 217)

2.3.12 Primitive Resident Commands

These commands are functionally similar to the commands used by Vector Graphic system monitor programs. They have been added to the resident command structure within the CCP. The command line format is given for each command; however, if only the command is given with nothing following, prompt(s) for the missing information will be given.

A. FILL Command

The FILL command will place a single byte instruction or datum in the memory block between the start and end address. This can also be useful for setting memory to zero to make the end of a file or assembled program to stand out more clearly. For test purposes, single instructions can be executed continuously so that bus waveforms can more easily be interpreted; this is done by filling a block of memory with a repeated instruction sequence with a jump to the start of the block enabling the program to loop continuously.* The format of the command line consists of the following:

FILL <Start Address (4 hex)> <End Address (4 hex)> <Data (2 hex)>

NOTE: An error will result if a FILL to "common RAM" (FC00H-FFFFH) is attempted.

B. FIND Command

The FIND command will search for a particular code sequence of up to 63.5 bytes in length in the range between the start and end address. This command is useful for locating a code sequence of jump address, text strings, etc. For example, if you want to change a control character (e.g., [CTRL C]) in a program, you can try 'FE Ø3', which is 'CPI Ø3' since this is a common way of testing input characters. If you want to find all locations that call or jump to a particular address, C7ØØH for example, simply search for 'ØØC7'; however, there is no guarantee that each location displayed is valid object code—it may be part of a data table, ASCII string, or second and third bytes of a three-byte instruction. Wildcards are available with the FIND command and consist of '?' and 'X', both equal to one nyble. The format of the command line consists of the following:

FIND <Start Address (4 hex)> <End Address (4 hex)> <Sequence (X(2 hex))>

C. DISP Command

The DISP command will display the memory contents between the start and end address as pairs of hexadecimal characters. The left character in each pair represents the four most significant bits of the memory location. The ASCII representation will also be displayed on the right. The display may be halted or interrupted by pressing [CTRL S] and continued by pressing any key except [CTRL C]. Pressing [CTRL C] will abort the display or entry. The format of the command line consists of the following:

DISP <Start Address (4 hex)> <End Address (4 hex)>

D. GOTO Command

The GOTO command will cause a call to the specified start address to execute a program or user subroutine. The address contained on the stack is the CCP RETCOM, and if the user routine at the address ends in 'RET', program execution will return to the CCP; pushing more registers on the stack than are popped will defeat the return feature with undesirable effects. The format of the command line consists of the following:

GOTO <Start Address (4 hex)>

E. PROG Command

The PROG command will display the content of the byte of memory in the specified start address in hex followed by a hyphen. The user can keep the current value and advance to the next byte by entering a [RETURN], or replace the existing contents by entering the appropriate hex byte. The current values for the memory segment will be displayed, and new values entered will replace the old values. A [RETURN] will cause the current memory location to be unaltered, while a [CTRL C] will exit and save the contents. The format of the command line consists of the following:

PROG <Start Address (4 hex)>

F. INP Command

The INP command will cause the CPU to display the contents of a specified port (2-byte hex code). This command is useful in the checking of peripheral equipment. Those ports used by the terminal, cassette interface, etc. will contain a usable value; all others will read 'FF' since the data bus will be floating when the command is executed. The format of the command line consists of the following:

INP <Port (2 hex)>

G. OUT Command

The OUT command will output a two-byte hex value or data to a specified port. This command is useful for checking out peripheral equipment; e.g., if a printer is connected to I/O port 6, the command line '**OUT 06 41**' will cause an 'A' to be printed ('41' = the hex code for the ASCII letter 'A'). The format of the command line consists of the following:

OUT <Port (2 hex)> <Value (2 hex)>

H. COMP Command

The COMP command will make a byte-by-byte comparison of the block of memory data between the source start and end address and a block of identical length starting at the destination start address. Only the differences will be printed out with the address and the byte of the first block and the address and byte of the second block. This command is useful to compare two versions of a program or to verify that PROMs have been programmed correctly. The format of the command line consists of the following:

COMP <SBSA> <SBEA> <DBSA>

where:

SBSA = Source Block Start Address (4 hex)
SBEA = Source Block End Address (4 hex)
DBSA = Destination Block Start Address (4 hex)

I. MOVE Command

The MOVE command will move the data contained in memory between the source start and end address to the memory location starting at the destination start address. This command is useful for moving a program from a temporary storage location to its correct address. This command can be useful for bank programming of PROMs, or for creating repeating instruction sequences for test purposes. The format of the command line consists of the following:

MOVE <SBSA> <SBEA> <DBSA>

where:

SBSA = Source Block Start Address (4 hex)
SBEA = Source Block End Address (4 hex)
DBSA = Destination Block Start Address (4 hex)

J. CHEK Command

The CHEK command will compute and display the MOD 256 checksum of the memory contents within the specified start and end address range. This command is useful for checking proms or files to see if anything has changed. Any source file or program area of pure code (it does not write on itself) will have the same checksum as when it was loaded. While debugging assembly language programs, it is useful to be able to verify that a program being debugged has not written garbage in the source file or assembler.

The format of the command line consists of the following:

CHEK <Start Address (4 hex)> <End Address (4 hex)>

K. XCOM Command

The XCOM Command will output anything typed at the keyboard through the serial port specified. Anything received from this serial port will be displayed on the screen.

XCOM <port> Port = 4 or 6 (4 is default value)

SECTION III - TRANSIENT COMMANDS

3.1 INTRODUCTION

Transient commands are programs, sometimes referred to as command files. A transient command, or program, can be executed by simply entering the filename, the .COM extension does not need to be entered in the command line; e.g., PIP.COM is a transient command and can be executed by entering 'PIP' after the 'A>' prompt. Transient commands are loaded from the currently logged disk and executed in the TPA.

There are three logical divisions of transient commands: the **Utility** commands, the **Programming** commands, and the **Test** commands. This division of commands is initiated in this manual for ease of use and understanding. The transient commands defined for execution under the CCP are shown below, while additional functions can easily be defined by the user (see the LOAD command definition).

Utility Commands

GENSYS Copies a VECTOR 4 CP/M operating system onto either initialized double-sided diskettes or a hard disk.

CONFIG Creates a personalized system by setting up CP/M for the printer, enabling automatic page feed, creating an 'auto boot' command, and setting logical drive assignments, printer driver selection, boot options, and other assorted user options.

SORTDIR Reads the directory from a specified disk, sorts the directory alphabetically, and writes the sorted directory back to the disk.

EDIR Lists the names of all files in an alphabetically sorted directory (other options for a directory listing are also available) to the terminal, ending with the remaining space left on disk.

ERAX Erases, or deletes, specified files from the currently logged disk; ERAX has several options available for file deletion.

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.

PIP Copies programs and files between disks and user areas, and also transfers files to or from logical devices, such as a printer or communication lines.

FORMAT Initializes a disk or diskette in order to 'save' or 'create' new files on the diskette; every new diskette must be 'formatted' before use, except for backup procedures using the DISKCOPY utility.

DISKCOPY Creates a copy or backup of single and double sided diskettes.

STORE Creates a backup of a hard disk with files exceeding the capacity of one floppy diskette but consisting of 5 Mbytes or less.

RESTORE Retrieves the backup files created by STORE and returns them to the hard disk.

Programming Commands

SCOPE (SC) Accesses Vector's SCreen Oriented Program Editor, a program editor for Vector computers, to create and edit any CP/M ASCII file, either program text files or data files, and offers horizontal scrolling, large file size (limited only to the size of the disk), full control character display, and elimination of non-printing characters in the text. See Vector's SCOPE REFERENCE MANUAL for detailed information on using SCOPE.

ZSM Loads the CP/M assembler and assembles a Z-80 assembly language text (source code) into a sequence of machine language instructions (object code) which can then be loaded into memory and executed. See the ZSM ASSEMBLER USER'S MANUAL for more detailed information.

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).

SUBMIT Submits a file of commands for batch processing; also allows chained batch commands, since the last command in a SUB file can initiate another SUB file.

XSUB Extends the power of the SUBMIT facility to include line inputto programs, as well as the CCP, and, if needed, is included as the first command in a SUB file.

DDT Loads the CP/M debugger into the TPA and begins execution. Allows user dynamic interactive testing and debugging of programs generated in the CP/M environment. See the CP/M DYNAMIC DEBUGGING TOOL (DDT) USER'S GUIDE for more detailed information.

RAID Allows access to Vector's RApid Interactive Debugger program which simulates a program step by step taking each instruction of the program, executing it, and displaying the result of the operation. See Vector's RAID - RAPID INTERACTIVE DEBUGGER, REFERENCE MANUAL for more detailed information.

DUMP Allows access at the console to the contents of a disk file in hexadecimal form, listing sixteen bytes at a time, along with a relative offset address.

GENLIST Creates the list device files (.LDn) for VECTOR 4 CP/M by converting a hex file to a '.LD1' or '.LD2' file for use by the system.

Test Commands

DISKTEST Checks and determines whether a system failure is due to a hardware or software problem; also used for obtaining required patterns for alignment of the floppy drive.

MEMTEST Checks common memory and each bank of memory listing bank addresses and status.

RECLAIM Attempts to recover a file with a bad sector and remaps the disk to prevent other programs from using the bad sector.

CRC Runs a Cyclic Redundancy Check (CRC) on a specified program and produces a CRC value, which can be checked against the System disk listing for program comparisons.

CPUTEST	Reads and writes address 1AAH with AAH.
SCNTEST	Used for adjusting focus of CRT.
PORTEST	Checks special port functions.
KYBDTEST	Tests all physical keyboard codes and prints them.
PRINTEST	Checks functioning of printers.

Transient commands are specified in the same manner as resident commands; additional commands can be easily defined by the user. As an added 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 'A>B:STAT' causes CP/M to temporarily 'log' into Drive B for the source of the STAT transient, and then return to the original logged disk for subsequent processing.

3.2 UTILITY COMMANDS

3.2.1 GENSYS Command

The GENSYS transient command allows generation of an VECTOR 4 CP/M operating system onto an initialized double-sided diskette or an initialized hard disk. The program will automatically determine the type of system that exists and will prompt for one of the two possible systems to generate. GENSYS will then prompt for information pertaining to the possible available options for the specific type of system; these options can only be altered by the program. If it can't determine the destination disk from the information provided, it will then prompt for the destination drive. The program places a copy of the VECTOR 4 CP/M onto the system tracks of the destination drive and will then prompt the user whether to copy the following files:

VECTOR.CPM	**Operating System file**
VECTOR.LD1	**List Device 1 (Serial driver) file**
VECTOR.LD2	**List Device 2 (Parallel driver) file**
VECTOR.CMD	**Auto-Boot Command file**

GENSYS.COM GENSYS program

When running GENSYS from a hard disk system, and the destination drive is also a hard disk, the VECTOR files will not be copied.

A. FLOPPY DISK ONLY SYSTEM

VECTOR 4 CP/M SYSTEM GENERATION UTILITY
VERSION 1.0

SELECT THE DESTINATION DRIVE [A], [B], [C], [D]:

Enter 'A', 'B', 'C', or 'D' as the final destination of the CP/M operating system currently being generated; the drives shown in the prompt will be the currently available double-sided floppy drives within the user's system; no single-sided drives will display because the system may only be written to a doubled-sided diskette.

INSERT DESTINATION DISKETTE AND PRESS THE [RETURN] KEY WHEN READY:

Insert the diskette that will be the destination of the CP/M system and enter a [RETURN] when ready.

ERROR - "VECTOR.CPM" FILE NOT FOUND ON THE SOURCE DRIVE

This error message will display if no VECTOR.CPM file is found on the source disk; the result being no operating system being generated.

DO YOU WANT THE VECTOR SYSTEM AND LIST DEVICE FILES COPIED? [Y] OR [N]:

If the destination diskette is new or has never had a system placed on it, it will not have these files. Enter 'Y' to copy the Vector System and List Device files onto the destination diskette; otherwise, enter 'N'. Entering a 'N' response will still copy the operating system onto the destination diskette even though it will not copy the List Device files. All of these files, when copied, will remain in System status and, therefore, will not appear in the directory.

INSERT SOURCE DISKETTE AND PRESS THE [RETURN] KEY WHEN READY:

This prompt and the following will display only if the Vector System and List Device files are to be copied. Insert the diskette that has CP/M on it and enter a [RETURN] when ready.

INSERT DESTINATION DISKETTE AND PRESS THE [RETURN] KEY WHEN READY:

Again, insert the destination diskette and enter a [RETURN] when ready.

VECTOR.CPM FILE COPIED TO DESTINATION DRIVE
VECTOR.LD1 FILE COPIED TO DESTINATION DRIVE
VECTOR.LD2 FILE COPIED TO DESTINATION DRIVE
VECTOR.CMD FILE COPIED TO DESTINATION DRIVE
GENSYS.COM FILE COPIED TO DESTINATION DRIVE

This message will display after the files have been copied to the destination disk successfully. The VECTOR.CMD file will only be found and copied if the user has created an auto command, using the CONFIG A option.

SYSTEM GENERATION COMPLETED

B. FLOPPY AND/OR HARD DISK SYSTEMS

VECTOR 4 CP/M SYSTEM GENERATION UTILITY
VERSION 1.0

[1] GENERATE FLOPPY ONLY SYSTEM
[2] GENERATE 5 MBYTE WINCHESTER SYSTEM

SELECT ONE OF THE ABOVE:

This prompt will display if the system has either a Floppy or a 5 Mbyte hard disk system. If you want to generate a CP/M operating system on a floppy, enter '1' and follow the prompts for the Floppy Only System as shown above in A.; if you want to generate an operating system on a hard disk, enter '2', for a 5 Mbyte system.

SELECT THE NUMBER OF LOGICAL SURFACES [1] OR [2]:

This message will display for a 5 Mbyte system only; enter '1' if the hard disk is to be just one logical drive (Drive A) or '2' if it is to be two logical drives (Drives A and B).

[1] 256 ENTRIES
[2] 512 ENTRIES
[3] 768 ENTRIES
[4] 1024 ENTRIES

SELECT THE DIRECTORY SIZE FROM ABOVE:

If a small number of large files are expected to be used, select a smaller directory size. With a smaller directory, the system will run marginally faster because the program will have to read fewer entries. If a large number of smaller files are expected to be used, select a larger directory size.

CAUTION: When installing an operating system on more than one drive (within a hard disk system), the same directory size for all drives must be selected in order to avoid conflicts.

SYSTEM GENERATION COMPLETED

Upon completion of a successful system generation, the new disk or diskette will contain the operating system; a system reboot of the surface having just received the new operating system is automatic. If the surface is blank or has just been formatted for some reason, only the operating system will now reside on it; however, only the resident commands are available. A freshly formatted diskette appears to CP/M as a diskette with an empty directory; therefore, the user must copy the appropriate COM files from an existing CP/M diskette to the newly constructed diskette using the PIP command.

The user can copy all files from an existing diskette to another diskette or to a hard disk by entering the PIP command

PIP B:=A:*.*[V]

which copies all files from Drive A to Drive B and verifies that each file has been copied correctly. The name of each file will be displayed at the terminal as the copy operation proceeds. Drives A and B can be switched in the PIP command line if the user is copying files onto the hard disk with the floppy (B or C) as the source; e.g.:

PIP A:=B:*.*[V]

It should be noted that a GENSYS does not destroy the files which already exist on a diskette; it results only in construction of a new operating system. Further, a system must be put on any floppy that will be used in booting the system.

3.2.2 CONFIG Command

The CONFIG command allows the user to create a personalized system by setting up various options within VECTOR 4 CP/M for the printer, to enable automatic page feed, to create an 'auto-boot' command (either temporarily, or to be saved on disk), and to set drive configurations, printer driver selection, and boot options. All options have initially been set and will display the default settings within parens when running any of the CONFIG options; however, these may be changed at any time according to the user's needs. Prompts for the different options can be changed by entering a letter, number, or command line, as in the case of 'auto-boot; to keep the present values, which will be displayed on screen, simply enter a [RETURN]. The options (F, D, S, P, A, O, C, and B) can be entered individually or in groups of options, as needed. All CONFIG commands will exit via a warm boot except CONFIG F, which exits via a cold boot. The command line should be in the following format:

CONFIG <option(s)>

e.g., **CONFIG FDSPAOCB** (no space or comma between options) or any combination of these options.

CONFIG

When 'CONFIG' is entered without any options added, it will default to the 'P' (printer) option and attach the list device temporarily. Faster selection of the print options can be achieved by specifying that option within the command line, e.g.:

CONFIG 1

CONFIG will respond with one of the following messages:

OPTION SELECTED
or
PRINTER IS BUSY

This feature will allow easier use of the configuration program for printer selection, and the ability to select a printer with either a cold or warm auto-boot command by using 'CONFIG x' as the auto-boot command, where 'x' is equivalent to one of the following:

0 = Discard Printout
1 = List Device 1
2 = List Device 2
3 = Route Printer Output to System Console

CONFIG F

This option will allow the user to select the configuration of the floppy disk drives. Each drive may be individually configured to either single or double sided. The current status of each drive is listed along with the prompt for that drive.

FLOPPY DISK CONFIGURATION (SINGLE/DOUBLE)

SET DRIVE d: TO [S] SINGLE OR [D] DOUBLE SIDED - (D):
SET DRIVE d: TO [S] SINGLE OR [D] DOUBLE SIDED - (D):
SET DRIVE d: TO [S] SINGLE OR [D] DOUBLE SIDED - (D):

The current assignment will display for each drive. The floppy disk configurations are initially set as double sided. To change the configuration to single sided, enter 'S' when prompted for that drive; otherwise, enter a [RETURN] for each prompt to keep the present drive configuration.

MAKE SELECTION PERMANENT [Y]-[N]?

Enter 'Y' to make the configuration changes permanent; otherwise, enter 'N' for a temporary (or in-memory) change only. CONFIG F will exit via a cold boot.

CONFIG D

This option will allow the user to alter the disk configuration options. However, its use and consequences should be fully understood. As you are prompted for each option, its current status will be displayed with the prompt.

DISK ACCESS OPTIONS FOR DRIVE d: - ENTER [E] ENABLE OR [D] DISABLE

READ BEFORE WRITE (ENABLED):
READ AFTER WRITE (ENABLED):
WRITE PROTECT DETECT (ENABLED):
READ ONLY (DISABLED):

The 'READ BEFORE WRITE' and 'READ AFTER WRITE' should be enabled to ensure error correction; however, both can be disabled to allow a slightly faster system. When disabled, however, errors may not be detected.

The 'WRITE PROTECT DETECT' should be kept enabled if the write-protect tabs are being used.

The 'READ ONLY' option will allow a software write-protect feature on selected disk drives.

MAKE SELECTION PERMANENT [Y]-[N]?

Enter 'Y' to make the changes permanent; otherwise, enter 'N' for a temporary (or in-memory) change only.

CONFIG S

This option will allow the user to alter the system options. It should normally not be necessary to execute this program as the options have been set for optimum system performance.

DESPOOL TAB EXPANSION (DISABLED):

Enter 'E' to enable the despool tab expansion at intervals of eight characters; enter 'D' to disable tab expansion. This option is initially set for 'DISABLED'.

DESPOOL CHARACTERS PER TIME SLICE (3):

Enter the number of characters (1 to 9) per time slice, as needed; the normal default is '3'. This is the number of characters that are sent out by the despooler during its execution interval.

CCP TO BDOS CROSS BANK CALLS (DISABLED):
BDOS TO BIOS CROSS BANK CALLS (DISABLED):

These crossbank call options should be left disabled except if a program either intercepts CCP BDOS calls or modifies the BIOS Jump Table.

RETURN CP/M VERSION NUMBER [2.5] OR [3.0] - (2.5):

This option will allow the user to alter the returned version number of CP/M through BDOS function 12, causing the record and file lock facilities to be disabled or enabled. Entering '2.5' will **disable** the facilities allowing most 2.2 version software to work (set initially to 2.5), while entering '3.0' will **enable** the record and file lock facilities.

SELECTION PERMANENT [Y]-[N]?

Enter 'Y' to make the changes permanent; otherwise, enter 'N' for a temporary (or in-memory) change only.

CONFIG P

This option will allow the user to specify the destination of the printed list output stream; the system supports two printers on-line. The system maintains two different physical printer drivers; the print output stream may be directed to either driver, the console output (the video screen), or simply discarded (one may want to discard printout when no printer is connected to the system or in cases where software testing is being performed). Generation of custom printer drivers is easily accomplished.

[0] DISCARD PRINTOUT

This option will simply discard the printout entirely.

[1] LIST DEVICE 1

See [2] below.

[2] LIST DEVICE 2

The two list device options will send the printout to one of two printers; the current assignments will be displayed for the two list devices. In the event that the despooler is using one of the two list devices, it will display a 'busy' message, ignoring any user 'select' function. Once selected, the device will remain assigned until the user selects another device or the system is turned off.

Printer driver protocols are present in the system for printers which use the following protocols:

- "Qume Type" parallel protocol (also "NEC Type" parallel protocol)
- "Centronics Type" parallel protocol.
- Standard Serial protocol.
- XON/XOFF Serial protocol.
- ETX/ACK Serial protocol.

These protocols are described in the VECTOR 4 - TECHNICAL INFORMATION hardware manual. Also refer to Section 3.3.9 and Exhibit 3-1 of this manual.

The user can choose any one of the printer drivers during the 'CONFIG P' procedure.

[3] ROUTE PRINTER OUTPUT TO SYSTEM CONSOLE

This option will send the printout to the terminal.

CONFIG A

This option will allow the user to set and clear 'auto-boot' commands. An auto-boot command is a CCP command that is executed automatically when CP/M is booted. The user has the choice of making the command execute either on a cold boot, a warm boot, or both; this will cause the system to boot directly into a given program bypassing the CP/M executive. You can only have one auto command per diskette or drive (the one from which the system is booted). This auto command is the one that is executed automatically when you boot CP/M from that diskette.

[0] CLEAR AUTO COMMAND
[1] AUTO COMMAND ON COLD BOOT
[2] AUTO COMMAND ON WARM BOOT
[3] AUTO COMMAND ON COLD AND WARM BOOT

CURRENTLY - 0 * NO COMMAND ASSIGNED *****

ENTER DESIRED SELECTION:

Enter '0' to clear any auto command already in effect; enter '1', '2', or '3' to enable an auto command, depending whether the command should activate on a cold boot, a warm boot, or both.

NOTE: It should be remembered that using the auto command on both cold and warm boot will lock the user into that particular program.

ENTER AUTO COMMAND STRING:

Enter the filename of the program and any necessary parameters that CP/M should boot into; e.g., 'MEMORITE', 'EPL', 'DIR', 'MBASIC5 MAILMENU.BAS /S:512', etc.

MAKE SELECTION PERMANENT [Y]-[N]?

Enter 'Y' to make the selected auto command permanent; otherwise, enter 'N' for a temporary change only.

PRESS [RETURN] TO CONFIRM SELECTIONS OR [ESC] TO ABORT...

Enter a [RETURN] to confirm the chosen auto command option or [ESC] to abort the program and return to CP/M.

CONFIG O

This option will allow the user to adjust the variables of a selected list device; these variables include the number of physical and printed lines per page, the number of lines skipped at the top of each page, and an auto-paging feature. The prompts, along with the current values in parens, consist of the following:

TOTAL NUMBER OF PHYSICAL LINES PER PAGE (55):
TOTAL NUMBER OF PRINTED LINES PER PAGE (50):
NUMBER OF LINES TO SKIP AT THE TOP OF EACH PAGE (5):
[E]NABLE OR [D]ISABLE AUTO-PAGING (ENABLED):

Enter the values to be changed for each variable. Note: Since the user can reassemble a printer driver to contain frequently used page settings, there is no need for the option of making these selections permanent.

CONFIG C

This option will allow the user to disable and enable certain character input; the control characters include the [CTRL K] and [CTRL P] that currently is intercepted by the BDOS and not printed. The user will have the option of making his selection permanent, if desired.

The prompt consists of the following:

ENTER [E] TO ENABLE TRAPPING OR [D] TO DISABLE TRAPPING OF THE FOLLOWING CONTROL CHARACTERS (ENTER A [RETURN] FOR NO CHANGE):

"P" (ENABLED):
"K" (ENABLED):

MAKE SELECTION PERMANENT [Y]-[N]?

Enter 'Y' to make the selected auto command permanent; otherwise, enter 'N' for a temporary change only. NOTE: This feature will be included in future releases of CP/M.

CONFIG B

This option will allow the user to adjust the baud rate of the selected list device; the adjustment made to the list device is not permanent. The prompt consists of the following:

SELECT THE BAUD RATE FROM THE FOLLOWING:
[110] [150] [300] [600] [1200] [2400] [4800] [9600]

3.2.3 SORTDIR Command

The SORTDIR command will read the directory from a specified disk, sort it in alphabetical order, and write the newly sorted directory back to the disk; SORTDIR will also automatically erase any files of 'ØK' in length from the directory. The command line consists of the following format:

SORTDIR <drive:>

3.2.4 EDIR Command

The EDIR command will read the directory from a specified disk, sort it in alphabetical order, and display the newly sorted directory on the terminal. EDIR has several options included for further flexibility which consist of the following:

? will list all the available options; does not require a filename in the command line.

S will list the file size of each file on the specified disk.

A will list the file attributes (R/O, R/W, SYS, etc.) of each file on the specified disk, including all system files not normally displayed.

N will list the directory unsorted; the EDIR function will normally sort the directory.

Un will list the directory from a specified user area (n = user number, 0-15), or from all user areas (n = *).

EDIR will accept either an afn or ufn, and when finished displaying the directory, will display the remaining disk space. The command line consists of the following format:

EDIR <drive:><filename> <option>

3.2.5 ERAX Command

The ERAX command will erase files from the currently logged disk allowing the user access to options not available with the ERA command. The ERAX command options consist of the following:

? will list all the available options; does not require a filename within the command line.

S will erase all files including system files.

Un will erase all files in a specified user area (n = user number, 0-15), or from all user areas (n = *).

Q will erase all files within a disk after an individual file query prompts the user whether to delete that particular file or not. This option will query with a '(Y-N)?' prompt for all files except Read/Only; if a R/O file is encountered during the Query, a message to that effect will display on screen and the next non-R/O file will display along with the Query prompt.

ERAX d:*.*[Q] not work

The ERAX command line consists of the following format:

ERAX <drive:><filename> <[option]>

and will return the following Query prompt line:

<drive:> [Un] <filename> (Y-N)?

If this file is to be deleted, enter 'Y'; if not, enter 'N' and ERAX will proceed to the next file.

3.2.6 STAT Command

The STAT command provides general statistical information about file storage and device assignment. It is initiated by entering one of the following forms:

STAT

STAT <command line>

Special forms of the 'command line' allow the current device assignment to be examined and altered as well. The various command lines which can be specified are shown below with an explanation of each form.

STAT

If the user types an empty command line, the STAT command calculates the storage remaining on all active drives and prints one of the two following messages:

d: R/W, Space: nnnk

or

d: R/O, Space: nnnk

where 'd' = the active drive, 'R/W' indicates the drive may be read or written to, 'R/O' indicates the drive is read only (a drive becomes R/O by explicitly setting it to be read only, as explained later, or by inadvertently changing diskettes without performing a warm boot) and 'nnn' indicates the space remaining on the diskette in Drive 'd' given in kilobytes.

STAT d:

If a drive name is given, the drive is selected before the storage is computed. Thus, the command 'STAT B:' could be entered 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 which satisfy the afn are listed in alphabetical order, with storage requirements for each file under the following heading:

```
RECS BYTS EX D:FILENAME.TYP
 rrr  bbbk ee d:ffffffff.ttt
```

where 'rrr' is the number of 128-byte records allocated to the file, 'bbb' is the number of kilobytes allocated to the file (bbb=rrr x 128/1024), 'ee' is the number of 16k extents (ee=bbb/16), 'd' is the drive name containing the file (A - Z), 'ffffffff' is the filename containing a maximum of eight characters, and 'ttt' is the extension type containing a maximum of three characters. After listing the individual files, the storage usage is summarized.

The drive name can be given ahead of the afn, as shown below; in this case, the specified drive is selected first and the specified afn form is then executed.

STAT d:<afn>

The STAT program has a number of additional functions which allow disk parameter display, user number display, and file indicator manipulation. The command **'STAT VAL:'** produces a summary of the available status commands, resulting in the following output:

```
Temp R/O Disk - d:R/O
Set Indicator - d:filename.ext $R/O $R/W $SYS $DIR
Disk Status   - DSK:d:DSK
User Status   - USR:
IObyte Assign - <possible assignments>
```

which gives an instant summary of the possible STAT commands.

The command form 'STAT d:filename.ext $S', where 'd:' is an optional drive name and 'filename.ext' is an unambiguous filename, produces the output display format:

Size	Recs	Bytes	Ext	Acc
48	48	6K	1	R/O A:GAM.COM
55	55	12K	1	R/O A:PIP.COM
65536	128	2K	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 the configuration time, and 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, may contain unallocated 'holes' even though they are added into the record count).

The 'Ext' field counts the number of logical 16K extents allocated to the file. The Ext count does not necessarily correspond to the number of directory entries given to the file, since there can be up to 128K bytes (8 logical extents) directly addressed by a single directory entry, depending upon allocation size (in a special case, there are actually 256K bytes which can be directly addressed by a physical extent).

The 'Acc' field gives the R/O or R/W access mode, which is changed using the commands shown below. Similarly, the parentheses shown around the PIP.COM filename indicate that it has the system indicator set, so that it will not be listed in DIR commands. The four command forms consist of the following and will set or reset various permanent file indicators:

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

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 boot operations.

The **R/W** indicator places the file in a permanent read/write status.

The **SYS** indicator removes the file from a normal directory display.

The **DIR** indicator removes the SYS indicator from a file.

The 'filename.ext' may be ambiguous or unambiguous, but in either case, the files whose attributes are changed are listed at the terminal 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 result in a BDOS error message **'ERROR - d:FILE IS READ-ONLY'**. The BDOS then waits for a console input before performing a subsequent warm boot (a [RETURN] is sufficient to continue). The command form 'STAT d:DSK:' lists the drive characteristics of the disk named by 'd:' which is in the range A:, B:,...P:. The drive characteristics are listed in the format:

```
    d: Drive Characteristics
62580: 128 Byte Record Capacity
 8160: Kilobyte Drive Capacity
  512: 32 Byte Directory Entries
    0: Checked Directory Entries
  512: Records/Extent
   64: Records/Block
   64: Sectors/Track
    4: Reserved Tracks
```

where 'd:' is the selected drive, followed by the total record capacity (65536 is an 8-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 usually one quarter the directory size for removable media, since this mechanism is used to detect changed media during CP/M operation without an intervening warm boot. For fixed media, the number is usually zero, since the media is not changed without at least a cold or warm boot. The number of records per extent determines the addressing capacity of each directory entry (512 times 128 bytes, or 64K in the example above).
The number of records per block shows the basic allocation size (in the example, 64 records/block times 128 bytes per record, or 8K bytes per block).

The listing is then followed by the number of physical sectors per track and the number of reserved tracks. 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 which 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 the above case, the active user number is '0' (default at cold boot), with three user numbers which have active files on the current disk. The user can subsequently examine the directories of the other user numbers by logging-in with **'USER 1'**, **'USER 2'**, etc. commands, followed by a DIR command at the CCP level.

3.2.7 PIP Command

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 entering one of the following forms:

PIP

PIP <command line> <parameters>

In both cases, PIP is loaded into the TPA and executed. In the first example, PIP reads command lines directly from the console, prompted with the asterisk (*) character, until an empty command line is entered, i.e., a [RETURN] is entered without preceeding data. Each successive command line causes some media conversion to take place according to the rules shown below. The second example 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 consists of the following:

Destination=Source 1,Source 2,...Source n

where 'Destination' is the file or peripheral device to receive the data, and Source 1 through n represents a series of one or more files or devices which are copied from left to right to the destination.

When multiple files are given in the command line, 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 which exceed the console width).

The destination and source elements can be ufn references to CP/M source files, with or without a preceeding disk drive name defining the particular drive where the file may be obtained or stored. 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, are valid as input to PIP:

X=Y

Copy to the file 'X' from file 'Y', where X and Y are unambiguous filenames; Y remains unchanged.

X=Y,Z

Concatenate files 'Y' and 'Z' and copy to file 'X', with Y and Z unchanged.

X.ASM=Y.ASM,Z.ASM,FIN.ASM

Create the file 'X.ASM' from the concatenation of the 'Y', 'Z', and 'FIN' files with the extension of '.ASM'.

B:A.U=B:B.V,A:C.W,D.X

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.

For more convenient use, PIP allows abbreviated commands for transferring files between disk drives consisting of the following:

PIP X:=<afn>
PIP X:=Y:<afn>
PIP <ufn>=Y:
PIP X:<ufn>=Y:

The first form copies all files from the currently logged disk which satisfy the afn to the same filenames on Drive X (X=A...Z). The second form is equivalent to the first, where the source for the copy is Drive Y (Y=A...Z). The third form is equivalent to the command 'PIP <ufn>=Y:<ufn>' which copies the file given by the ufn from Drive Y to the file ufn on Drive X. The fourth form is equivalent to the third, where the source disk is explicitly given by 'Y'.

Note that the source and destination disks must be different in all of these cases. If an afn is specified, PIP lists each ufn which satisfies the afn as it is being copied. If a file exists by the same name as the destination file, it is removed upon 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 which have the extension of '.COM' to Drive B from the current drive.

A:=B:ZAP.*

Copies all files which have the filename of 'ZAP' to Drive A from Drive B.

ZAP.ASM=B:

Equivalent to ZAP.ASM=B:ZAP.ASM

B:ZOT.COM=A:

Equivalent to 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 also allows reference to physical and logical devices which are attached to the CP/M system, along with a number of specially named devices; the logical devices consist of:

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

while the physical devices are:

TTY: (console, reader, punch or list)
CRT: (console, or list)
PTR: (reader), UR1: (reader), UR2: (reader)
PTP: (punch), UP1: (punch), UP2: (punch)
LPT: (list), UL1: (list)

Note that this assignment is used only to indicate that the RDR: and LST: devices are to be used for console I/O.

The RDR:, LST:, and CON: devices are all defined within the BIOS portion of CP/M. The destination device must be capable of receiving data (e.g., data cannot be sent to the punch), and the source devices must be capable of generating data (e.g., the LST: device cannot be read).

The additional device names which can be used in PIP commands are the following:

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: Special PIP input source which can be 'patched' into the PIP program itself.

The user can also specify one or more PIP parameters, enclosed in left and right brackets and 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 exception). Following is a list of valid PIP parameters:

B Block mode tranfer; data is 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 which can be buffered is dependent upon the memory size of the host system (PIP will issue an error message if the buffers overflow).

Dn Delete characters which extend past column 'n' in the transfer of data to the destination from the character source. This parameter is used most often to truncate loing lines which are sent to a (narrow) printer or console device.

E Echo all transfer operations to the console as they are being performed.

F Filter form feeds from the file. All imbedded form feeds are remove4d. The P parameter can be used simultaneously to insert new form feeds.

Gn Get file from user number 'n' (n = Ø - 15).

H Hex data transfer; all data is checked for proper Intel hex file format. Nonessential characters between hex records are removed during the copy operation. The console will be prompted for corrective action in case errors occur.

I Ignore ":00" records in the transfer of Intel hex format file (the I parameter automatically sets the H parameter).

L Translate upper case alphabetics to lower case.

N Add line numbers to each line transferred to the destination starting at one, and incrementing by 1. Leading zeros are suppressed, and the number is followed by a colon. If 'N2' is specified, leading zeros are included and a tab is inserted following the nubmer. The tab is expanded

if 'T' is set.

Pn Include 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 Quit copying from the source device or file when the string 's' (terminated by [CTRL Z]) is encountered.

R Read system files; files with the system attribute can be included in PIP tranfers using the R parameter; otherwise, system files are not recognized. The system file attributes are copied also.

Ss z Start copying from he source device when the string 's' is encoutnered (terminated by [CTRL Z]). The S and Q parameters can be used to 'abstract' a particular section of a file, such as subroutine. The start and quit strings are always included int he copy operation.

Tn Expand tabs ([CTRL L] characters) to every 'nth' column during the transfer of characters to the destination from the source.

U Translate lower case alphabetics to upper case during the copy operation.

V Verify the data has been copied correctly by rereading after the write operation (the destination must be a disk file).

W Copy all nonsystem files and overwrite any R-O (read-only) files in the process; eliminates the 'delete' prompt when purposely overwriting an existing file.

Z Zero the parity bit on input for each ASCII character.

3.2.8 FORMAT Command

The FORMAT command allows the user access to a general purpose disk formatter, written to run under all present versions of CP/M supported by the DualMode Controller Board.

There are two types, or options, of formatting available: The initial physical hard disk format (when there is no system or data on the disk), and the regular floppy and hard disk format.

A. FORMAT P - Initial Physical Hard disk Format

This command option will format a Winchester hard disk from a floppy disk. The following dialogue consists of the prompts and explanations when using the 'P' option.

A>FORMAT P

The user enters this command line format to run the physical hard disk FORMAT program.

DISK FORMATTER VERSION 1.0

SELECT [A] SURFACES 0,1
[B] SURFACES 2,3
[C] ALL SURFACES

SELECT SURFACES FROM ABOVE:

This prompt will only be displayed if your 5-inch Winchester is configured as a dual logical drive; Enter 'A' to format surfaces Ø and 1 (Drive A), 'B' to format surfaces 2 and 3 (Drive B), or 'C' to format all surfaces (Drives A and B.

Otherwise the following prompt will be displayed.

WARNING - PROCEDURE ERASES DATA.
PRESS [RETURN] TO CONTINUE, [CTRL C] TO ABORT

This warning message will informs the user that to continue will erase all data on the disk. If there is any information (data or programs) on the disk that the user does not want to lose, enter a [CTRL C] to abort the operation and save that information; otherwise, enter a [RETURN] to continue.

FORMAT COMPLETE

The selected physical surfaces have been successfully formatted; it is now possible to create a CP/M operating system and transfer any needed programs to the formatted disk.

B. FORMAT /options - Regular formatting with options

This command, with options, will format single or double sided diskettes and, if necessary, selected areas of the Winchester hard disk, in case of non-correctable disk errors or other problems with the disk. The options available are:

- R - for Repeat FORMAT function.
- E - "E" provides error messages as to specific BIOS error codes.

The following dialogue consists of the prompts and explanations of FORMAT without options:

FORMAT

The user enters this command line format, without options, to run the regular FORMAT program. After the title and version number display, the following prompt will appear (the selections vary depending on the system configuration).

```
DRIVE: [A] 5 MBYTE WINCHESTER - SURFACE 0,1
       [B] 5 MBYTE WINCHESTER - SURFACE 2,3
       [C] 5 INCH SINGLE-SIDED FLOPPY
       [D] 5 INCH DOUBLE-SIDED FLOPPY
       [E] 5 INCH DOUBLE-SIDED FLOPPY

MAKE SELECTION FROM ABOVE:
```

Enter the specific drive type to be formatted (usually this will be for formatting floppies). The only time the hard disk should be formatted is if the disk is damaged or is getting permanent disk errors. The drives shown above may be different depending on the configuration of the user's system.

If one of the hard disk surfaces is chosen, the following message will display:

```
WARNING - PROCEDURE ERASES DATA.
PRESS [RETURN] TO CONTINUE, [CTRL C] TO ABORT
```

This warning message will display to inform the user that continuing will erase all data on the disk.

If there is any information (data or programs) on the disk that the user does not want to lose, enter a [CTRL C] to abort the operation and save that information; otherwise, enter a [RETURN] to continue.

FORMAT COMPLETE

If one of the floppy drives is chosen, the following message will display:

INSERT DISKETTE AND
PRESS [RETURN] TO BEGIN, [CTRL C] TO ABORT

If there is any information (data or programs) on the disk that the user does not want to lose, enter a [CTRL C] to abort the operation and save that information; otherwise, enter a [RETURN] to begin formatting.

FORMAT COMPLETE

If the user wants to format several diskettes, without entering 'FORMAT' each time, the Repeat ('R') option can be used within the command line, that is:

FORMAT R

This command option will display the regular FORMAT prompts; however, instead of the program ending after the prompt 'FORMAT COMPLETE', it will continue to prompt the user to make a drive selection to be formatted. FORMAT will continue prompting until the user enters a [CTRL C] to terminate the program.

If the user wants to format diskettes, which will return specific types of permanent errors, the Error ('E') option can be used within the command line, that is:

FORMAT E

This command option will, again, display the regular FORMAT prompts, except that when it is finished formatting, it will give a more specific error message, i.e.:

FORMATTING DISKETTE IN DRIVE C

PERMANENT ERROR - FLOPPY WRITE PROTECT ERROR

3.2.9 DISKCOPY Command

The DISKCOPY command allows the user access to the floppy disk backup program designed to run under CP/M versions supporting the DualMode Controller Board. This program will perform backups for single- and double-sided diskettes, and, if a Winchester hard disk is available, will allow the hard disk to act as temporary storage for the source diskette, eliminating the swapping of source and destination diskettes. With this option, the source diskette is copied to the hard disk temporarily, if there is enough space available (storage on the hard disk will be reduced by a maximum of 600K), and finally copied to the backup diskette, without the need to swap source and destination diskettes over and over; also, multiple copies of one diskette can be made without further accessing the source diskette using the 'Repeat' option shown below. If the current drive is not a Winchester or if insufficient space exists on the drive, the backup must be made in the normal fashion.

NOTE: This program does not allow backup from a drive configured as double-sided onto a drive configured as single-sided. Also, any attempt to copy from a single-sided diskette in a drive configured as double-sided may result in a 'PERMANENT ERROR ON SOURCE' message.

There are two options available with DISKCOPY: the Repeat (R) copy option and the return Error (E) option. The command line format consists of the following form, with or without options:

DISKCOPY <option(s)>

where either option, 'R' and/or 'E', may be used. The dialogue consists of the following:

A>DISKCOPY <option(s)>

The user enters this command line format, with or without options, to run the floppy backup program.

DISKCOPY - VERSION 1.3

DRIVE: [C] 5 INCH SINGLE-SIDED FLOPPY
[D] 5 INCH DOUBLE-SIDED FLOPPY
[E] 5 INCH DOUBLE-SIDED FLOPPY

SELECT SOURCE DRIVE FROM ABOVE:

The drives shown above may vary with what is shown on screen, again because of the different configuration of the user's system. Enter the source drive name; i.e., the drive the source diskette is in (the one to be copied).

SELECT DESTINATION DRIVE FROM ABOVE:

Enter the destination drive, or where the information is to be copied. Remember, if the source is in a single-sided drive and the destination is in a double-sided drive, the destination diskette will be single-sided. However, the opposite is not valid; a double-sided source drive will not write to a single-sided destination drive. A drive will read a diskette exactly as the drive was configured; i.e., a single-sided drive will read any diskette in it as single-sided, and a double-sided drive will try to read any diskette in it as double-sided.

DO YOU WANT TO USE THE
WINCHESTER FOR TEMPORARY STORAGE (Y/N):

This prompt will only display when a Winchester hard disk is in use; this option allows floppy disk duplication on a system with a Winchester drive and one floppy drive. Enter 'Y' to use the Winchester in a diskette to diskette backup; enter 'N' and the program will display the 'INSERT' diskette prompts. If insufficient free space exists on the currently logged drive (if the Winchester option is chosen), the program will respond with

INSUFFICIENT FREE SPACE

If the user wants to use the hard disk for copy purposes, log onto another drive of the hard disk which has more space available and rerun the program.

INSERT SOURCE DISKETTE AND
PRESS [RETURN] TO BEGIN, OR [CTRL C] TO ABORT:

This prompt and the following prompt will display when the Winchester is **not** being used or if the Winchester does not have enough space to temporarily store the contents of the source diskette. DISKCOPY will erase or delete the original contents of the destination diskette during the copy operation. Insert the diskette, if not already inserted, and enter [RETURN] to begin the backup, or [CTRL C] to abort the program, if there is information on the destination diskette that should not be deleted.

INSERT DESTINATION DISKETTE AND
PRESS [RETURN] TO BEGIN, OR [CTRL C] TO ABORT:

This prompt and the previous prompt will alternate until the contents of the source diskette has been written to the destination diskette.

DISKCOPY COMPLETE

3.2.10 STORE Command

The STORE command allows the user to 'backup' files from a specified user area on the hard disk which exceed the capacity of one floppy diskette. STORE has a command syntax which is virtually identical to the PIP program; the one difference is that it can only copy files which have names that are seven letters or fewer--eight letter filenames are ignored.

The most important point to remember about STORE is that it erases all files in all user areas of the floppy which has been inserted into the drive for backup. The program checks to confirm that the output diskette is a floppy.

When STORE copies a file, it adds a '$' to the end of the filename. For this reason, a filename can only consist of up to seven characters instead of the usual eight. When the first diskette becomes full, the '$' at the end of the file being copied is changed to a zero (Ø) and the file is closed. A new diskette is requested and a file is created with a '1' as the last character of the filename. If the file fills the second diskette to capacity, the program will request a third diskette. A file with a '2' at the end of the filename will be created. This can continue until the entire file has been copied to as many floppies as needed. When the file has been completely written, the program will change the number at the end of the filename to a '$' and begin copying the next file, should there be another file.

Therefore, a '$' at the end of a filename signifies the last section of the file, and a number signifies that more of the file exists on another diskette. Note that if the filename in the command line ends with an asterisk (*), any eight-character filename fitting the specification will not be copied. For example, if you enter:

STORE C:=REST*.COM

RESTORE.COM would be copied but RESTORES.COM would not. If you are using STORE to backup your files on a regular basis, it is imperative that all filenames consist of seven characters or less. It is also important that the user number each backup diskette so when a RESTORE is attempted, the diskettes can be entered in proper sequence.

To abort the program, enter a [CTRL C] and a [RETURN] when prompted for another backup diskette. This is not recommended because it leaves an incomplete backup; however, it can be useful should be become necessary for the user to quit processing prematurely.

3.2.11 RESTORE Command

The RESTORE command allows the user to transfer programs created by STORE, from the floppy diskette back to the hard disk; only use RESTORE to copy files created by STORE.

When entering a filename for RESTORE, it should be entered as it originally appeared in the directory of the hard disk; i.e., the filename should not have the '$' or number at the end, unless the number was part of the original filename.

Be sure to insert the diskettes in the correct order. Once RESTORE has begun copying files, it will ask the user for new diskettes as it needs them. The file which RESTORE searches for must be at the beginning of the diskette or 'FILE NOT FOUND' will display and another diskette will be requested.

Only after RESTORE ascertains that it has copied all files, will it display the message

INSERT NEXT DISK TO CONTINUE OR [CTRL C] TO EXIT

It may happen that, when using the 'wild card' to copy several files, RESTORE will find a filename ending with '$' and the file will be the last file on the diskette. There may be other files on the diskettes to be copied; however, RESTORE will have no way to know this.

The user will have to be alert to the situation and know to

INSERT THE NEXT DISK TO CONTINUE

which will allow RESTORE to proceed.

The only time RESTORE can be aborted, without resetting the system, is when the program has finished copying a diskette.

As in PIP, the 'G' option is used to get files from other user areas. The user number following the 'G' must be in the range of Ø to 15!

If the hard disk is full or out of directory space, the program will abort. The problem of a full disk can sometimes be circumvented with the 'K' (Kill) option. This option is available with RESTORE only. It may precede or follow the 'G' option. It should be used cautiously because the 'K' option causes the file it is copying to be deleted or 'killed' on the hard disk before it is copied. Normally, a file is created with the filename specified but with an extension of '$$$'. Only after this file has been written to and closed will the original output file with the same name be killed if it exists. This temporary file is then renamed so that it has the proper extension. The 'K' option should only be used in the following cases:

- The file you want to copy already exists on the hard disk
- There is not enough room on the hard disk for the file
- You cannot make enough room on the hard disk by deleting unwanted files

3.3 PROGRAMMING COMMANDS

3.3.1 SCOPE Command

The SCOPE (SC) command will allow the user to access SCOPE, Vector's SCreen Oriented Program Editor, a program editor for Vector computers to create and edit any VECTOR CP/M ASCII file, either program text files or data files. See the VECTOR SCOPE REFERENCE MANUAL for more information on this editor.

3.3.2 ZSM Command

The ZSM command will allow the user to convert Z-80 assembly language text (source code) into a sequence of machine language instructions (object code) which can then be loaded into the computer's memory and executed. See the ZSM ASSEMBLER USER'S MANUAL for more information on this assembler.

3.3.3 LOAD Command

The LOAD command will read the file (ufn), which is assumed to contain Intel 'hex' format machine code, and produce a memory image file which can be subsequently executed. The input file is assumed to have an extension of '.HEX' and, therefore, only the filename need be specified in the command. The LOAD command creates a file with an extension of '.COM' which marks it as containing machine executable code. The file is actually loaded into memory and executed when the user enters the filename immediately after the 'A>' displayed by the CCP (this loaded program becomes a new command under the CCP).

In general, the CCP reads the filename following the prompt and looks for a resident function name. If no function name is found, the CCP searches the system disk directory for the file with an extension of '.COM'. If found, the machine code is loaded into the TPA and the program is executed. The user need only load a hex file once; it can be subsequently executed any number of times by simply typing the filename. In this way, the user can create new commands for use within the CCP (initialized disks contain the transient commands as COM files, which can be deleted at the user's option).

The operation can take place on an alternate drive if the filename is prefixed by a drive name; e.g., **'LOAD B:BETA'** brings the LOAD program into the TPA from the currently logged disk and operates in Drive B after execution of the program begins. It must be noted that the 'BETA.HEX' file must contain valid Intel hexadecimal machine code records (as produced by the ZSM program) beginning at 1ØØH, the beginning of the TPA. Further, the addresses in the hex records must be in ascending order, gaps in unfilled memory regions are filled with zeros by the LOAD command as the hex records are read. Therefore, LOAD must be used only for creating CP/M-standard '.COM' files which operate in the TPA. Programs which occupy regions of memory other than the TPA can sometimes be loaded under DDT.

3.3.4 SUBMIT Command

The SUBMIT command will allow CP/M commands to be batched together for automatic processing. The ufn given in the SUBMIT command must be a filename which exists on the currently logged disk, with an assumed extension 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 SCOPE 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 command is executed, the actual parameters 'parm#1' are paired with the formal parameters '$1...$n' in the prototype commands. If the number of formal and actual parameters does not correspond, the SUBMIT function is aborted with an error message at the terminal.

The SUBMIT function creates a file of substituted commands with the name '$$$.SUB' on the logged disk. When the system reboots (at the termination of the SUBMIT), this command file is read by the CCP as a source of input rather than the terminal. If the SUBMIT function is performed on any disk other than Drive A, the commands are not processed until the disk is inserted into Drive A and the system reboots. Further, the user can abort command processing at any time by entering a [DEL] when the command is read and echoed. In this case, the $$$.SUB file is removed, and the subsequent commands come from the terminal. Command processing is also aborted if the CCP detects an error in any of the commands.

In order to introduce dollar signs into a SUBMIT file, the user may enter a '$$' which reduces to a single '$' within the command file. Further, an up arrow may precede an alphabetic character 'x', which produces a single [CTRL X] character within the file. The last command in a SUB file can initiate another SUB file, thus allowing chained batch commands.

For example, the file 'ASMBL.SUB' exists on disk and contains the prototype commands:

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

and the command 'SUBMIT ASMBL X PRN' is entered by the user. The SUBMIT program reads the ASMBL.SUB file, subsitituting 'X' for all occurrences of '$1' and 'PRN' for all occurrences of '$2', resulting in a '$$$.SUB' file containing the following commands which are executed in sequence by the CCP:

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

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

In certain cases, STAT will show user areas that are between 16 and 31, and the data in these areas cannot be accessed. This is the result of previously running SUBMIT and interrupting or aborting it in an unusual manner. In order that other users can execute a SUBMIT file, the system creates the $$$.SUB file in the default User area plus 16. If the SUBMIT aborts in an unusual manner this will reflect when using STAT. If you want to take the data off the disk and, therefore, the directory, execute the SUBMIT again and exit in a 'normal' manner.

3.3.5 XSUB Command

The XSUB utility program will extend the power of the SUBMIT facility to include line input to programs as well as the console command processor. The XSUB command is included as the first line of the SUBMIT file and, when executed, self-relocates directly below the CCP.

All subsequent SUBMIT command lines are processed by XSUB so that programs which read buffered terminal input (BDOS function 10) receive their input directly from the SUBMIT file. For example, the file 'SAVER.SUB' could contain the following SUBMIT lines:

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

with a subsequent SUBMIT command (SUBMIT SAVER X Y) which substitutes X for $1 and Y for $2 in the command stream. The XSUB program loads, followed by DDT which is sent the command lines 'IX.HEX', 'R', and 'G0' and is finally returned 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 boot operation to indicate its presence. Subsequent SUBMIT command streams do not require the XSUB, unless an intervening cold boot has occurred. Note that XSUB must be loaded after DESPOOL, if both are to run simultaneously.

3.3.6 DDT Command

The DDT command will allow dynamic interactive testing and debugging of programs generated in the CP/M environment. See the CP/M DYNAMIC DEBUGGING TOOL (DDT) USER'S GUIDE for more information on this debugging program.

3.3.7 RAID Command

The RAID command will allow the user to implement Vector's Rapid Interactive Debugger program. RAID simulates a program step by step by taking each instruction of the program, executing it and displaying the result of the operation, a valuable tool for assembly language software development. See the RAID - RAPID INTERACTIVE DEBUGGER, REFERENCE MANUAL for more information on this debugging program.

3.3.8 DUMP Command

The DUMP command will display the contents of the disk file (ufn) at the terminal 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 displays can be aborted by entering any key (except [CTRL S]) during the printout.

3.3.9 GENLIST Command

The GENLIST command will allow the user to create the list device files for VECTOR 4 CP/M. It will take any 'hex' file and convert it to an '.LD1' or '.LD2' file for use by the system. To execute the program enter 'GENLIST <filename>'. GENLIST will sign on and then process the file. Provided there are no errors, GENLIST will display the load address, the last address, and then sign off. The generated file may then be copied (using PIP) into the appropriate 'VECTOR.LDn' file for use by the system.

To write and load a printer driver program, the code should be assembled to execute at address Ø1ØØH for Driver 1 and Ø5ØØH for Driver 2 (see Exhibit 3-1). The code for Driver 1 cannot exceed 1K, and the code for Driver 2 cannot exceed 1.5K.

The code should be written in an 'in-line' fashion; i.e., do not use multiple 'ORG' statements. This will generate an error when the GENLIST program is run. The following format must be adheared to for all printer drivers:

Exhibit 3-1. PRINTER DRIVER TABLE

LOCATION	BYTE	REMARKS
DVR + ØØ	TYPE1	The first byte contains the printer Class (C) and Type (T) codes. The printer class code is contained in the upper nybble (bits 4-7) while the printer type code is contained in the lower nybble (bits 0-3); e.g.: BIT 7 —> CCCCTTTT <— BIT Ø The assigned printer class codes consist of the following: ØØØØ = Undefined ØØØ1 = Draft Printer (Lineprinters) ØØ1Ø = Word Processing Printer (Letter Quality) ØØ11 - 1111 = Undefined

The assigned printer type codes depend on the printer class and are assigned as follows:

Type Code	Class 1	Class 2
ØØØØ	No Printer	No Printer
ØØØ1	S.D.*	P.W.P.**
ØØ1Ø	P.D.***	P.W.P.
ØØ11	S.D.	S.W.P.****
Ø1ØØ		
Ø1Ø1	Undefined	Undefined
1111	Custom	Custom

* Serial Draft Printer
** Parallel Word Processor Printer
*** Parallel Draft Printer
**** Serial Word Processor Printer

DVR + Ø1 TYPE2

The second byte contains the interface descriptor byte. It is composed of four separate fields: Hardware (H), Undefined (U), Protocol (P), and Interface (I); i.e.:

BIT 7 —> HHH U PPP I <— BIT Ø

The upper three bits (HHH) contain the hardware environment code and have been assigned the following values:

ØØØ = Undefined
ØØ1 = Not Applicable
ØlØ = Vector 4 SBC
Ø11 = Undefined
1ØØ = Undefined

Bit 4 (U) is an undefined field.

Bits 1 through 3 (PPP) contain the protocol code and have been assigned the following values:

ØØØ = Hardware Handshaking
ØØ1 = ETX/ACK Protocol
ØlØ = X-ON/X-OFF Protocol
Ø11 - 111 = Undefined

		Bit Ø (I) is the interface type bit. If this bit is set (1), the interface will be serial; if this bit is reset (Ø), the interface is parallel.
DVR + Ø2	INIT	This routine should take care of any printer initialization.
DVR + Ø5	INSTAT	This is the input character status; it should return 'ØØ' if no character is ready or 'FF' if ready. If no input is possible, this routine should always return 'ØØ'; a returned value should always be in the A register.
DVR + Ø8	INCHAR	This is the input character from the printer which should return the character in the A register. If no input is possible, this routine should return a 'ØØ' in the A register.
DVR + ØB	OTCHAR	This is the output character to the printer which is passed in the C register.
DVR + ØE	OTSTAT	This is the character output status. This routine should return a 'ØØ' if the printer is not ready for output or an 'FF' if it is ready; return values should be in the A register.
DVR + 11	FORMLN	This is the total number of physical lines per page.
DVR + 12	AUTOPG	This is the auto paging flag. This byte should be set to 'ØØH'.
DVR + 13	FORMPR	This is the total of the number of printed lines per page and the number of blank lines at the top of each page.
DVR + 14	TOPMRG	This is the number of blank lines at the top of each page.

Printer Type Bytes

There are two bytes, located at the front of all VECTOR CP/M printer driver modules (i.e., all '.LD1' and '.LD2' files), that provide information about the printer and its interface to the system.

The '.LDn' files can be generated from an appropriate hex file by using the GENLIST program. Several drivers are included on the system disk and may be copied (using PIP) into the appropriate driver file (i.e., **PIP VECTOR.LD1=SERIAL.LD1**). After a printer driver file has been copied to the VECTOR.LD1 or .LD2 file, it is necessary to reset and reboot the system in order to use the newly defined driver file.

The source files for both the serial and parallel drivers are included as an example on how to generate a driver file. These may simply be altered, or a new file may be generated. Following is a list of the printer driver files included on the system disk:

VECTOR.LD1	Copy of SER4.LD1
VECTOR.LD2	Copy of NV4.LD2
SER4.LD1	Serial Driver
SER4/H.LD1	Serial Driver w/ ETX/ACK protocol
SER4/X.LD1	Serial Driver w/ XON/XOFF protocol
PAR4.LD1	Parallel Driver w/ "Centronics Type" protocol
NV4.LD2	Parallel Driver w/ "NEC Type" protocol
QV4.LD2	Parallel Driver w/ "Qume Type" protocol
SERIAL.ASM	Source File for Serial Driver
PARALL.ASM	Source File for Parallel Driver

3.4 TEST COMMANDS

3.4.1 DISKTEST Command

See the DISKTEST USER'S MANUAL for a complete description of this command.

3.4.2 MEMTEST Command

See the VECTOR 4 - TECHNICAL INFORMATION MANUAL for a complete description of this command.

3.4.3 CRC Command

The CRC (Cyclic Redundancy Check) Command will allow the user to return the CRC value(s) of a specified file, or files, for a comparison check with the CRC listing for each program or file on the System disk. Each file on the disk produces a specific CRC, and, when the CRC program returns each file's CRC value, these values may be checked against known values to determine if the file is bad or if the file or program is a non-usable or incompatible version.

The CRC value is based on the polynomial 'X*16+X*15+X*7+X*2+X*1+1'. The file parameters include normal CP/M wildcards and are in the same form as the DIR function. The CRC value will be calculated and displayed for each file matching the file parameters given. The CRC program interfaces with the user through the CCP, and the file specification parameters are passed by the normal CCP linkages. Following are examples of valid command lines to find the CRC values for specified file(s):

```
CRC TEST.COM
CRC A:*.*
CRC MBASIC?.*
CRC B:*.COM
CRC A:VECTOR.LD?
```

Each file matching the file specifications will be listed (the output will not be paged) along with its CRC value in the following format:

```
NAME            CRC

XXXXXXXX.XXX    XXXX
```

For example, the command line

CRC *.COM

could result in the following display:

```
CRC VER 1.0

NAME            CRC

d:PIP     COM   1847
d:STAT    COM   B31E
d:SUBMIT  COM   154B
```

where 'd:' = the currently logged drive.

3.4.4 CPUTEST Command

See the VECTOR 4 - TECHNICAL INFORMATION MANUAL for a complete description of this command.

3.4.5 SCNTEST Command

See the VECTOR 4 - TECHNICAL INFORMATION MANUAL for a complete description of this command.

3.4.6 PORTEST Command

See the VECTOR 4 - TECHNICAL INFORMATION MANUAL for a complete description of this command.

3.4.7 KYBDTEST Command

See the VECTOR 4 - TECHNICAL INFORMATION MANUAL for a complete description of this command.

3.4.8 PRINTEST Command

See the VECTOR 4 - TECHNICAL INFORMATION MANUAL for a complete description of this command.

3.4.9 RECLAIM Command

A. HOW TO RECOVER A FILE USING RECLAIM

In rare instances, an error message may inform you that a disk has a bad sector. This could occur because of surface damage or because data has been otherwise scrambled. The bad sector could prevent you from recalling or transferring a file. RECLAIM is a special purpose program, included on your system disk, to help you recover data from a bad sector.

A bad sector refers to a damaged spot on the disk, either because of wear, mishandling, or because the disk has been subjected to a magnetic field or other condition which would cause the data to be inadvertently altered. When VECTOR 4 CP/M encounters a bad sector, it will report the error condition with the general message:

ERROR - d:DISK READ/WRITE

where 'd' is the disk drive name where the error occurred.

To recover from the error, you can enter either a [CTRL C], which will abort the program and return you to the operating system, or a [RETURN], which will tell the system to ignore the error and continue. However, the integrity of the disk might be destroyed if you ignore the error, and the bad sector may prevent you from reading or copying a file. RECLAIM is a program specifically designed for these situations, where a bad sector could cause you to lose data.

RECLAIM can be used to test the entire disk or to read a single file. It can be used to recover data from the hard disk or from a floppy diskette. The program reads and validates each sector of the user area of the disk, or each sector of the particular file to be recovered.

RECLAIM can be used in the 'Auto' or 'Edit' modes or together. The Auto mode will try to remap the disk or diskette if it encounters any errors. In the Edit mode, sectors with errors will be displayed on the screen and the operator may edit the sector, either in hex or ASCII.

If an error is found in the file space, when running RECLAIM, the program will prevent other programs from using the bad sector. When using RECLAIM with a particular file, the program will also attempt to transfer the data in the bad sector to another part of the disk.

If an error is found on the directory tracks, (the area of the disk where operating and storage information is kept), RECLAIM will abort. Information stored on these tracks is so critical to the operation of your disk, that once damaged, it cannot be recovered with RECLAIM. However, even if the directory tracks are damaged other data on the disk may be intact. It may help to transfer the data from the hard disk using the PIP or STORE programs.

In the Edit mode, use the [CTRL F] function to switch from hex to ASCII. To move the cursor, use the cursor control and [RETURN] keys. To terminate the edit and allow the rewrite or transfer to continue, enter an [ESC]. Only sectors containing errors will be presented for editing.

The Edit mode can only be used to test a particular file, i.e., it cannot be used if the entire disk is to be tested.

B. HOW TO USE RECLAIM

If you are testing the hard disk, boot from the hard disk and run the RECLAIM program (RECLAIM may be loaded from the hard disk if it is present, or may be loaded from a floppy, if necessary). The drive which will be reclaimed is determined in the command line; for example, to test Drive A, enter

RECLAIM A: You will see the banner and message:

VECTOR GRAPHIC DISK RECLAIMER - VER. 1.1

TESTING DIRECTORY SPACE

and a block of numbers in hexadecimal, which will indicate the block being tested. The numbers will rapidly change as the program tests each sector on the disk.

Should RECLAIM find an error in the directory space, it will inform the user with one of several error messages and end the program. See the following section on RECLAIM error messages for further explanation and procedures.

If RECLAIM finds the directory space to be in order, it will display the message:

TEST COMPLETE

and continue by testing the file space, displaying the message:

TESTING FILE SPACE

The block numbers will be displayed and will change as the test proceeds.

Should RECLAIM find an error in the file space, it will automatically remap the disk (if you are using the program in Auto mode) such that the space will not be used again. If you are trying to recover a particular file, RECLAIM will also attempt to rewrite the data in the bad sector to an unused part of the disk. Having tested all of the disk space, RECLAIM will display the message:

TEST COMPLETE

and will return control to the operating system.

If you are testing the hard disk and have configured it as more than one logical drive, run RECLAIM again, to test the other drive(s). The format for the command line should be:

RECLAIM d:<$A>

where 'd' represents the drive to be tested and '$A' denotes the Auto mode option. When testing the disk or diskette, only the Auto mode can be used as an option.

To use the Edit mode of RECLAIM, enter

RECLAIM d:<filename.ext> $E

where 'd' is the drive to be tested and 'filename.ext' is the specified file to be edited. Note that the '$E' option will only work on a single file at any one time, and never for an entire disk or diskette.

The follwing Edit Mode functions available will display on screen and can be used during the edit cycle:

EDIT MODE FUNCTIONS

-- EDIT FUNCTIONS --

USE ARROW KEYS TO POSITION CURSOR

[CTRL B] - HOME CURSOR

[CTRL F] - TOGGLE HEX/ASCII

[RETURN] - NEXT LINE

[ESC] - END EDIT

The Edit mode and the Auto mode can be run at the same time by entering:

RECLAIM d:<filename.ext> $AE

substituting the letter of the logical drive and the filename.

PRESS [RETURN] TO REWRITE OR [ESC] TO RECLAIM

C. READING RECLAIM ERROR MESSAGES

When the RECLAIM program encounters an error, it will inform the operator with one of several messages. Write the message on a piece of paper because it will help to troubleshoot 'fatal' errors; that is, errors which the RECLAIM program itself cannot resolve.

OPERATOR SIGN-ON ERROR MESSAGES:

ILLEGAL FILE NAME
FILE NOT FOUND

The filename has been improperly entered or the file does not exist.

ILLEGAL DRIVE
DRIVE NOT READY

The drive to be accessed does not exist in the System's logical drive configuration or no diskette has been mounted in the drive.

FILE IS READ ONLY
WRITE PROTECTED

The diskette has a write-protect tab or the file is protected within VECTOR 4 CP/M.

ERRORS THAT RECLAIM WILL RESOLVE

CHECK SUM
TRACK ADDRESS
SECTOR ADDRESS
HEAD ADDRESS
ECC VERIFICATION
DATA VERIFICATION
UNRECOVERABLE ECC
SYNC BYTE ERROR

These messages refer to errors which occur because data related to addressing and verification has been damaged. If you are using the program in the Auto mode, RECLAIM will remap the disk to ensure that no other programs write data to the bad sector. If you are using the Auto mode and you are testing a particular file, RECLAIM will also try to transfer the data to another sector. After the operation, if you have reclaimed a floppy diskette, copy whatever files you can save and discard the worn diskette.

HARDWARE ERROR MESSAGES

HOME ERROR
CONTROLLER BUSY
SEEK TIMEOUT
LOSS OF SYNC
WRITE FAULT
IMPROPER CONTROLLER JUMPER
SPASM ERROR

These messages infer that a hardware problem prevents correct operation of the disk drive controller. Call your Vector Graphic Dealer for assistance.

DIRECTORY ERROR MESSAGES

DIRECTORY I/O ERROR
FATAL ERROR IN DISK DIRECTORY

These messages are displayed because the error occurred in the directory space of the disk; in which case the program will abort. When a bad sector error occurs in the directory space, you must reinitialize the disk. First, backup whatever files you can access using PIP or STORE. Then run the FORMAT program as described in this manual. After running FORMAT, put a new VECTOR 4 CP/M Operating System on the hard disk, following the procedures in this manual and write your files back to the hard disk using PIP or RESTORE. If you are testing a floppy diskette, you should copy as many of your files as possible to another disk and discard the worn diskette.

CAN'T OPEN TEMP FILE
CAN'T OPEN TRANSFER FILE
CAN'T OPEN XXX.SYS

These messages occur because the directory is full or because of a directory error. First, try to erase one or more files from the directory and run RECLAIM again. If the message repeats itself, it is probable that an error occurred in the directory space, in which case the program will end. When this happens, backup the data you can access, using either PIP or STORE. Format the disk again and give it a new CP/M operating system. When testing a floppy diskette, copy as many files as you can access and discard the worn diskette.

VERSION ERROR MESSAGES

NOT SUPPORTED UNDER THIS CP/M VERSION

RECLAIM will not run with your system if you are using the wrong version of VECTOR 4 CP/M.

OTHER MESSAGES

SECTOR REWRITE SUCCESSFUL

The data was rewritten to the same sector successfully.

SECTOR REWRITE UNSUCCESSFUL

The data could not be recovered, and the bad sector was mapped so as not to be used again; if it is a file, data is moved to a different location on the disk.

UNUSEABLE DISK SPACE

When RECLAIM has finished, the program will inform you how much disk space was found to be unuseable.

ABORTING—CAN'T OPEN
TEMPORARY FILE
ABORTING—FATAL DISK ERROR
ABORTING—TEMP FILE ERROR
TEST ABORTED

These messages inform the operator that RECLAIM will end the program and return control to the VECTOR 4 CP/M operating system.

D. HOW TO FIND A TRACK AND SECTOR FROM A GIVEN BLOCK NUMBER

After running RECLAIM and backing up all of the data on the disk, you may want to use DISKTEST to step through the block which has the error. The following example will show you how to calculate a track and record number from a given block number. This equation will allow you to calculate the first sector of the block which contained the error.

Following is an example which works out the equation for a 5 1/4-inch hard disk drive, where the disk has a directory set for 256 entries and the given block number is 3.

1. Run RECLAIM to find the Block Number of the sector with the error. Convert the Block Number from hexadecimal to decimal.

2. Use the Parameter Table (Exhibit 3-2) to find the number of Records per Track, the number of Records per Sector, the number of Blocks for the Directory, the number of Reserved Tracks, and the number of Records per Block for your system.

Records per Track:_____	Records per Track:__64__
Records per Sector:_____	Records per Sector:__2__
Blocks for Directory:_____	Blocks for Directory:__2__
Reserved Tracks:_____	Reserved Tracks:__4__
Records per Block:_____	Records per Block:__32__

3. Work the following equation to find the Base Value.

   BASE VALUE = (RESERVED TRACKS)*(RECORDS PER TRACK)

   4(64)

   Base Value:_____ Base Value:__256__

4. Work the following equation to find the Total Number of Records.

   TOTAL RECORDS =

   BASE VALUE + (BLOCK NUMBER * RECORDS PER BLOCK)

   256 + (3 x 32)

   Total Records:_____ Total Records:__352__

5. Work the following equation. The integer portion of the result gives you t track number. The fractional portion of the result will be used to calcula the sector number.

TRACK = TOTAL RECORDS / RECORDS PER TRACK

352 / 64 = 5.5

Track:_____ Track:__5__

6. Multiply the fractional portion, left from the division performed in step 6, the number of records per track, and divide by the number of records p sector. The result gives you the Sector.

SECTOR = FRACTIONAL RESULT * RECORDS PER TRACK / RECORDS PER SECTOR

.5 * 64 / 2

Sector:_____ Sector:__16__

Exhibit 3-2. RECLAIM PARAMETER TABLE

IF THE DRIVE USES:	RECORDS PER TRACK	RECORDS PER SECTOR	BLOCKS FOR DIRECTORY	RESERVED TRACKS	REC PER BLO
5 1/4" SINGLE-SIDED FLOPPY	32	2	1	2	16
5 1/4" DOUBLE-SIDED FLOPPY	32	2	2	4	16
5 1/4" HARD DISK	64	2	2 (256 entries) 4 (512 entries)	4	32

SECTION IV - BDOS SYSTEM CALLS

4.1 INTRODUCTION

VECTOR 4 CP/M is an extremely powerful and flexible operating system that can support many different type of programs. It accomplishes this task by setting up a protocol, or method of communication, so that all programs, whether they are BASIC interpreters, word processors, or any other application program or utility, will be able to run effectively.

VECTOR 4 CP/M is logically divided into four parts: the Basic Input/Output System (BIOS), the Basic Disk Operating System (BDOS), the Console Command Processor (CCP), and the Transient Program Area (TPA).

The BIOS is a hardware-dependent module which defines the exact low level interface to a particular computer system which is necessary for peripheral device I/O. The BIOS and BDOS are logically combined into a single module with a common entry point, and referred to as the FDOS. The CCP is a distinct program which uses the FDOS to provide a human-oriented interface to the information which is cataloged on the backup storage device. The TPA is an area of memory (i.e., the portion which is not used by the BDOS, BIOS, and CCP) where various nonresident operating system commands and user programs are executed.

All standard CP/M versions, assume 'BOOT = ØØØØH', which is the base of Random Access Memory (RAM). The machine code found at location BOOT performs a system 'warm start' which loads and initializes the programs and variables necessary to return control to the CCP; therefore, transient programs need only jump to location BOOT to return control to CP/M at the command level. Further, the standard versions assume 'TBASE = Ø1ØØH'. The principal entry point to the FDOS is at location 'ØØØ5H' where a jump to FBASE is found. The address field at 'ØØØ6H' contains the value of FBASE and can be used to determine the size of available memory, assuming the CCP is being overlayed by a transient program.

The transient program may use the CP/M I/O facilities to communicate with the operator's console and peripheral devices, including the drive subsystem. The I/O system is accessed by passing a 'function number' and an 'information address' to CP/M through the FDOS entry point at location ØØØ5H. In the case of a disk read, for example, the transient program sends the number corresponding to a disk read, along with the address of an FCB to the CP/M FDOS. The FDOS, in turn, performs the operation and returns with either a disk read completion indication or an error number indicating that the disk read was unsuccessful.

All disk-dependent portions of CP/M are placed into a BIOS-resident 'Disk Parameter Block'. The end user need only specify the maximum number of active disks, the starting and ending sector numbers, the data allocation size, the maximum extent of the logical disk, directory size information, and reserved track values. The macros use this information to generate the appropriate tables and table references for use during CP/M operation. Use of these subroutines, together with the table driven data access algorithms, make CP/M truly a universal data management system.

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 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 overlayed; therefore, the program need only branch to the bootstrap loader at the end of execution, and the CP/M user interface is reloaded from system memory. The result of overlaying the BIOS area is a direct return to the EXECUTIVE monitor; a warm boot is not possible without the BIOS, thereby making a cold boot necessary to return to the VECTOR 4 CP/M operating system.

All commands to CP/M from a program are accomplished through 'system calls' or routines within CP/M which will perform specific low level functions, such as getting a character from the keyboard or displaying text on the terminal. System calls are used in the program whenever control of the computer is passed to CP/M to accomplish a specific task. Control is then returned to the program by CP/M once that task has been executed.

System calls are executed by an instruction passed to the CP/M entry point, which is contained in the reserved memory area below Ø1ØØH. The entry point is located at ØØØ5H and is a 'JMP' instruction to the actual entry point in the BDOS; however, because CP/M resides in the top portion of memory, the actual entry point Maybe different for each system.

When a program passes a system call, the most common place where information can be passed is in the internal registers of the CPU; however, if the amount of information that needs to be passed can't be stored in the registers, the protocol specifies that a block of memory be set up with the information in it and the address of that block of memory be passed in the registers.

The first piece of information passed to CP/M from the program is the specific call needed to execute a particular function. Each system call is assigned a number, which is placed in register C prior to executing the instruction to get CP/M. CP/M will then read the C register, determine which call was requested, and execute that function. Often information, in addition to the system call number, needs to be passed to CP/M; e.g., to change the beginning memory address of the

disk I/O buffer which CP/M uses for all disk read/write operations, a 16-bit address needs to be passed to CP/M as well as the system call number. Generally, 8-bit values, such as a character to be output to the terminal, are passed from CP/M to the calling program in the Accumulator (register A), and to CP/M in register E. The 16-bit values, however, such as addresses, are passed in register DE.

This section will detail the protocol which the programmer must follow in order to pass interface information between the calling program and the VECTOR 4 CP/M operating system. It will detail the file control structure on disk and the logical mechanisms CP/M uses to manage the files and disk space. In this section are detailed descriptions of each operating system call (BDOS functions), with the entry parameters and the values that are returned to the calling program.

4.2 VECTOR 4 CP/M FILE STRUCTURE

A. FILENAMES

VECTOR 4 CP/M organizes the files on a disk with particular regard for the names that have been assigned to the files. The filenames are used to build a directory, which is used in accessing files where each floppy diskette or each logical hard disk drive has its own directory. CP/M stores the directory beginning at the first track and sector following the reserved system tracks.

The disk filenames consist of three parts: the drive select code, the filename consisting of from one to eight non-blank characters, and the extension consisting of zero to three non-blank characters. The extension identifies the generic category of a particular file while the filename distinguishes individual files in each category. The extensions listed below in Exhibit 4-1 name a few generic categories which have been established, although they are generally arbitrary:

Exhibit 4-1. GENERIC EXTENSIONS

ASM	Assembler Source	PLI	PL/I Source File
PRN	Printer Listing	OVL	PL/I Overlay file
HEX	Hex Machine Code	DAT	Program data file
BAS	Basic Source File	BAK	Backup file
SYM	Symbol File	LNK	PLINK Link command file
COM	CCP Command File	REL	Relocation Module
$$$	Temporary File	SUB	Submit file
MAP	PLINK Map file (compiled BASIC)		

B. RECORDS

While tracks and sectors divide a disk physically, records, blocks or 'allocation blocks' and extents are the units CP/M uses to organize files logically. Information is transferred to and from a disk by moving it on a block basis; that is, by moving 128-byte blocks of data at a time. CP/M refers to these 128-byte blocks as 'records'.

A distinct feature of CP/M is that files can be any size from 0 to 8 megabytes. CP/M allows file compaction or expansion in size. CP/M makes this elasticity possible by managing logical storage units called 'allocation blocks'.

C. ALLOCATION BLOCKS

The size of an allocation block varies depending on the type of disk being used. Exhibit 4-2 describes the allocation blocks for each of the disks currently being used in Vector systems. CP/M numbers the allocation blocks consecutively beginning with the first track and sector, following the reserved system tracks, and continuing until the last sector on the disk has been numbered. As different disks reserve a different number of system tracks, the track and sector where the directory begins will vary; however, the reserved system tracks will always be the first tracks on the disk and the directory will follow on the next track(s).

As a file gets larger, CP/M assigns new allocation blocks for it, and as it gets smaller, or is deleted, CP/M reclaims allocation blocks. In this fashion, the operating system can always account for the space available on disk and a file's contents can be spread across the disk in randomly located sectors. The records may be considered logically adjacent but they may not be physically adjacent on disk.

4-2. ALLOCATION PARAMETER TABLE

TYPE OF DISK USED	RECORDS PER TRACK	RECORDS PER SECTOR	RESERVED TRACKS	RECORDS PER ALLOCATION BLOCK	SECTORS PER ALLOCATION BLOCK
5 1/4" Single Sided Floppy	32	2	2	16	8
5 1/4" Double Sided Floppy	32	2	4	16	8
5 1/4" Hard Disk	64	2	4	32	16

NOTE: All Vector 4 disks have 256-byte sectors. Even numbered records are stored in the first half of the sector and odd numbered records are stored in the second half of the sector.

D. EXTENTS

The final unit of measure needed to complete the file structure is called an extent. A logical extent is equal to 128 records, or 16K of memory.

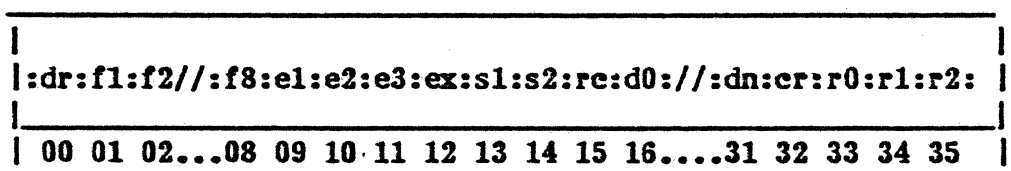
Files in CP/M can be thought of as a sequence of up to 65,536 records of 128 bytes each, numbered from 0 through 65,535, or they can be thought of as 512 logical extents of 16K each. Internally, all files are broken into logical extents so that counters are easily maintained as 8-bit values. However, the decomposition into extents is of no particular consequence to the programmer since each extent is automatically accessed in both sequential and random access modes. The important point to remember is that CP/M files can be as large as 8 megabytes in length.

4.3 FILE CONTROL BLOCKS

Earlier it was noted that the filenames on disk are used to construct a directory for each drive. The directory contains an entry for each extent of each file. Using the filename as a starting point, CP/M constructs a File Control Block (FCB) which contains pointers to the data on disk. The FCB consists of a sequence of 33 bytes for sequential access and a series of 36 bytes in the case that the file is to be accessed randomly.

An example of the File Control Block format appears in Exhibit 4-3 below. Each field of the FCB must be filled in properly before a disk I/O system call is issued. Generally, it is the programmer's responsibility to fill in the lower 16 bytes and initialize the 'cr' field. Normally, bytes 1 through 11 are set to the ASCII character values for the filename and extension, while all other fields are zero.

Exhibit 4-3. FILE CONTROL BLOCK FORMAT



Where:

dr — drive code (Ø-16)
Ø=> use the default drive for the file
1=> auto disk select drive A
2=> auto disk select drive B
...
16=> auto disk select drive P

f1...f8 — Contain the filename in ASCII

e1, e2, e3 — Contain the extension in ASCII, with high bit = Ø; e1', e2', e3' denote the high bit (7) of these positions,
e1' = 1 => Read/Only file,
e2' = 1 => SYS file, no DIR list

ex — Contains the current extent number, which is normally set to ØØ by the programmer but can be in the range Ø-31 during file input and output (5 bits).

s1	Reserved for internal use. Contains MOD number- 4 bits which when combined with 'ex' provides a 9 bit value that specifies one of 5 possible extents.
s2	Reserved for internal use. Set to 8ØH on call to OPEN, MAKE or SEARCH.
rc	Record count beginning of extent 'ex', takes on a value from Ø-128 (128 records x 128 bytes/record = 1 extent = 1 block).
dØ...dn	Disk allocation map reserved for system use and maintained by CP/M; contains the allocation block numbers.
cr	Current record to read or write in a sequential file operation, normally set to zero by the programmer.
rØ, r1, r2	Optional random record number in the range Ø-65,535 with overflow to r2. RØ, r1 constitute a 16-bit value with low byte rØ and high byte r1.

On using the OPEN or MAKE function and providing a blank FCB containing only an ufn, the BDOS will scan the directory of the drive that is indicated and match the user supplied ufn with a filename in the disk directory. The BDOS will then complete the FCB in memory, retrieving the allocation pointers from the FCB on disk. The Directory structure is the same as the FCB structure, except taht the first byte consists of the user area (Ø-16) and the structure only includes the first 32 bytes.

CP/M will update the allocation block map in the FCB in memory as new data is written to the file or as data is erased from the file. When file updating has been completed, the new FCB must be written to the disk using the CLOSE system call. Remember that once a file has changed size, it must always be closed or the new revision will be lost.

The FCB may be located anywhere in memory. When making file related system call operations the address of the FCB is passed in register DE. Transient programs often use the default FCB area reserved by CP/M at location ØØ5CH for simple file operations. This default location can be used for random access files, since the three bytes starting at location ØØ7DH are available for this purpose. Often, transient programs will specify two filenames for operation. The CCP constructs the first sixteen bytes of two optional FCBs for a transient by scanning the remainder of the line following the transient command name. The first FCB is constructed at location ØØ5CH, and can be used 'as-is' for subsequent file operations. The second FCB occupies the dØ...dn portion of the first FCB, and must be moved to another area of memory before use.

If, for example, the user wants to copy a file and change the name using the PIP program, he would enter:

PIP B:DATA.MEM=INPUT.DAT

the file PIP.COM is loaded into the TPA, and the default FCB at location ØØ5CH is initialized to drive code 2, filename 'DATA and extension 'MEM'. All remaining fields through 'cr' are set to zero. If two separate filenames were given, the second would be automatically parsed and placed at location ØØ6CH. Note that it is the programmer's responsibility to move this second filename and extension to another area, usually a separate FCB, before opening the file which begins at location ØØ5CH, because the open operation will overwrite the second filename and extension.

If no filenames are specified in the original command, the fields beginning at location ØØ5DH and ØØ6DH will contain blanks. In all cases, the CCP translates lower case alphabetics to upper case to be consistent with the CCP file naming conventions. As an added convenience, the default buffer area at location ØØ8ØH is initialized to the command line entered by the programmer following the program name. The first position contains the number of characters, with the characters themselves following the character count. Given the above command line, the area beginning at location ØØ8ØH is initialized as follows:

Location ØØ8ØH

+00	+01	+02	+03	+04	+05	+06	+07	+08	+09	+0A	+0B	+0C	+0D	+0E	...
15	"	"	"B"	":"	"D"	"A"	"T"	"A"	"."	"M"	"E"	"M"	"="	"I"	"N"	...

where the characters are translated to upper case ASCII with uninitialized memory following the last valid character. Again, it is the responsibility of the programmer to extract the information from this buffer before any file operations are performed, unless the default DMA address is explicitly changed.

A 128-byte record default location for disk I/O is provided by VECTOR 4 CP/M at location ØØ8ØH, which is the initial default DMA address (see function 26). If either the SEARCH FIRST (function 17) or SEARCH NEXT (function 18) is successful in finding a user specified file, it will return a directory record (consisting of four directory entries) at the DMA address, destroying any information residing there.

CP/M treats ASCII files as a sequence of ASCII characters, where each 'line' of the source file is followed by a [RETURN] line-feed sequence (ØDH followed by ØAH). Therefore, one 128-byte CP/M record could contain several lines of source text. The end of an ASCII file is denoted by a [CTRL Z] character (1AH) or a real end-of-file, returned by the CP/M read operation.

[CTRL Z] characters embedded within machine code files (e.g., COM files) are ignored, however, and the end-of-file condition returned by CP/M is used to terminate read operations.

Memory located from '0000H to 00FFH is called the 'base page'. The base page contains several segments of code and data that are used by transient programs while running under VECTOR 4 CP/M; the code and data areas are shown for reference in Exhibit 4-4. Addresses are relative to the beginning of the memory segment.

4-4. BASE PAGE AREAS

Address	Contents
0000H-0002H	Contains jump instruction to the BIOS termination entry point at BIOS + 3 (warm boot).
0003H	Contains the IOBYTE.
0004H	High-order nybl = User Number Low-order nybl = Drive Number
0005H-0007H	Contains jump instructions to the BDOS: JMP 0005H provides primary entry point to the BDOS, LHLD 0006H places the jump instruction address field into register HL. This value, minus one, equals the highest address of memory that is available to the program. **NOTE:** DDT and RAID programs will change this address field to reflect the reduced memory size.
0008H-0037H	Reserved Restart locations 1 - 6.
0038H-003FH	Restart 7 - Contains a jump instruction into the DDT, when running in debug mode for programmed breakpoints; otherwise not used by VECTOR 4 CP/M.
0040H-005BH	Reserved - not currently used.
005CH-007BH	Default FCB (area 1) initialized by CCP for a program from existing first command-tail operand of command line.

ØØ6CH-ØØ7BH	Default FCB (area 2) initialized by CCP for a program from existing second command-tail operand of command line. This area will overlay the last 16 bytes of default FCB (area 1); to use this area's information, the program must copy it to another location before using area 1.
ØØ7CH	Contains current record position of default FCB (area 1); this field is used with area 1 in processing sequential records.
ØØ7DH-ØØ7FH	Contains optional default random record position; this field is an extension of area 1 used in processing random records.
ØØ8ØH-ØØFFH	Default disk buffer (128 bytes); filled with command-tail when the CCP loads a program.

4.4 OPERATING SYSTEM CALLS

4.4.1 General

An operating system call (BDOS function) instructs VECTOR 4 CP/M to perform specific functions needed to interface a program with the entry, output, and storage devices. Generally, the operating calls govern input and output to peripherals such as the keyboard or the printer, or they interface the programmer's instructions with the file structure on disk. When such a call is used as part of a program, control of the computer is passed from the program to CP/M. CP/M runs the specific routine which was requested, returns the result of the operation, and gives control of the computer back to the program at the line where the call was initiated. Information is passed between the calling program and the CP/M operating system by moving values to and from the internal registers of the CPU. If the amount of information to be passed cannot be stored in the registers, protocol specifies that a block of memory be established with the information in it and the address of that block be passed in the registers.

A call to the CP/M operating system must include a function number which specifies the operation being required, and may include a single-byte word such as an ASCII character or a double-byte word such as an information address.

The Z80B CPU in your Vector 4 System has 22 principal registers including twin register sets comprised of an Accumulator, (Register A), a Status register, and six 8-bit general purpose registers, B, C, D, E, H and L.

The first piece of information to be passed is the function number for the specific routine that is being requested; the function number is always passed in register C. Single-byte values (8-bit values) are passed to CP/M in register E and returned in the Accumulator. Double-byte values are passed to CP/M in the double-byte pair DE and returned in registers HL.

NOTE: For reasons of compatibility, register A = L and register B = H upon return in all cases. Note that the register passing conventions of CP/M agree with those of Intel's PL/M systems programming language. If a call to a function within the BDOS is not implemented under the current version of VECTOR 4 CP/M, the C register will contain 'FF' on return.

Upon entry to a transient program, the CCP leaves the stack pointer set to an eight level stack area with the CCP return address pushed onto the stack, leaving seven levels before overflow occurs. Although this stack is usually not used by a transient program (i.e., most transients return to the CCP through a jump to location ØØØØH), it is sufficiently large enough to make CP/M system calls since the FDOS switches to a local stack at system entry.
The following assembly language program segment, for example, reads characters continuously until an asterisk is encountered, at which time control returns to the CCP.

```
BDOS  EQU    0005H     ;STANDARD CP/M ENTRY
CONIN EQU    1         ;CONSOLE INPUT FUNCTION
;
      ORG    0100H     ;BASE OF TPA
NEXTC:MVI    C,CONIN   ;READ NEXT CHARACTER
      CALL   BDOS      ;RETURN CHARACTER IN <A>
      CPI    '*'       ;END OF PROCESSING?
      JNZ    NEXTC     ;LOOP IF NOT
      RET              ;RETURN TO CCP
      END
```

4.4.2 Functional Description

The function number and a short description of each system call follows Exhibit 4-5 at the end of this section shows a summary chart for all functions:

FUNCTION 0: SYSTEM RESET

Entry Parameters: Register C - ØØH

The SYSTEM RESET function will return control to the VECTOR 4 CP/M operating system at the CCP level. All records and files locked by the calling program are released. The CCP reinitializes the disk subsystem by selecting and logging-in Drive A. To this particular process, this has exactly the same effect as a freshly booted system.

FUNCTION 1: CONSOLE INPUT

Entry Parameters: Register C - Ø1H

Returned Value: Register A - ASCII Character

The CONSOLE INPUT function will read the next console character to register A. Graphic characters, along with carriage return, line feed, and backspace, [CTRL H], are echoed to the console. The FDOS does not return to the calling program until a character has been typed, thus suspending execution if a character is not ready.

FUNCTION 2: CONSOLE OUTPUT

Entry Parameters: Register C - Ø2H
Register E - ASCII Character

The CONSOLE OUTPUT function will send the ASCII character from register E to the console device. Similar to function 1, tabs are expanded and checks are made for terminate process, start/stop scroll and printer echo.

FUNCTION 3: READER INPUT

Entry Parameters: Register C - Ø3H

Returned Value: Register A - ASCII Character

The READER INPUT function will read the next character from the logical reader into register A. Control does not return until the character has been read. This function (along with function 4) are included even though it is realized that the Vector 4 may never be connected to a READER or PUNCH device. These functions can be used "logically" to connect other types of devices to the Vector 4.

FUNCTION 4: PUNCH OUTPUT

Entry Parameters: Register C - Ø4H
Register E - ASCII Character

The Punch Output function will send the character from register E to the logical punch device.

FUNCTION 5: LIST OUTPUT

Entry Parameters: Register C - Ø5H
Register E - ASCII Character

The LIST OUTPUT function will send the ASCII character in register E to the selected logical listing device. In the event that the spooler is being used and a disk error occurs, the function returns ØFFH in register A, otherwise a Ø is returned in register A.

FUNCTION 6: DIRECT CONSOLE I/O

Entry Parameters: Register C - Ø6H
Register E - ØFFH - Console Input/Status
ØFEH - Keyboard Status
ØFDH - Input
ASCII character (Output)

Returned Value: Register A - Status or ASCII Character (No Value)

The DIRECT CONSOLE I/O function is supported under CP/M for those specialized applications where unadorned console input and output is required. Use of this function should, in general, be avoided since it bypasses all of CP/M's normal control character functions (e.g., [CTRL S], [CTRL X]). Programs which perform direct I/O through the BIOS under previous releases of CP/M, however, should be changed to use direct I/O under the BDOS so that they can be fully supported under future releases of CP/M.

Upon entry to function 6, register E either contains a hexadecimal value FDH - FFH, denoting a console input request, or an ASCII character. If the input value is not in the range ØFDH - ØFFH, function 6 will output the character in the E register to the system console.

REGISTER E:	MEANING:
ØFFH	Console Input and status: register A = input character or if no character is ready, zero is returned.
ØFEH	Console status: register A = ØØ if no character is ready or FF if a character is available at the console.
ØFDH	Console input: returns an input character in register A and will suspend the calling process until a character is ready.
ASCII	Console output: Sends value in register E to the console. No value returned in register A.

FUNCTION 7: GET IOBYTE

Entry Parameters: Register C - Ø7H

Returned Value: Register A - IOBYTE Value

The GET IOBYTE function will return the current value of the IOBYTE in register A.

FUNCTION 8: SET IOBYTE

Entry Parameters: Register C - Ø8H
Register E - New IOBYTE value

Returned Value: Register A - Return Code

The two most significant bits of the IOBYTE are used to define the destination of the list output stream. A value of 'Ø' for the two bits specifies that all list output is to be discarded. A value of '1' selects echo to console and values of '2' and '3' select list device one and list device two, respectively.

All changing of the IOBYTE should be done using this function. The least significant 6 bits are reserved for future use and should be maintained when changing the list device. This can be accomplished by first getting the IOBYTE (see function 7) and making only the necessary changes and then performing function 8.

In the event that the user attempts to select a printer which is busy (the printer is being used by the despooler or by another process), the IOBYTE will not be altered and an ØFFH will be returned in the A register; in other cases the A register will return a ØØH.

FUNCTION 9: PRINT STRING

Entry Parameters: Register C - Ø9H
Registers DE - String Address

The PRINT STRING function will send the character string stored in memory at the location given by DE to the console device, until a '$' is encountered in the string. Tabs are expanded as in function 2, and checks are made for terminate process, start/stop scroll, and printer echo. To print a '$' character, set bit 7 of the character and the '$' will not be recognized as an end of string delimeter.

FUNCTION 10: READ CONSOLE BUFFER

Entry Parameters: Register C - ØAH
Register DE - Buffer Address

Returned Value: Console Characters in Buffer

The READ BUFFER function will read a line of edited console input into a buffer addressed by register DE. Console input is terminated when the input buffer overflows. The Read Buffer takes the form:

DE:	+0	+1	+2	+3	+4	+5	+6	+7	+8	. . .	+n
	:mx	:nc	:c1	:c2	:c3	:c4	:c5	:c6	:c7	. . .	:??:

where 'mx' is the maximum number of characters which the buffer will hold (1 to 255), 'nc' is the number of characters read (set by FDOS upon return), followed by the characters read from the console. If the number of characters read did not fill the buffer to capacity, (nc is less than mx), then uninitialized positions follow the last character, denoted by '??' in the above figure. A number of control functions are recognized during line editing:

[CTRL C] Reboots when at the beginning of line
[CTRL E] Causes physical end of line
[CTRL H] Backspaces one character position
[CTRL J] (Line feed) terminates input line
[CTRL M] (Return) terminates input line

[CTRL R] Retypes the current line after new line
[CTRL U] Removes current line after new line
[CTRL X] Backspaces to beginning of current line
[CTRL K] Form feeds current list device
[CTRL P] Toggles printer echo
[DEL] Same as [CTRL H or [BACKSPACE]

Note also that certain functions which return the carriage to the leftmost position (e.g., [CTRL X]) do so only to the column position where the prompt ended (in earlier releases, the carriage returned to the extreme left margin). This convention makes operator data input and line correction more legible. The input string will be terminated by a null.

FUNCTION 11: GET CONSOLE STATUS

Entry Parameters: Register C - ØBH

Returned Value: Register A - Console Status

The CONSOLE STATUS function will check to see if a character has been entered at the console. If a character is ready, the value ØFFH is returned in register A; otherwise, a ØØH value is returned.

FUNCTION 12: RETURN VERSION NUMBER

Entry Parameters: Register C - ØCH

Returned Value: Registers HL - Version Number

The RETURN VERSION NUMBER function will provide information which allows version independent programming. A two-byte value is returned, L = ØØ, for all releases previous to Version 2.0. CP/M 2.0 returns a hexadecimal '2Ø' in register L, with subsequent Version 2.0 releases in the hexadecimal range 21 and 22 through 2F. VECTOR 4 CP/M will return a value greater than or equal to '25H'.

Using function 12, for example, application programs can be written which provide both sequential and random access functions, with random access disabled when operating under early releases of CP/M.

FUNCTION 13: RESET DISK SYSTEM

Entry Parameters: Register C - ØDH

Returned Value: Register A - Return Code
Register H - Physical or Logical Error

The RESET DISK SYSTEM function will initialize the BDOS, reset the Read/Write state for all disks, selects Drive A, and sets the default DMA to 8ØH. Normally, it returns ØØH in register A; however, if a SUBMIT file is open on the drive, this function returns ØFFH in register A. If a physical error is returned during drive select, register H will return the physical error code (see Appendix A for a discussion of errors).

FUNCTION 14: SELECT DISK

Entry Parameters: Register C - ØEH
Register E - Selected Disk

Returned Value: Register A - Return Code
Register H - Physical or Logical Error

The SELECT DISK function will designate a drive as the default drive for subsequent file operations, with the value in register E = Ø for Drive A, 1 for Drive B, etc., through 15 corresponding to Drive P in a full sixteen drive system.

The drive is placed in an 'on-line' status which, in particular, activates its directory until the next cold start, warm start, or disk system reset operation. If the disk media is changed while it is on-line, the drive automatically goes to a Read-Only status in a standard CP/M environment (see function 28). FCBs which specify drive code zero (dr = ØØH) automatically reference the currently selected default drive. FCBs with drive code values between 1 and 16, however, ignore the selected default drive and directly reference Drives A through P.

If the SELECT DISK operation was successful, register A is zero upon return. If a physical error is returned, register A is ØFFH and one of the following physical error codes is returned in register H:

Ø1 - Permanent error
Ø4 - Select error

(See Appendix A for description of error modes)

FUNCTION 15: OPEN FILE

Entry Parameters: Register C - ØFH
Register DE - FCB address

Returned Value: Register A - Directory Code
Register H - Physical or Logical Error

The OPEN FILE function is used to activate a file which has already been written to the disk for the currently active user number. The BDOS scans the disk directory for a match in positions 1 though 14 with the FCB referenced by register DE. Note that if a file is to be accessed sequentially starting at record Ø, the current record byte ('cr') must be zeroed (byte s1 is automatically zeroed). Normally, no question marks are included and, further, bytes 'ex' and 's2' of the FCB are zero. An error will occur if any field in the filename contains a '?'.

If a directory entry is matched, the relevant directory information is copied into bytes dØ through dn of the FCB, thereby allowing subsequent file access in read or write operations. At the same time, an FCB checksum is computed and is used to verify that the file hasn't been changed by another process or task during subsequent operations.

Attribute bits f5' and f6' of the filename in the FCB specify in which of three modes the file will be opened:

f6':	f5':	MODE:
0	0	LOCKED MODE (DEFAULT)
0	1	UNLOCKED MODE
1	0 or 1	READ-ONLY MODE

Opening files in the Locked mode prevents other tasks from accessing the same files.

Opening files in the Unlocked mode allows other tasks to access the same file, unless the user locks specific records which then prevents other tasks from accessing those records of the file.

Opening files in the Read-Only mode allows only read operations. The Read-Only mode allows access for other tasks to the file provided that they also open it in the Read-Only mode.

When an open is successful, the opened file is registered in a File Lock list. While this file remains in the list, no other task can perform any operations on the file in any mode other than the mode specified by the current task. Also, no other task can delete, rename or modify any of the file's attributes. The file remains in the lock list until permanently closed or the task that opened it terminates.

Note that an existing file must not be accessed until a successful open operation has been completed.

The OPEN function returns a 'directory code' with a value of Ø through 3 if the OPEN was successful, or ØFFH (255 decimal) if the file cannot be found. Register H is set to zero in both of these cases but if a physical or logical error is returned, register H contains one of the following error codes:

Ø1 - PERMANENT ERROR
Ø4 - SELECT ERROR
Ø5 - FILE OPEN BY ANOTHER PROCESS OR BY CURRENT PROCESS IN AN INCOMPATIBLE MODE
Ø9 - '?' IN FILENAME OR EXTENSION FIELD
1Ø - PROCESS OPEN FILE LIMIT EXCEEDED
11 - NO ROOM IN THE SYSTEM LOCK LIST

(See Appendix A for Error messages)

FUNCTION 16: CLOSE FILE

Entry Parameters: Register C - 1ØH
Register DE - FCB address

Returned Value: Register A - Directory Code
Register H - Physical/Logical Error
Registers DE - FCB Address

The CLOSE FILE function will perform the inverse function of OPEN FILE. Given that the FCB addressed by DE has been previously activated through an OPEN or MAKE function (see functions 15 and 22), CLOSE will write the new FCB to the referenced disk directory. The FCB matching process for the CLOSE is identical to the OPEN function. Interface attribute f5' is used to specifies the mode in which the file is to be closed:

f5'	MODE
Ø	Permanent close (default)
1	Partial close

The CLOSE FILE function will first verify that the referenced FCB has a valid checksum; if it is valid and the referenced FCB contains new information because of write operations to the FCB, the function permanently records the new information in the referenced disk directory.

The FCB will not contain new information and the directory update step is bypassed if only READ and/or UPDATE operations have been made to the FCB. The function, however, always attempts to locate the corresponding FCB entry in the directory and will return an error if the entry cannot be found.

If the CLOSE has performed successfully and is permanent, the function removes the file's item from the system lock list; if the FCB was opened in unlocked mode, it also deletes all the file's record lock items from the system lock list. Since the file's lock list item has been removed, the function will invalidate the FCB checksum to ensure that the FCB cannot be subsequently used with other BDOS functions requiring an open FCB.

In the event that a partial close is performed, the FCB and directory will be updated as above. The file list information is not removed, however, and the FCB checksum remains valid to allow further file access.

A successful operation returns Ø, 1, 2, or 3 in register A, while ØFFH (255 decimal) is returned if the file name cannot be found in the directory; register H is set to zero in both cases. However, if the file has been written to, it must be closed or the new data will be lost.

Opened files should always be closed to optimize system operations in systems utilitizing file tracking or in systems maintaining active file lists. When a physical error is returned, register A is ØFFH and register H will display one of the following:

Ø1 - PERMANENT ERROR
Ø2 - READ-ONLY DISK
Ø4 - SELECT ERROR
Ø6 - FCB CHECKSUM ERROR

FUNCTION 17: SEARCH FOR FIRST

Entry Parameters: Register C - 11H
Register DE - FCB Address

Returned Value: Register A - Directory Code
Register H - Physical or Logical Code

The SEARCH FIRST function will scan the directory for a match with the file given by the FCB addressed by DE. If the file is found, the current DMA address is filled with the record containing the directory entry, and the relative starting position is A * 32 (i.e., rotate the register A left 5 bits, or ADD 'A' five times). Although not normally required for application programs, the directory information can be extracted from the buffer at this position.

An ASCII question mark (63 decimal, 3F hexadecimal) in any position from 'f1' through 'ex' matches the corresponding field of any directory entry on the default or auto-selected disk drive. If the 'dr' field contains an ASCII question mark, then the auto disk select function is disabled, the default disk is searched, with the search function returning any matched entry, allocated or free, belonging to any user number. This latter function is not normally used by application programs, but does allow complete flexibility to scan all current directory values. If the 'dr' field is not a question mark, the 's2' byte is automatically zeroed.

The value 255 (hexadecimal FF) is returned if the file is not found, otherwise Ø, 1, 2, or 3 is returned indicating the file is present. In either case, register H is zerd. If a physical or logical error is returned, register A contains ØFFH and register H contains one of the following error codes:

Ø1 - PERMANENT ERROR
Ø4 - SELECT ERROR

FUNCTION 18: SEARCH FOR NEXT

Entry Parameters: Register C - 12H

Returned Value: Register A - Directory Code
Register H - Physical or Logical Error Code

The SEARCH NEXT function will find the next occurence of a match after Function 17 is used. It returns the decimal value of 255 in register A when no more directory items match.

NOTE: Vector has changed SEARCH NEXT so that it will keep track of its last position in the directory. In this way other disk functions (i.e., DELETE file, RENAME file) may be executed in the middle of a search.

If a physical or logical error is returned, register A contains ØFFH and register H contains one of the following error codes:

Ø1 - PERMANENT ERROR
Ø4 - SELECT ERROR

FUNCTION 19: DELETE FILE

Entry Parameters: Register C - 13H
Register DE - FCB address

Returned Value: Register A - Directory Code
Register H - Physical or Logical Error

The DELETE FILE function will remove files which match the FCB addressed by register DE. The filename and extension may contain ambiguous references but the drive select code must be unambiguous, as in the SEARCH and SEARCH NEXT functions. An open file can be deleted if opened in locked mode by the same process; however, a checksum error will be returned if a further reference to the file occurs using a BDOS function that requires an open FCB. Open Read-Only or unlocked files cannot be deleted.

DELETE FILE will return a decimal '255' if the referenced file or files cannot be found; otherwise, a value in the range of Ø to 3 is returned if the delete was successful.

In both cases, register H is set to zero. If a physical or logical error is returned, register A contains ØFFH and register H contains one of the following error codes:

Ø1 - PERMANENT ERROR
Ø2 - READ-ONLY DISK
Ø3 - READ-ONLY FILE
Ø4 - SELECT ERROR
Ø5 - FILE OPEN BY ANOTHER PROCESS OR BY CURRENT PROCESS IN AN INCOMPATIBLE MODE

FUNCTION 20: READ SEQUENTIAL

Entry Parameters: Register C - 14H
Register DE - FCB Address

Returned Value: Register A - Error Code
Register H - Physical or Logical Error

The READ SEQUENTIAL function will read the next one to sixteen 128-byte records from a file into memory beginning at the current DMA address. The BDOS function 44 will determine the number of records to be read; the default is one record. The FCB addressed by register DE must already be activated by an OPEN or MAKE function call in order that this function can read the next record from the file into memory at the current DMA address. Each record is read from position 'cr' of the extent, then the 'cr' field is automatically incremented to the next record position. If the 'cr' field overflows, the next logical extent is automatically opened and the 'cr' field is reset to zero, to prepare for the next READ operation. The value ØØH is returned in the A register if the read operation was successful; otherwise, one of the following errors is returned if no data exists at the next record position or if a physical or logical error is returned:

Ø1 - READING UNWRITTEN DATA
Ø8 - RECORD LOCKED BY ANOTHER PROCESS
Ø9 - INVALID FCB
11 - UNLOCKED FILE VERIFICATION ERROR
255 - PHYSICAL ERROR (REFER TO REGISTER H)

PHYSICAL ERROR CODES: (FOUR LEAST SIGNIFICANT BITS OF H)

Ø1 - PERMANENT ERROR
Ø4 - SELECT ERROR

FUNCTION 21: WRITE SEQUENTIAL

Entry Parameters: Register C - 15H
Register DE - FCB Address

Returned Value: Register A - Return Code
Register H - Physical or Logical Error

The WRITE SEQUENTIAL function will write one to sixteen 128-byte records beginning at the current DMA address into the file named by the FCB addressed in register DE. BDOS function 44 (SET MULTI-SECTOR COUNT) will determine the number of 128-byte records that are written; the default is one record. The FCB must have been activated previously by an OPEN or MAKE function call. If the FCB has been activated through an OPEN or MAKE function, the WRITE SEQUENTIAL function will write 128 bytes starting at the current DMA address to the file named by the FCB. The record is placed at position 'cr' of the file, and the 'cr' field is automatically incremented to the next record position. If the 'cr' field overflows then the next logical extent is automatically opened and the 'cr' field is reset to zero. Write operations can take place into an existing file, in which case newly written records overlay those which already exist in the file. The WRITE function will, upon return, set register A to zero if the operation was successful; otherwise, A may contain one of the following error codes. If a physical or logical error is returned, register A will be ØFFH and the error code will consist of one of the following in register H:

Ø1 - WRITE MODE: NO DIRECTORY SPACE
Ø2 - NO AVAILABLE DATA BLOCKS (Not returned in random mode)
Ø8 - RECORD LOCKED BY ANOTHER PROCESS
Ø9 - INVALID FCB
1Ø - FCB CHECKSUM ERROR
11 - UNLOCKED FILE VERIFICATION ERROR
255 - PHYSICAL ERROR (REFER TO REGISTER H)

PHYSICAL ERROR CODES: (LEAST SIGNIFICANT FOUR BITS IN REGISTER H)

Ø1 - PERMANENT ERROR
Ø2 - READ-ONLY DISK
Ø3 - READ-ONLY FILE
Ø4 - SELECT ERROR

FUNCTION 22: MAKE FILE

Entry Parameters: Register C - 16H
Register DE - FCB Address

Returned Value: Register A - Directory Code
Register H - Physical or Logical Error

The MAKE FILE function will create a new file under the current user number; it is used for files that have never been written to disk. The FDOS creates the file and initializes the FCB in memory, as well as on disk, as an empty file. Vector Graphic has changed the original CP/M function to prevent creation of duplicate files. A checksum is computed and assigned to the FCB; BDOS functions that require an open FCB will verify that the FCB checksum is valid before performing their operation. The MAKE function has the side-effect of activating the FCB and thus a subsequent OPEN is unnecessary. (Refer to OPEN FILE function for discussion of attributes.) The READ-ONLY mode attribute bit
(f6) is ignored by the MAKE FILE funcion.

This function returns ØFFH in register A if there is no directory space. If the operation was successful register A will return Ø, 1, 2, or 3. In either case, register H is zero; but if a physical or logical error is returned, register H contains one of the following error codes:

Ø1 - PERMANENT ERROR
Ø4 - SELECT ERROR
Ø5 - FILE OPEN BY ANOTHER PROCESS OR BY CURRENT PROCESS IN AN INCOMPATIBLE MODE
Ø8 - FILE ALREADY EXISTS
Ø9 - '?' IN FILENAME OR EXTENSION FIELD
1Ø - PROCESS OPEN FILE LIMIT EXCEEDED
11 - NO ROOM IN THE SYSTEM LOCK LIST

FUNCTION 23: RENAME FILE

Entry Parameters: Register C - 17H
Register DE - FCB Address

Returned Value: Register A - Directory Code
Register H - Physical or Logical Error

The RENAME FILE function will use the FCB addressed by register DE to change all occurrences of the file named in the first 16 bytes to the file named in the second 16 bytes. The call will check that the filenames specified in the FCB are valid and unambiguous, and that the new filename does not already exist on the drive. The drive code 'dr' at position Ø is used to select the drive, while the drive code for the new filename at position 16 of the FCB is assumed to be zero.

A file can be renamed by a process if the file was opened in lock mode, but if the file is accessed by a function which requires an open FCB, an FCB checksum error will be returned. A file cannot be renamed if opened in Read-Only or Unlocked mode.

Upon return, register A is set to a value between Ø and 3 if the rename was successful, and ØFFH (255 decimal) if the first filename could not be found in the directory scan. In either case register H is zero. If a physical or logical error is returned, register A contains ØFFH and one of the following error codes is returned in H:

Ø1 - PERMANENT ERROR
Ø2 - READ-ONLY DISK
Ø3 - READ-ONLY FILE OR FILE OPEN IN READ-ONLY MODE
Ø4 - SELECT ERROR
Ø5 - FILE OPEN BY ANOTHER PROCESS OR BY CURRENT PROCESS IN AN INCOMPATIBLE MODE
Ø8 - FILE ALREADY EXISTS
Ø9 - '?' IN FILENAME OR EXTENSION FIELD

FUNCTION 24: RETURN LOGIN VECTOR

Entry Parameters: Register C - 18H

Returned Value: Register HL - Login Vector

The RETURN LOGIN VECTOR value returned by CP/M is a 16-bit value in HL, where the least significant bit of L corresponds to the first Drive A, and the high-order bit of H corresponds to the sixteenth drive, labelled P. A 'Ø' bit indicates that the drive is not on-line, while a '1' bit marks a drive that is actively on-line due to an explicit disk drive selection, or an implicit drive select caused by a file operation which specified a non-zero 'dr' field. Note that compatibility is maintained with earlier releases, since registers A and L contain the same values upon return.

FUNCTION 25: RETURN CURRENT DISK

Entry Parameters: Register C - 19H

Returned Value: Register A - Current Disk

The RETURN CURRENT DISK function returns the currently selected default disk number in register A. The disk numbers range from Ø through 15 corresponding to Drives A through P.

FUNCTION 26: SET DMA ADDRESS

Entry Parameters: Register C - 1AH
Register DE - DMA Address

Direct Memory Address (DMA) is often used in connection with disk controllers which directly access the memory of the mainframe computer to transfer data to and from the disk subsystem. VECTOR 4 CP/M uses non-DMA access where the data is transferred through programmed I/O operations.

In VECTOR 4 CP/M, the DMA address has come to mean the address at which the 128-byte data record resides before a disk write and after a disk read. On a cold start, warm start, or disk system reset, the DMA address is automatically set to ØØ8ØH. The SET DMA function, however, can be used to change this default value to address another area of memory where the data records reside. Thus, the DMA address becomes the value specified by DE until it is changed by a subsequent SET DMA function, cold start, warm start, or disk system reset.

FUNCTION 27: GET ADDR (ALLOC)

Entry Parameters: Register C - 1BH

Returned Value: Register HL - ALLOC Address

The System maintains an "allocation vector" in the main memory for each on-line disk drive (Bitmap). Programs like STAT use this table to determine the amount of remaining storage space. Function 27 will return the address of the allocation table for the currently selected drive. If the selected disk has been marked Read-Only, the allocation information may be invalid. This function is not normally used by application programs. Application programs should use function 46.

FUNCTION 28: WRITE PROTECT DISK

Entry Parameters: Register C - 1CH

The WRITE PROTECT DISK function will provide temporary write protection for the currently selected disk. Any attempt to write to the disk, before the next cold or warm start operation produces the message:

ERROR - d: DRIVE IS READ-ONLY

NOTE: A drive can be permanently set to READ-ONLY by using the CONFIG D option.

FUNCTION 29: GET READ/ONLY VECTOR

Entry Parameters: Register C - 1DH

Returned Value: Register HL - R/O Vector Value

The GET READ/ONLY VECTOR function will return a bit vector in register HL which indicates drives which have the temporary read-only bit set. Similar to function 24, the least significant bit corresponds to Drive A, while the most significant bit corresponds to Drive P. The Read-Only bit is set either by an explicit call to function 28, or by the automatic software mechanisms within CP/M which detect changed disks.

FUNCTION 30: SET FILE ATTRIBUTES

Entry Parameters: Register C - 1EH
Register DE - FCB address

Returned Value: Register A - Directory Code
Register H - Permanent or Logical Error

The SET FILE ATTRIBUTES function will set or clear indicators attached to files; in particular, the R-O and System attributes (t1' and t2') can be set or reset. Register DE will address an unambiguous FCB with the appropriate attributes set or reset. Function 30 will search for a match and change the matched directory entry to contain the selected indicators. Indicator f1' is used to make a file 'invisible' to other users (other account nubers); while f2' through f4' are not presently used, but may be useful for applications programs since they are not involved in the matching process during file open and close operations. Indicators f5' and f6' are used during OPEN, MAKE, and CLOSE functions to specify access mode; f7', f8' and t3' are reserved for future system expansion.

This function cannot be performed on any files opened by another process. It can be performed on files opened by the current process in lock mode but any subsequent operations which require an open FCB will return a FCB checksum error. No attributes may be set on any file open in unlocked or read/only mode.

If successful, this function returns a value of Ø through 3 in register A. If the file is not found, register A will return FFH. Register H is zero in either case. If a physical or logical error is returned, register A is FFH and register H contains one of the following:

Ø1 - PERMANENT ERROR
Ø2 - READ-ONLY ERROR
Ø4 - SELECT ERROR
Ø5 - FILE OPEN BY ANOTHER PROCESS
Ø9 - "?" IN FILE NAME OR EXTENSION

FUNCTION 31: GET ADDR (DISK PARMS)

Entry Parameters: Register C - 1FH

Returned Value: Register HL - DPB address

The GET ADDR function will return the address of the BIOS resident Disk Parameter Block (DPB) in register HL. The address is useful for computing space or for changing the disk parameter values when the disk environment changes. Normally, application programs will not require this facility.

FUNCTION 32: SET/GET USER CODE

Entry Parameters: Register C - 2ØH
Register E - ØFFH (GET)
Register E - < FFH User Code (SET) (E MOD 32)

Returned Value: Register A - Current Code (or no value)

The SET/GET USER CODE function will change or query, within an application program, the currently active user number. If register E = ØFFH the value of the current user number is returned in register A, where the value is in the range of Ø to 31. If register E is not ØFFH, then the current user number is changed to the value of E (modulo 32).

FUNCTION 33: READ RANDOM

Entry Parameters: Register C - 21H
Register DE - FCB Address

Returned Value: Register A - Error Code
Register H - Physical or Logical Error

The READ RANDOM function will use the read random field, (the last three bytes) of the FCB to select a particular record number and read the record. The read operation takes place at a record number indicated by the 24-byte value constructed from byte positions rØ at 33, r1 at 34, and r2 at 35. Note that the sequence of 24 bits is stored with the least significant byte first (rØ), the middle byte next (r1), and the high byte last (r2).

VECTOR 4 CP/M does not reference byte r2, except in computing the size of a file (function 35). Byte r2 must be zero because a non-zero value indicates overflow past the end of file; therefore, the rØ, r1 byte pair is treated as a double byte, or 'word' value, which contains the record to read. This value ranges from Ø to 65,535 providing access to any particular record of the 8 Mbyte file.

In order to process a file using random access, the base extent (extent Ø) must first be opened. Although the base extent may or may not contain any allocated data, this ensures that the file is properly recorded in the directory, and is visible in DIR requests. The selected record number is then stored into the random record field (rØ,r1), and the BDOS is called to read the record. If the BDOS Multi-Sector Count is greater than one (see function 44), the READ RANDOM function will read multiple consecutive records into memory beginning at the current DMA. The rØ, r1, and r2 field of the FCB will automatically be incremented to read each record; however, the FCB's random record number is restored to the first record's value upon return to the calling process.
Upon return from the call, register A either contains an error code or the value ØØH indicating the operation was successful, in which case the current DMA address contains the randomly accessed record or multiple records if using function 44 (see paragraph on function 34). If register A is non-zero, one of the following error codes will be returned:

Ø1 - READING UNWRITTEN DATA
Ø3 - CANNOT CLOSE CURRENT EXTENT
Ø4 - SEEK TO UNWRITTEN EXTENT
Ø6 - SEEK PAST PHYSICAL END OF DISK (RANDOM RECORD NUMBER OUT OF RANGE)
Ø8 - RECORD LOCKED BY ANOTHER PROCESS
1Ø - FCB CHECKSUM ERROR
11 - UNLOCKED FILE VERIFICATION ERROR
255 - PHYSICAL ERROR (REFER TO REGISTER H)

If a physical error is returned, register H will contain one of the following error codes:

Ø1 - PERMANENT ERROR
Ø4 - SELECT ERROR

NOTE: If a physical error is returned in the RANDOM READ function, the four MSBs of register H contain an integer set to the number of records successfully read before an error was encountered. Note that contrary to the SEQUENTIAL READ operation, the record number is not advanced; therefore, subsequent RANDOM READ operations continue to read the same record.

Upon each RANDOM READ operation, the logical extent and current record values are automatically set; therefore, the file can be sequentially read or written, starting from the position from which it was accessed randomly. Note, however, that in this case, the last randomly read record will be read again as you switch from random mode to sequential read, and the last record will be written again as you switch to a sequential write operation. You can, of course, simply advance the random record position following each random read or write to obtain the effect of a sequential I/O operation.

FUNCTION 34: WRITE RANDOM

Entry Parameters: Register C - 22H
Registers DE - FCB Address

Returned Value: Register A - Error Code
Register H - Physical or Logical Error

The WRITE RANDOM operation works in nearly the same fashion as the READ RANDOM call, except that data is written to the disk from the current DMA address. Further, if the file's space has yet to be allocated, the function allocates space before writing. As in the READ RANDOM operation, the random record number is not advanced as a result of the write. The logical extent number and current record positions of the FCB are set to correspond with the random record being written.

SEQUENTIAL READ or WRITE operations can commence following a RANDOM WRITE, with the notation that the currently addressed record is either READ or WRITE again as the sequential operation begins. Simply advance the random record position following each write to get the effect of a sequential write operation. Note that reading or writing the last record of an extent in random mode does not cause an automatic extent switch as it does in sequential mode.

The error codes returned by a random write are identical to the RANDOM READ operation with the addition of error code Ø5, which indicates that a new extent cannot be created due to directory overflow.

To write to a file using the WRITE RANDOM function, the calling program must first open the base extent (extent Ø). This ensures that the FCB is initialized properly for subsequent random access operations.

The base extent may or may not contain any allocated data, however, opening extent Ø records the file in the directory so that it can be displayed by the DIR utility (if a process does not open extent Ø and allocate data to some other extent, the file will be invisible to the DIR utility).

The WRITE RANDOM function will set the logical extent and current record positions to correspond with the random record being written, but does not change the random record number. Therefore, sequential read or write operations can follow a random write, with the current record being reread or rewritten as the calling process switches from random to sequential mode.

If the BDOS Multi-Sector Count is greater than one (see function 44), the WRITE RANDOM function will write multiple consecutive records from memory beginning at the current DMA. The rØ, r1, and r2 field of the FCB will automatically be incremented to write each record; however, the FCB's random record number is restored to the first record's value upon return to the calling process. Upon return, the WRITE RANDOM function will set register A to zero if the write operation was successful; otherwise, register A will contain one of the following error codes:

Ø2 - NO AVAILABLE DATA BLOCKS (Not returned in random mode)
Ø3 - CANNOT CLOSE CURRENT EXTENT
Ø5 - NO AVAILABLE DIRECTORY SPACE (WRITE MODE ONLY)
Ø6 - SEEK PAST PHYSICAL END OF DISK (RANDOM RECORD NUMBER OUT OF RANGE)
Ø8 - RECORD LOCKED BY ANOTHER PROCESS
1Ø - FCB CHECKSUM ERROR
11 - UNLOCKED FILE VERIFICATION ERROR
255 - PHYSICAL ERROR (REFER TO REGISTER H)

If a physical error is returned, the four LSBs of register H will contain one of the following error codes:

Ø1 - PERMANENT ERROR
Ø2 - READ-ONLY DISK
Ø3 - READ-ONLY FILE OR FILE OPEN IN READ-ONLY MODE
Ø4 - SELECT ERROR

NOTE: If a physical error is returned in the RANDOM WRITE function, the four MSBs of register H contain an integer set to the number of records successfully written before an error was encountered.

FUNCTION 35: COMPUTE FILE SIZE

Entry Parameters: Register C - 23H
Register DE - FCB address

Returned Value: Random Record Field Set
Register A - Error code
Register H - Physical or Logical Error

When computing the size of a file, register DE will address an FCB in random mode format (bytes rØ, r1, and r2 are present). The FCB contains an unambiguous filename which is used in the directory scan. Upon return, the random record bytes contain the 'virtual' file size which is, in effect, the record address of the record following the end of the file.

If, following a call to function 35, the high record byte 'r2' is Ø1, the file will contain the maximum record count of 65,536. Otherwise, bytes rØ and r1 constitute a 16-bit value (rØ is the least significant byte, as before) which is the file size.

Data can be appended to an existing file by using this information to SET the RANDOM RECORD position before performing a series of RANDOM WRITE operations.

The virtual size of a file corresponds to the physical size when the file is written sequentially. If, instead, the file was created in random mode and 'holes' exist in the allocation, the file may, in fact, contain fewer records than the size indicates. If, for example, only the last record of an eight megabyte file is written in random mode (i.e., record number 65535), the virtual size will be 65,536 records, although only one block of data is actually allocated.

On return, register A is set to zero if the specified file was found, or ØFFH if the file was not found; in either case, register H is set to zero. If a physical error is returned, register A contains 'ØFFH' and register H will contain one of the following error codes:

Ø1 - PERMANENT ERROR
Ø4 - SELECT ERROR
Ø9 - '?' IN FILENAME OR EXTENSION FIELD

FUNCTION 36: SET RANDOM RECORD

Entry Parameters: Register C - 24H
Register DE - FCB address

Returned Value: Random Record Field Set

The SET RANDOM RECORD function will cause the BDOS to automatically produce the random record position from a file which has been read or written sequentially to a particular point. The function can be useful in two ways.

First, it is often necessary to initially read and scan a sequential file to extract the positions of various 'key' fields. As each key is encountered, function 36 is called to compute the random record position for the data corresponding to this key. If the data unit size is 128 bytes, the resulting record position is placed into a table with the key for later retrieval. After scanning the entire file and tabularizing the keys and their record numbers, you can move instantly to a particular keyed record by performing a random read using the corresponding random record number which was saved earlier. The scheme is easily generalized when variable record lengths are involved since the program need only store the buffer-relative byte position along with the key and record number in order to find the exact starting position of the keyed data at a later time.

Second, the use of function 36 occurs when switching from a sequential read or write over to random read or write. A file is sequentially accessed to a particular point in the file, function 36 is called which sets the record number, and subsequent random read and write operations continue from the selected point in the file.

FUNCTION 37: RESET DRIVE

Entry Parameters: Register C - 25H
Registers DE - Drive Vector

Returned Value: Register A - ØØH

The RESET DRIVE function will allow resetting of specified drive(s). The passed parameter is a 16-bit vector of drives to be reset, the least significant bit is Drive A. If another process has a file open on the drive and if the drive is removable or READ-ONLY, no drives are reset.

Register A is ØØH if no errors occur, but if a physical or logical error is returned, register A is FFH and register H returns the error code.

FUNCTION 40: WRITE RANDOM WITH ZERO FILL

Entry Parameters: Register C - 28H
Register DE - FCB address

Returned Value: Register A - Error Code
Register H - Physical or Logical Error

The WRITE RANDOM WITH ZERO FILL function is similar to the WRITE RANDOM function (function 34) except that a previously unallocated data block is filled with zeros before the record is written. If this function has already been used to create a file, records accessed by a READ RANDOM operation that contain all zeroes will identify unwritten random record numbers. Unwritten random records in allocated data blocks of files created using the WRITE RANDOM function contain uninitialized data (see WRITE RANDOM FUNCTION and Appendix A for a description of returned values).

FUNCTION 41: TEST AND WRITE RECORD

Entry Parameters: Register C - 29H
Register DE - FCB address

Returned Value: Register A - Error Code
Register H - Physical or Logical Error

The TEST AND WRITE RECORD function will provide a means of verifying the current contents of a record on disk before updating it. The calling program must set bytes rØ, r1, and r2 of the FCB addressed by register DE to the random record number of the record to be tested. The original version of the record (i.e., the record to be tested) must reside at the current DMA address, followed immediately by the new version of the record. The record size can range from 128 bytes to 16 times that value depending on the BDOS Multi-Sector Count (see function 44).

Function 41 verifies that the first record is identical to the record on disk before replacing it with the new version of the record.

If the record on disk does not match, the record on disk is not changed and an error code is returned to the calling program. This function is intended for use in situations where more than one process has Read/Write access to a common file. Function 41 is a logical replacement for the record Lock/Unlock sequence of operations because it prevents two processes from simultaneously updating the same record.

On return, function 41 will set register A to zero if the function was successful; otherwise, register A contains one of the following error codes:

01 - READ MODE: READING UNWRITTEN DATA
03 - CANNOT CLOSE CURRENT EXTENT
04 - SEEK TO UNWRITTEN EXTENT
06 - SEEK PAST PHYSICAL END OF DISK (RANDOM RECORD NUMBER OUT OF RANGE)
07 - RECORD DID NOT MATCH

NOTE: This function is the only function which returns an error code of '07' in register A indicating non-matching sectors.

08 - RECORD LOCKED BY ANOTHER PROCESS
10 - FCB CHECKSUM ERROR
11 - UNLOCKED FILE VERIFICATION ERROR
255 - PHYSICAL ERROR (REFER TO REGISTER H)

If a physical or logical error is returned, the four LSBs of register H contain one of the following error codes:

01 - PERMANENT ERROR
02 - READ-ONLY DISK
03 - READ-ONLY FILE OR FILE OPEN IN READ-ONLY MODE
04 - SELECT ERROR

TEST AND WRITE RECORD function also sets the four high order bits of register H to the number of records successfully tested or written.

FUNCTION 42: LOCK RECORD

Entry Parameters:	Register C - 2AH Register DE - FCB Address
Returned Value:	Register A - Error Code Register H - Physical Error

The LOCK RECORD function will lock one or more consecutive records so that no other program with access to the records can simultaneously lock or update them. This function is only supported for files open in unlocked mode. If it is called for a file open in Locked or Read-Only mode, no locking action is performed and a successful result is returned.

The calling process passes in register DE, the address of an FCB in which the Random Record field is filled with the random record number of the first record to be locked. The number of records to be locked is determined by the BDOS Multi-Sector Count (see Function 44).

The LOCK RECORD function requires that each record number to be locked reside in an allocated block for the file. In addition, Function 42 verifies that none of the records to be locked are currently locked by another process. Both of these tests are made before any records are locked. Each locked record consumes an entry in the BDOS system lock table which is shared by locked record and open file entries. If there is not sufficient space in the system lock table to lock all the specified records, or the process record lock limit is exceeded, the LOCK RECORD function locks no records and returns an error code to the calling process.

Upon return, the LOCK RECORD function sets register A to zero if the lock operation was successful; otherwise, register A contains one of the following error codes:

01 - READ MODE: READING UNWRITTEN DATA
03 - CANNOT CLOSE CURRENT EXTENT
04 - SEEK TO UNWRITTEN EXTENT
06 - SEEK PAST PHYSICAL END OF DISK (RANDOM RECORD NUMBER OUT OF RANGE)
08 - RECORD LOCKED BY ANOTHER PROCESS
10 - FCB CHECKSUM ERROR
11 - UNLOCKED FILE VERIFICATION ERROR
12 - PROCESS RECORD LOCK LIMIT EXCEEDED
13 - ACCESSED FILE NOT PREVIOUSLY OPENED
14 - NO ROOM IN THE SYSTEM LOCK LIST
255 - PHYSICAL ERROR (REFER TO REGISTER H)

If a physical or logical error is returned, register A is ØFFH and register H will consists of one of the following:

PHYSICAL ERROR CODES: (IN THE FOUR LOW-ORDER BITS OF H)

Ø1 - PERMANENT ERROR
Ø4 - SELECT ERROR

FUNCTION 43: UNLOCK RECORD

Entry Parameters: Register C - 2BH
Register DE - FCB Address

Returned Value: Register A - Error Code
Register H - Physical Error

The UNLOCK RECORD function will unlock one or more consecutive records previously locked by the LOCK RECORD function. This function is only supported for files open in unlocked mode. If it is called for a file open in Locked or Read-Only mode, no locking action is performed and a successful result is returned.

The calling program passes, in register DE, the address of an FCB in which the Random Record field is filled with the random record number of the first record to be unlocked. The number of records to be unlocked is determined by the BDOS Multi-Sector Count (see Function 44).

The UNLOCK RECORD function will not unlock a record that is currently locked by another process. However, no error is returned if a process attempts the process. Therefore, if the Multi-Sector Count is greater than one, the UNLOCK RECORD function unlocks all records locked by the calling program, while skipping those records locked by other programs.

On return, the UNLOCK RECORD function sets register A to zero if the unlock operation was successful; otherwise, register A will contain one of the following error codes:

Ø1 - READ MODE: READING UNWRITTEN DATA
Ø3 - CANNOT CLOSE CURRENT EXTENT
Ø4 - SEEK TO UNWRITTEN EXTENT
Ø6 - SEEK PAST PHYSICAL END OF DISK (RANDOM RECORD NUMBER OUT OF RANGE)

1Ø - FCB CHECKSUM ERROR
11 - UNLOCKED FILE VERIFICATION ERROR
13 - ACCESSED FILE NOT PREVIOUSLY OPENED
255 - PHYSICAL ERROR (REFER TO REGISTER H)

If a physical or logical error is returned, register A is ØFFH and register H consists of one of the following:

PHYSICAL ERROR CODES: (IN THE FOUR LOW-ORDER BITS OF H)

Ø1 - PERMANENT ERROR
Ø4 - SELECT ERROR

FUNCTION 44: SET MULTI-SECTOR COUNT

Entry Parameters: Register C - 2CH
Register E - Number of sectors

Returned Value: Register A - Return Code

The SET MULTI-SECTOR COUNT function will provide logical record blocking. It enables a process to read and write from one to 16 'physical' records of 128 bytes at a time during subsequent BDOS Read and Write functions. It also specifies the number of 128-byte records to be locked or unlocked by the BDOS LOCK and UNLOCK functions.

Function 44 sets the Multi-Sector Count value for the calling program to the value passed in register E. Once set, the specified Multi-Sector Count remains in effect until the calling process makes another 'set count' call and changes the value. Note that the CCP will set the count to one when it initiates a transient program.

The Multi-Sector Count affects BDOS error reporting for the BDOS READ, WRITE, LOCK and UNLOCK functions. If an error interrupts these functions when the Multi-Sector Count is greater than one, they return the number of records successfully processed in the for high-order bits of register H.

Upon return, register A is set to zero if the specified value is in the range of 1 to 16; otherwise, register A is set to ØFFH.

FUNCTION 45: SET BDOS ERROR MODE

Entry Parameters: Register C - 2DH
Register E - BDOS Error mode

The SET BDOS ERROR MODE function will determine how physical and logical errors are handled for a process. The function can exist in one of three modes: the Default mode, the Display and Return mode, and the Return Error mode.

In the Default mode, the BDOS will display a system message at the console identifying the error and will then terminate the calling program.

In the Return Error mode, the BDOS will set register A to ØFFH (255 decimal), place an error code identifying the physical or logical error in the four low-order bits of register H, and return to the calling program. No system messages are displayed, however, when the BDOS is in Return Error mode.

In the Display and Return mode, the BDOS will display the error on the console and return the error code in the H register in the same manner as mode two.

Function 45 will set the BDOS error mode for the calling program to the mode specified in register E. If this register is set to 'ØFEH', **mode three** (Display and Return) will be set; if the register is set to 'ØFFH', **mode two** (Return Error) will be set; if set to any other value, **mode one** (Default) will be set.

FUNCTION 46: RETURN FREE DISK SPACE

Entry Parameters: Register C - 2FH
Register E - ØØ - Drive A
Ø1 - Drive B
Ø2 - Drive C
Ø3 - Drive D

Returned Value: Current DMA Buffer - Number of free records on disk

The RETURN FREE DISK SPACE function will return the number of free records remaining on disk. To maintain upward compatability, use this function instead of counting space from allocation records.

The function returns a 24-bit value in the first three bytes of the current DMA buffer, with the low bits in the first byte, the middle bits in the second byte and the high bits in the third byte.

Normally, register A is zero upon return, but if a physical or logical error is returned, register A is ØFFH and register H is one of the following codes:

Ø1 - Permanent error
Ø4 - Select error

FUNCTION 47: CHAIN TO PROGRAM

Entry Parameters: Register C - 2FH

The CHAIN TO PROGRAM function will provide a means of chaining from one program to the next without operator intervention. Although there is no passed parameter for this call, the calling program must place a command line terminated by a null byte in the default DMA buffer (8ØH).

Function 47 does not return any values to the calling program because any errors encountered are handled by the CCP.

FUNCTION 48: FLUSH BUFFERS

Entry Parameters: Register C - 3ØH

Returned Value: Register A - Error flag
Register H - Permanent or Logical Error

The FLUSH BUFFERS function will force the write of any write-pending records contained in internal blocking/deblocking buffers. This function only affects those systems that have implemented a write-deferring blocking/deblocking algorithm in their BIOS.

Upon return, register A is set to zero if the flush operation was successful or if a physical or logical error is returned, register A is ØFFH and register H is Ø1 - Permanent Error.

FUNCTION 152: PARSE FILENAME

Entry Parameters: Register C - 98H
Register DE - PFCB Address

Returned Value: Register HL - Return Code
(Parsed File Control Block)

The PARSE FILENAME function will parse an ASCII filename and prepare an FCB; the calling program will pass the address of the Parse File Control Block (PFCB) in register DE. The PFCB contains the address of the ASCII filename string followed by the address of the target FCB. Initialization of the PFCB data structure is shown below in assembly language:

```
PFNCB:
        DW      FLNAME
        DW      FCB
FLNAME:
        DS      128
FCB:
        DS      36
```

The file specification should be written in the following form:

d:filename.ext

Function 152 will parse the first file specification found in the input string, first eliminating leading blanks and tabs. It will then assume the file specification ends on the first delimiter encountered that is out of context with the specific field it is parsing.

The specified FCB will be initialized as follows:

BYTE	DESCRIPTION
Ø	Drive field set to specified drive number. If drive not specified, default value is used. Ø = default 1 = Drive A 2 = Drive B . . . 16 = Drive P
1-8	Name field set to specified filename, and all letter are converted to uppercase. Filename < 8 characters, remaining bytes in field padded with blanks. Filename > 8 characters, error message displays. Filename has (*) , all remaining bytes filled with (?).
9-11	Extension field set to specified extension; if none specified, field is initialized to blanks (all letters converted to uppercase). Extension < 3 characters, remaining bytes padded with blanks. Extension has (*), all remaining bytes filled with (?).

In case of an error, all fields not parsed will be set to their default values and register HL will return a ØFFFFH indicating the error.

The PARSE FILENAME function, on a successful parse, will check the next item in the Filename string, skipping over trailing blanks and tabs and look at the next character. If the character is a null or a carriage return, it will return a Ø indicating the end of the Filename string; if the next character is a delimiter, it will return the address of the delimiter; if the next character is <u>not</u> a delimiter, it will return the address of the delimiting blank or tab.

If the first non-blank or non-tab character is a null (Ø) or a carriage return within the Filename string, function 152 will return a zero indicating the end of the string, and the FCB will be initialized to its default value. If function 152 is to be used to parse a subsequent filename in the Filename string, the returned address should be advanced over the delimiter before placing it in the PFCB.

FUNCTION 158: ATTACH LIST

Entry Parameters: Register C - 9EH

Returned Value: None

The ATTACH LIST function will attach the previously specified list device assignment to the calling program. If the list device is already attached to another job or program, the calling program will relinquish the CPU and wait until the other program detaches from the list device; the attach operation will take place when it becomes free. See FUNCTION 250.

FUNCTION 159: DETACH LIST

Entry Parameters: Register C - 9FH

Returned Value: None

The DETACH LIST function will detach the previously specified list device assignment from the calling program. If no list device is currently attached, no action will take place. See FUNCTION 250.

FUNCTION 160: SET LIST

Entry Parameters: Register C - AØH
Register E - List Device

Returned Value: None

The SET LIST function will detach the currently attached list device from the calling program and attach the newly specified list device. If the list device is already attached to another job or program, the calling program will relinquish the CPU and wait until the other program detaches from the list device; the attach operation will take place when it becomes free. The list device can also be set using function 8, SET IOBYTE.

The value of register E can be one of the following:

E = Ø - No List Device
E = 1 - Echo to Console
E = 2 - Logical List Device 1
E = 3 - Logical List Device 2

See FUNCTION 250.

FUNCTION 161: CONDITIONAL ATTACH LIST/

Entry Parameters: Register C - A1H

Returned Value: Register A - Return Code

The CONDITIONAL ATTACH LIST function will attach the previously specified list device assignment to the calling program if the list device is currently unattached. If the list device is currently attached to another job or program, a value of ØFFH in register A is returned, indicating that the list device could not be attached. See FUNCTION 250.

FUNCTION 164: GET LIST NUMBER

Entry Parameters: Register C - A3H

Returned Value: Register A - List Number
Register H - Selected List Device Number
Register L - Attached List Device Number

The GET LIST NUMBER function will return the value of the specified list device assignment of the calling program. See FUNCTION 250.

FUNCTION 217: GET/SET CONFIG BYTE

Entry Parameters: Register D - 8-bit Mask
Register E - ØØH - SET Mode
Register E - Ø1H - RESET Mode
Register E - Ø2H - TOGGLE Mode
Register E - Anything other - GET Mode

Returned Values: Register A - ØØH - SET, RESET, & TOGGLE Modes
Register A - Current CONFIG BYTE (GET Mode)

The GET/SET CONFIG BYTE function will enable or disable the [CTRL C], [CTRL P], and the [CTRL K] functions by setting a bit within the CONFIG BYTE. Any bit set in the D register mask will affect the user's CONFIG BYTE according to the mode set by the value in the E register. When the values change within the different modes, the results will vary; e.g.:

SET MODE

```
CONFIG BYTE = ØØ1Ø 1Ø1Ø
Register D  = ØØØØ ØØØ1
Result      = ØØ1Ø 1Ø11
```

RESET MODE

```
CONFIG BYTE = ØØ1Ø 1Ø1Ø
Register D  = ØØØØ ØØ1Ø
Result      = ØØ1Ø 1ØØØ
```

TOGGLE MODE

```
CONFIG BYTE = ØØ1Ø 1Ø1Ø
Register D  = ØØØØ ØØ11
Result      = ØØ1Ø 1ØØ1
```

GET MODE

The GET Mode will ignore the D register and return the CONFIG BYTE in the A register.

A bit **set** within the CONFIG BYTE will **enable** the associated function, while a bit **reset** within the CONFIG BYTE will **disable** the associated function; a 'Ø' equates to 'off' and a '1' equates to 'on'. The CONFIG BYTE bits have been defined as follows:

Bit 0 = [CTRL C] Warm Boot function
Ø = Disable
1 = Enable

Bit 1 = [CTRL P] Trap
Ø =Disable (disables printer echo enable/disable)
1 = Enable

Bit 2 = [CTRL K] Trap
Ø = Disable (disables printer form feed)
1 = Enable

Bits 3-7 = Reserved for Future Use

FUNCTION 218: RETURN CURRENT CURSOR POSITION

Entry Parameters: Register C - ØDAH

Returned Values: Register H - Current Cursor X
Register L - Current Cursor Y

The RETURN CURRENT CURSOR POSITION function will return the X and Y values corresponding to the position of the cursor on the screen. The position of the cursor may be set directly by using the cursor positioning function in the video driver.

FUNCTION 222: OUTPUT TONE

Entry Parameters: Register C - ØDEH
Register E - Tone Generator
Register B - Attenuation
Register D - Time Duration
Register HL - Frequency/Noise Control

Returned Values: None

The OUTPUT TONE function will output a desired frequency of sound, for a specified amount of time and at a specified volume, through one of the three tone generators within the Vector 4. This function is also capable of outputting one of two different types of 'noise' (white noise and periodic noise) allowing user-defined frequency ranges, volume, and duration.

The **tone generator** is selected from the contents of the E register, the values consist of the following:

Ø = Tone Generator 1
1 = Tone Generator 2
2 = Tone Generator 3
3 = White Noise Generator
4 = Periodic Noise Generator

The **attenuation,** or reduction of volume, of the tone generator is selected by using the value in the B register (the higher the value, the lower the volume), the values consist of the following:

Ø = Ødb	8 = 16db
1 = 2db	9 = 18db
2 = 4db	10 = 20db
3 = 6db	11 = 22db
4 = 8db	12 = 24db
5 = 10db	13 = 26db
6 = 12db	14 = 28db
7 = 14db	15 = OFF

Note: Since only the low-order four bits determine the attenuation, the high-order bits are ignored.

The **time duration** is selected by a single byte value from the D register in the range of Ø to 255; this range, measured in seconds, converts from 1/64th second to 255/64 seconds. The unit of time measurement, therefore, is based on an interval of 1/64th of a second; e.g., the value '58' equals 58/64 of a second. If a time of zero (Ø) is selected, the time for the tone generator will not be reset. CP/M decrements each timer every 1/64th of a second; when a given timer reaches zero, CP/M automatically turns the appropriate tone generator off.

The **frequency** is selected by a frequency range between 62.5 KHz to 61.Ø Hz from the HL register (for more information on frequency settings for pitch control, see Appendix B). If a frequency of zero (Ø) is selected, the frequency of the tone generator remains unaltered.

The **noise generator,** either white noise (random) or periodic noise (selective pulse), is selected from the value of the E register; however, only one noise generator can be selected at a time. The frequency range of the generated noise can be set by using one of the modes from the following table:

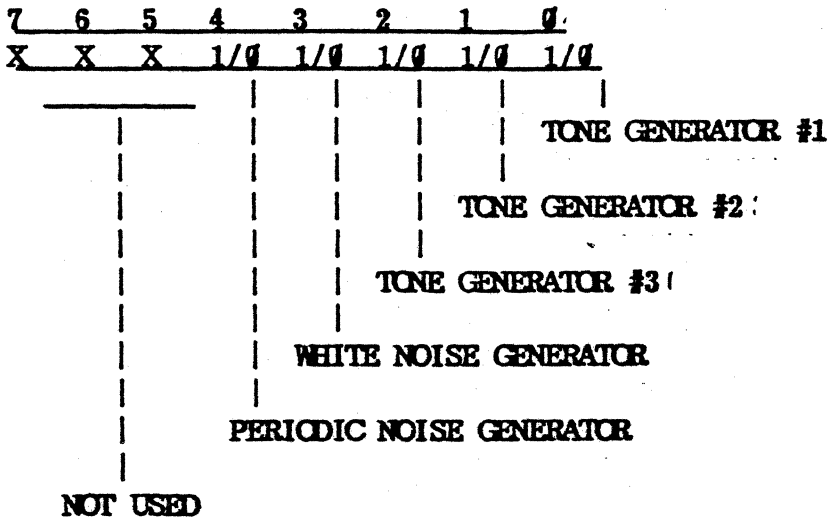
Ø = No Frequency Change
1 = Higher Frequency (3906 Hz)
2 = Medium Frequency (1953 Hz)
3 = Lower Pitch (976 Hz)
4 = Use Tone Generator 3 for Pitch Control

Use of the tone generator (4 in above table) will enable a finer control of frequency range, i.e., white noise or periodic noise will be more finely controlled around a given pitch rather than a random generation of noise. The mode of operation, selected from the above table, for the noise generator is placed into register L; however, any of the values given in Appendix B can be used to base the noise generator around a specific pitch. The volume and duration of the noise generator is controlled in the same manner as the tone generators.

FUNCTION 223: RETURN TONE GENERATOR STATUS

Entry Parameters:	Register C - ØDFH
Returned Value:	Register A - Tone Generator Status Vector

The RETURN TONE GENERATOR STATUS function will return to the caller the status of all of the three tone generators and the two noise generators. The bit positions of each tone and noise generator is as follows:



A bit that is ON (1) designates that the tone/noise generator is active.

A bit that is OFF (Ø) designates that the tone/noise generator is inactive.

NOTE: The white and periodic noise generators are actually one generator; when the white noise generator is active, the periodic noise generator is also active.

FUNCTION 224: DETECT R-O STATUS

Entry Parameters: Register C - ØEØH

Returned Value: Register A - ØØ - R/W
Ø1 - R-O

The DETECT R-O STATUS function will detect if a disk is (physically) write protected. It returns a Boolean value in the A register.

FUNCTION 225: RETURN BIOS ERROR CODE

Entry Parameters: Register C - ØE1H

Returned Value: Register HL - Error Code

The RETURN BIOS ERROR CODE function will return the BIOS error code from the last disk access in register HL. It is used to test for errors when error messages are inhibited by function 226. Register H returns a BIOS code related to a specific BIOS error message. Register L returns a BDOS error code related to a specific BDOS error message.

FUNCTION 226: INHIBIT/ENABLE BDOS ERRORS

Entry Parameters: Register C - ØE2H
Register E - ØFFH - Enable
ØØH - Disable

Returned Value:

The INHIBIT/ENABLE BDOS ERRORS function will allow the programmer to inhibit or enable BDOS error messages. When the errors are disabled, function 225 should be invoked after every disk access in order to test for errors. At the end of execution of the program the errors should be enabled.

FUNCTION 227: INHIBIT/ENABLE BIOS ERRORS

Entry Parameters: Register C - ØE3H
Register E - ØFFH - Enable
ØØH - Disable

The INHIBIT/ENABLE BIOS ERRORS function will enable and disable BIOS soft errors. If enabled, BIOS soft errors are either printed on the console (if error reporting enabled, see function 226) or can be read from register H after a function 225 call.

FUNCTION 228: GET/SET ACCOUNT CODE

Entry Parameters: Register C - ØE4H
Register E - ØFFH - Get Code
Account Code (to set)

Returned Value: Register A - Account Code

The GET/SET ACCOUNT CODE function will get or set the account code. The account code works in a manner similar to the user number. It is used to coordinate auto boot commands and to secure disk files. A file is locked under an account code if bit 7 of the first letter of the filename is high. This function was designed primarily for multiuser systems. The code may be in the range from ØØ-ØFH.

FUNCTION 229: START DESPOOLER

Entry Parameters: Register C - ØE5H
Register DE - FCB address

Returned Value: Register A - Return Code

The START DESPOOLER function will start the print despooler; a despooler outputs data from a disk file to a list device. The FCB addressed by register DE must point to an unambiguous filename. The current record of the FCB (FCB + 32) should contain the list device driver number, (1 or 2). If bit 7 of the current record (FCB + 32) is high, the despooler will use auto-paging. The function can return the following codes in register A:

ØØH - Despooler Activated
FFH - Despooler Busy
FEH - Bad Filename
FDH - File Not Found
FCH - Illegal Print Device
ØFBH - Printer Busy

FUNCTION 230: ABORT DESPOOLER

Entry Parameters: Register C - ØE6H

Returned Value: Register A - Return Code

The ABORT DESPOOLER function will abort operation of the despooler. In the event that an FEH is returned, it means that a user has attempted to abort the despool process of another user. The function will return the following code in register A:

ØØH - Despool Aborted
FFH - Despool Not ctive
FEH - Account Number Mismatch

FUNCTION 231: START SPOOLER

Entry Parameters: Register C - ØE7H
Register DE - FCB

Returned Value: Register A - Return Code

The START SPOOLER function will intercept characters going to the printer and redirect them to a disk file. The current list device will be released if the spooler is successfully activated. The function will return the following code in register A:

ØØH - Spooler Activated
FFH - Spooler Busy
FEH - Destination File Exists
FDH - No Directory Space
FCH - Illegal Filename

FUNCTION 232: CLOSE SPOOLER

Entry Parameters: Register C - ØE8H

Returned Value: Register A - Return Code

The CLOSE SPOOLER function will terminate the use of the print spooler. The spooler must be closed to assure that no file data is lost. In the event of a disk error (such as disk full), the spooler will terminate on its own. The function will return one of the following codes in register A:

ØØH - Spooler Closed
FFH - Spooler was not Active
FEH - Disk Write During Closing
FDH - Unable to Close File

FUNCTION 233: RELEASE TIME SLICE

Entry Parameters: Register C - ØE9H

The RELEASE TIME SLICE function will release the current time slice. It should be incorporated in all long delay loops as it will improve the system efficiency. This is especially important if the program will run in a multiuser system.

FUNCTION 234: SET DMA TASK

Entry Parameters: Register C - ØEAH
Register E - DMA task number

The SET DMA TASK function will set the DMA task for disk I/O. It is used by the operating system and should normally not be used in application programs. The DMA task number is passed in register E.

FUNCTION 235: CHAIN CCP COMMAND

Entry Parameters: Register C - ØEBH
Register DE - Address of Command String

The CHAIN CCP COMMAND function will chain CCP commands and works similarly to a single line submit file. Use function 235 at the end of a program when you want to invoke another program. Pass the address of a valid CCP command string in the DE register pair.

The string must be terminated by a null (ØØH). The system will warm boot and the command will be executed by the CCP.

FUNCTION 236: RETURN OUTPUT STATUS

Entry Parameters: Register C - ØECH

Returned Value: Register A - ØØ - Not Ready
- ØFFH - Ready

The RETURN OUTPUT STATUS function will return the output status, (ready or not ready to print), of the currently selected print device.

FUNCTION 237: RETURN INPUT STATUS

Entry Parameters: Register C - ØEDH

Returned Value: Register A - ØØ - No Character Present
- ØFFH - Character Present

The RETURN INPUT STATUS function will return the input status (character present or not) of the currently selected list device.

FUNCTION 238: LIST INPUT

Entry Parameters: Register C - ØEEH

Returned Value: Register A - Input Character

The LIST INPUT function will input a character from the currently selected list device. It is used primarily for printers that require a special communications protocol, or can be used with printers that have a keyboard.

FUNCTION 239: RETURN PRINTER TYPE

Entry Parameters: Register C - ØEFH
Register E - ØØH - Current List Device
ØlH - List Device 1
Ø2H - List Device 2

Returned Values: Registers HL Type Bytes (see text)

The RETURN PRINTER TYPE function will return the type bytes for the currently selected printer. Register L contains Type I byte (printer type) and register H contains the Type II byte (the interface byte). For a list of possible values see the documentation for the List Device Drivers (paragraph 3.3.9 - GENLIST Command). Pass a ØØH, ØlH or Ø2H in Register E, depending on which list device type is wanted.

FUNCTION 240: INITIALIZE PRINTER

Entry Parameters: Register C - ØFØH

The INITIALIZE PRINTER function will perform any necessary initialization required by the currently selected print device.

FUNCTION 241: ENABLE/DISABLE CIRCULAR BUFFERS

Entry Parameters: Register C - ØF1H
Register E - ØFFH - Enable
ØØH - Disable

The ENABLE/DISABLE CIRCULAR BUFFERS function will enable and disable the console input circular buffers, for use with a multiuser system. The buffers will be automatically re-enabled on a cold or warm boot.

FUNCTION 242: ENABLE/DISABLE KEYBOARD CONVERSIONS

Entry Parameters: Register C - ØF2H
Register E - ØFFH - Enable
ØØH - Disable

The ENABLE/DISABLE KEYBOARD CONVERSIONS function will enable and disable the keyboard conversions normally carried out by the Monitor. The conversions are automatically re-enabled on a cold or warm boot.

FUNCTION 243: ENABLE/DISABLE AUTO PAGING

Entry Parameters: Register C - ØF3H
Register E - ØFFH - Enable
ØØH - Disable

The ENABLE/DISABLE AUTO PAGING function will enable or disable the printer auto-paging. The function is automatically toggled along with the printer when the [CTRL P] function is used in function 1Ø, READ CONSOLE BUFFER. It is also automatically cleared on warm or cold boot or by the INITIALIZE PRINTER, function 240.

FUNCTION 244: RETURN REVISION LEVEL

Entry Parameters: Register C - ØF4H

Returned Value: Register H - Version Levels
H=0 (Other Versions of CP/M)
H=1 (Other Versions of CP/M)
H=2 (Vector 4 CP/M)
Register L -Revision Levels

The RETURN REVISION LEVEL function will return the revision level of the current version of CP/M that is being used. The function will be used to determine if certain BDOS features are installed.

FUNCTION 245: ENABLE/DISABLE CROSS-BANK CALLS

Entry Parameters: Register C - ØF5H
Register E - ØØ - Disable
- ØFFH - Enable

The ENABLE/DISABLE CROSS-BANK CALLS function will either enable or disable the CCP from calling the BDOS, and the BDOS from calling the BIOS, through the BDOS or BIOS Jump Tables within the user's memory. This feature allows the user to modify the BIOS Jump Table, or intercept BDOS calls, and reroute them to the user's own routines. The BDOS and BIOS Jump Tables may then be restored to their original form by initiating a cold boot sequence, or by calling this function with a zero in the E register. The default used by the BDOS and CCP is to not use the user's vectors and jump directly tot he BDOS and BIOS respectively. This default is used because the system throughput is increased enormously when the CCP can call the BDOS, and the BDOS can call the BIOS directly; therefore, the program should disable the cross-bank calls upon termination of its task.

Without setting the BDOS Base Address down (#246) the BIOS Jump Table will be refreshed on Warm Boots.

FUNCTION 246: CHANGE BDOS BASE ADDRESS

Entry Parameters: Register C - ØF6H
Register E - Page Address

The CHANGE BDOS BASE ADDRESS function provides a means to the programmer to lower the base address of the BDOS. This feature is useful for programs that want to intercept system calls, to perform additional tasks, such as plotting, asychronous communications, etc., and yet not be disturbed by a warm boot, or the end of execution of a program. Function 246 will adjust both the BDOS vector at location 5 and the pseudo-BDOS located at the top of the TPA to the page address specified in the E register (i.e., E register equals ØCEH, the vector at location 5 will change to JMP CEØ6H; and the pseudo-BDOS will be relocated to CEØØH). Subsequent attempts from the BDOS to access the user's disk I/O error vectors will get these vectors from the CEØØH pseudo-BDOS. If the E register contains a zero, or if a cold boot sequence is initiated, the BDOS will return to its default location.

FUNCTION 247: RETURN MICROPROCESSOR CLOCK SPEED

Entry Parameters: Register C - ØF7H

Returned Value: Register A - Clock Speed in MHz
(4 = 4MHz, 5 = 5MHz, 6 = 6MHz)

The RETURN MICROPROCESSOR CLOCK SPEED function will return the clock speed of the microprocessor to the calling program. This can be useful for programs that require semiaccurate time delays, and these delays need to be adjusted according to the clock speed of the processor.

FUNCTION 248: READ ACTUAL RECORD

Entry Parameters: Register C - ØF8H
Register DE - Record Number

Returned Values: Register A - Error Code
Register H - Physical or Logical Error

The READ ACTUAL RECORD function will read from the currently selected drive one to sixteen 128-byte records starting at the record number passed in the E register into memory beginning at the DMA address. This function will allow the programmer to read all but the system tracks on a logical disk unit. Since the largest logical disk unit that can be supported by VECTOR 4 CP/M can only be 8 Mbytes in size (including the directory), this allows for 65,536 128-byte records. The programmer need only select the appropriate drive and then read the actual record desired. This facility allows VECTOR 4 CP/M to retrieve information from non-CP/M structured diskettes. The value ØØH is returned in the A register if the read operation was successful; otherwise, an error code will be returned as described in functions 20 and 33, READ SEQUENTIAL and RANDOM.

FUNCTION 249: WRITE ACTUAL RECORD

Entry Parameters: Register C - ØF9H
Register DE - Record number

Returned Values: Register A - Error Code
Register H - Physical or Logical Error

The WRITE ACTUAL RECORD function will write from one to sixteen 128-byte records, beginning at the current DMA address onto the currently selected disk starting at the record number passed. This funciton is the compliment to function 248, READ ACTUAL RECORD, and enables the programmer to create and maintain a non-CP/M compatible diskette and/or disk. The programmer must be specially careful in the use of this funciton for a number of reasons. First, the actual disk directory must be contained within the 8 Mbyte limit; therefore, the first several records starting with record zero are the actual disk directory on a CP/M diskette. Once the directory information has been lost, there is no way to recover the files that were contained within the directory record. Second, the programmer must carefully insure that he is not writing to arecord that is being used by a file on that disk. Third, there is no need for an FCB since this function is more of a physical interface to a disk rather than a logical file interface. Since there is no FCB maintained, there is no way for VECTOR 4 CP/M to maintain any allocation information about records that are wirtten using this function.

This function will return ØFFH in the A register if the write operation was successful; otherwise, an error code is returned as described in functions 21 and 34, WRITE SEQUENTIAL and RANDOM.

FUNCTION 250: SELECT LIST DEVICE

Entry Parameters: <e> = List device (same as function 160)

Returned Values: None

Selects desired list device for subsequent operations by functions (158, 159, 161, 164).

FUNCTION 255: RELEASE FILE RESOURCES

Entry Parameters: Register C - ØFFH

The RELEASE FILE RESOURCES function will release all open files and locked records used by the calling task. This function is executed automatically upon entry to the CCP.

Exhibit 4-5. BDOS FUNCTION CHART

NO	FUNCTION NAME	INPUT	OUTPUT
Ø	SYSTEM RESET	None	None
1	CONSOLE INPUT	None	A = Char
2	CONSOLE OUTPUT	E= Char	None
3	READER INPUT	None	A=Char
4	PUNCH OUTPUT	E= Char	None
5	LIST OUTPUT	E=Char	See Def
6	DIRECT CONSOLE I/O	See Def	See Def
7	GET I/O BYTE	None	A = IOBYTE
8	SET I/O BYTE	E = IOBYTE	See Def
9	PRINT STRING	DE = Buffer	None
1Ø	READ CONSOLE BUFFER	DE = Buffer	See Def
11	GET CONSOLE STATUS	None	A = ØØ/FF
12	RETURN VERSION NUMBER	None	HL = Version #
13	RESET DISK SYSTEM	None	See Def
14	SELECT DISK	E=Disk Number	See Def
15	OPEN FILE	DE = FCB	A = Dir Code
16	CLOSE FILE	DE = FCB	A = Dir Code
17	SEARCH FOR FIRST	DE = FCB	A = Dir Code
18	SEARCH FOR NEXT	None	A = Dir Code
19	DELETE FILE	DE = FCB	A = Dir Code
2Ø	READ SEQUENTIAL	DE = FCB	A = Err Code
21	WRITE SEQUENTIAL	DE = FCB	A = Err Code

22	MAKE FILE	DE = FCB	A = Dir Code
23	RENAME FILE	DE = FCB	A = Dir Code
24	RETURN LOGIN VECTOR	None	HL = Login Vec
25	RETURN CURRENT DISK	None	A = Cur Disk*
26	SET DMA ADDRESS	DE = DMA	None
27	GET ADDR (Alloc)	None	HL = Alloc
28	WRITE PROTECT DISK	None	See Def
29	GET R/O VECTOR	None	HL = R/O Vect*
3Ø	SET FILE ATTRIBUTES	DE = FCB	See Def
31	GET ADDR (Disk Parms)	None	HL = DPB
32	SET/GET USER CODE	See Def	See Def
33	READ RANDOM	DE = FCB	A = Err Code
34	WRITE RANDOM	DE = FCB	A = Err Code
35	COMPUTE FILE SIZE	DE = FCB	rØ, r1, r2
36	SET RANDOM RECORD	DE = FCB	rØ, r1, r2
37	RESET DRIVE	DE = Drive	A = ØØH
4Ø	WRITE RANDOM WITH ZERO FILL	DE = FCB	A = Ret Code
41	TEST AND WRITE RECORD	See Def	See Def
44	SET MULTI-SECTOR COUNT	See Def	See Def
45	SET BDOS ERROR MODE	See Def	None
46	RETURN FREE DISK SPACE	See Def	See Def
47	CHAIN TO PROGRAM	C = 2FH	None
48	FLUSH BUFFERS	See Def	See Def
152	PARSE FILENAME	See Def	See Def
158	ATTACH LIST	C = 9EH	None
159	DETACH LIST	C = 9FH	None
160	SET LIST	See Def	See Def
161	CONDITIONAL ATTACH LIST	C = A1H	A = Ret Code
164	GET LIST NUMBER	C = A3H	A = List #
217	GET/SET CONFIG BYTE	See Def	See Def
218	RETURN CURRENT CURSOR POSITION	C = ØDAH	HL = Cursor Pos
219	RESET DEFAULT CONSOLE CONVERSION	C = ØDBH	A - Conv Table
220	CONVERT CODE	See Def	A = Converted V
222	OUTPUT TONE	See Def	None

3	RETURN TONE GENERATOR STATUS	C = ØDFH	A = Status vector
4	DETECT R/O STATUS	None	A = Boolean Value
5	RETURN BIOS ERROR CODE	None	HL = Err Code
6	INHIBIT/ENABLE BDOS ERRORS	See Def	None
7	INHIBIT/ENABLE BIOS ERRORS	See Def	None
8	GET/SET ACCOUNT CODE	See Def	See Def
9	START DESPOOLER	DE = FCB	A = Ret Code
1Ø	ABORT DESPOOLER	None	A = Ret Code
11	START SPOOLER	None	A = Ret Code
12	CLOSE SPOOLER	None	A = Ret Code
13	RELEASE TIME SLICE	None	None
14	SET DMA TASK	See Def	None
15	CHAIN CCP COMMAND	See Def	None
16	RETURN OUTPUT STATUS	None	A = Boolean Value
17	RETURN INPUT STATUS	None	A = Boolean Value
18	LIST INPUT	See Def	See Def
19	RETURN PRINTER TYPE	See Def	See Def
2Ø	INITIALIZE PRINTER	None	None
21	ENABLE/DISABLE CIRC BUFFERS	See Def	See Def
22	ENABLE/DISABLE KEY CONVERS	See Def	See Def
23	ENABLE/DISABLE AUTO-PAGING	See Def	See Def
24	RETURN REVISION LEVEL	C = ØF4H	HL = Rev Level
25	ENABLE/DISABLE CROSS-BANK CALLS	See Def	None
26	CHANGE BDOS BASE ADDRESS	See Def	None
27	RETURN MICROPROCESSOR CLOCK SPEED	C = ØF7H	A = Clock Speed
28	READ ACTUAL RECORD	See Def	See Def
29	WRITE ACTUAL RECORD	See Def	See Def
3Ø	SELECT LIST DEVICE	See Def	See Def
35	RELEASE FILE RESOURCES	None	None

Note that A = L, and B = H upon return.

SECTION V - BIOS JUMP ROUTINES

5.1 INTRODUCTION

The entry points into the BIOS from the cold start loader and the BDOS are detailed below in Exhibit 5-1. Entry to the BIOS is accomplished through a 'jump vector' located at 'BIOS', as shown. The jump vector is a sequence of 43 jump instructions which send program control to the individual BIOS subroutines.

All simple Character I/O operations are assumed to be performed in ASCII, upper and lower case, with high order (parity bit) set to zero. An end-of-media condition for an input device is given by an ASCII [CTRL Z] (1AH). Peripheral devices are seen by CP/M as 'logical' devices, and are assigned to physical devices within the BIOS. All Character I/O operations will check the IOBYTE for the current settings of the specific device.

The jump vector at 'BIOS' takes the form shown below where the individual jump addresses are given to the left in numeric order:

5-1. BIOS JUMP CHART

```
BIOS        CBOOT       ;ARRIVE HERE FROM COLD START LOAD
BIOS +3     WBOOT       ;ARRIVE HERE FOR WARM START
BIOS +6     CONST       ;CHECK FOR CONSOLE CHAR READY
BIOS +9     CONIN       ;READ CONSOLE CHARACTER IN
BIOS +12    CONOUT      ;WRITE CONSOLE CHARACTER OUT
BIOS +15    LIST        ;WRITE LISTING CHARACTER OUT
BIOS +18    PUNCH       ;WRITE CHARACTER TO PUNCH DEVICE
BIOS +21    READER      ;READ READER DEVICE
BIOS +24    HOME        ;MOVE TO TRACK ØØ ON SELECTED DISK
BIOS +27    SELDSK      ;SELECT DISK DRIVE
BIOS +30    SETTRK      ;SET TRACK NUMBER
BIOS +33    SETSEC      ;SET SECTOR NUMBER
BIOS +36    SETDMA      ;SET DMA ADDRESS
BIOS +39    READ        ;READ SELECTED SECTOR
BIOS +42    WRITE       ;WRITE SELECTED SECTOR
BIOS +45    LISTST      ;RETURN LIST STATUS
BIOS +48    SECTRAN     ;SECTOR TRANSLATE SUBROUTINE
BIOS +51    DISKCMND    ;DISK COMMAND ENTRY
BIOS +54    DMABANKSET  ;SET DMA BANK #
```

```
BIOS +57     SETIOBYTE     ;SET IOBYTE
BIOS +60     GETIOBYTE     ;RETURN IOBYTE
BIOS +63     ATTACH        ;ATTACH/DEATTACH CONSOLE TO TASK
BIOS +66     SLICERELEASE  ;RELEASE TIME SLICE
BIOS +69     TASKRELEASE   ;KILL CURRENT TASK
BIOS +72     CRNTASK       ;RETURN CURRENT TASK #
BIOS +75     TASKTABLE     ;RETURN POINTER TO TASK TABLE
BIOS +78     ALLOCLOCK     ;LOCK-OUT TASK SWITCHING
BIOS +81     ALLOCUNLOCK   ;UNLOCK TASK SWITCHING
BIOS +84     SOFTERROR     ;SET/RESET SOFT ERROR REPORT FLAG
BIOS +87     SUBNKMOV      ;MOVE MEMORY FROM ONE BANK TO ANOTHER
BIOS +90     SUBNKCAL      ;SETUP CROSS-BANK CALL
BIOS +93     USERDCOM      ;USER'S VERSION OF DISK COMMAND ENTRY
BIOS +96     CONFIG.SR     ;SET/RETURN CONFIGURATION BYTES
BIOS +99     LINSTAT       ;PRINTER INPUT STATUS
BIOS +102    LINCHAR       ;PRINTER INPUT CHARACTER
BIOS +105    LTYPE         ;RETURN PRINTER TYPE BYTES
BIOS +108    LINIT         ;INITIALIZE PRINTER
BIOS +111    HDVECTR       ;REMOVABLE MEDIA VECTOR RETURN
BIOS +114    E.DKYBD       ;ENABLE/DISABLE KEYBOARD CIRCULAR
                           BUFFER
BIOS +117    E.DCONV       ;ENABLE/DISABLE MONITOR CONVERSION
BIOS +120    LPAGE         ;SET PRINTER PAGING BYTES
BIOS +123    MOVBDOS       ;MOVE BDOS BASE ADDRESS
BIOS +126    RBUFSYS       ;RESET DISK BUFFERS
BIOS +129    TASKWAIT      ;SUSPENDS TASK-UNCONDITIONAL BASIS
```

Each jump address corresponds to a particular subroutine which performs the specific function.

5.2 CHARACTER I/O

CONSOLE

This is the principal interactive device which communicates with the operator,

LIST

This is the principal listing device, one of two with VECTOR 4 CP/M, accessed through LIST, LISTST, LINSTAT, LINCHAR, LTYPE, LINIT, and LPAGE, such as a printer.

PUNCH

This is the principal tape punching device, if it exists, and is normally a high-speed paper punch or Teletype.

READER

This is the principal tape reading device, such as a simple optical reader or Teletype.

If no peripheral device is assigned as the LIST, PUNCH, or READER device, the input devices will return no input and the output devices will ignore output characters.

5.3 DISK I/O

Disk I/O is always performed through a sequence of calls on the various disk access subroutines which set up the unit number to access, the track and sector on a paticular disk, the Direct Memory Address (DMA) address, and the DMA task involved in the I/O operation. After all these parameters have been set, a call is made to the READ or WRITE function to perform the actual I/O operation. Note that there is often a single call to SELDSK to select a disk drive, followed by a number of read or write operations to the selected disk before selecting another drive for subsequent operations. Similarly, there may be a single call to set the DMA address and DMA task, followed by several calls which read or write to/from the selected DMA address and task before the DMA address and task is changed. The sector subroutine should always be called before the READ or WRITE operations are performed.

Note that the READ and WRITE routines do perform several retries (3 reselections of the disk times 4 restores of the disk times 5 step-offs/ons consisting of a total of 72 retries) before reporting the error condition to the BDOS. If the error condition is returned to the BDOS, it will report the error to the user.

5.4 BIOS JUMP CALLS

BIOSINIT - BIOS Initialization

This subroutine will receive control from the Initial Program Loader (IPL) and is responsible for basic system initialization to the default values. The common RAM routines are moved into common RAM for use. The Z-80's interrupt system is set to 'mode 2' and initialized. The microprocessor clock speed is determined, set, and recorded for system use. The BIOS disk system is configured for the number of physical and logical drives, and all BIOS disk buffers are then cleared and initialized. The BIOS begins the initialization of the CCP, BDOS, and despooler. Control is transferred to the BIOS cold boot entry.

CBOOT - Cold Boot

This subroutine is responsible for placing copies of the BIOS Jump Table (ØF8ØØH), BDOS Cross Bank Jump (ØDDØØH), the Page Zero Jump Vectors to the BIOS and BDOS, and DPHs into the user's bank. All stacks for this user are reinitialized and control is then transferred to the CCP's Cold Boot entry address (the C register contains the unit number of the current drive to use; i.e., Ø = Drive A, 1 = Drive B, ..., 15 = Drive P).

WBOOT - Warm Boot

This subroutine will receive control when a warm start occurs. A warm start is performed whenever a user program branches to location ØØØØH. System parameters must be initialized as shown in the following Low RAM jumps:

location Ø,1,2 **Set to JMP WBOOT for warm starts (ØØØØH: JMP BIOS +3)**

location 3 **Set initial value of IOBYTE to default values, which is implemented in the BIOS**

location 4 **Set high-order nybl to current user number; set low-order nybl to current drive number (each nybl consists of values Ø-15 decimal)**

location 5,6,7 Set to JMP BDOS, which is the primary entry point to CP/M for transient programs (ØØØ5H: JMP 3ED06H)

The buffers for all removable media are flushed and cleared, and, after entry into the CCP, register C is set to the drive to select after system initialization. If the BDOS has not been moved by the BIOS +123 call, the BIOS will then move the user's copy of the BDOS Jump Table and the Low RAM jumps into the user's bank of memory. Control is then transferred to the CCP +3 with the C register set to the drive to select.

CONST - Console Status

The Console Status routine will return the status of the input port.

Passed: N/A

Returned: A = ØFFH if character is present
= Ø if no character present

Side effect: If the console has been defined by the IOBYTE to be background null, then the status returned is always a null (no character present); if the status is background suspend, the task is suspended until the console has been reattached.

CONIN - Console Input

The Console Input routine will wait for a character to become available and then return with the character in the Accumulator.

Passed: N/A

Returned: A = Character received

Side effect: If the console has been defined by the IOBYTE to be background null, then the character returned is always a null; if the status is background suspend, the task is suspended until the console has been reattached.

CONOUT - Console Output

The Console Output routine will output a character to the screen using the video drivers (Paragraph 5.6).

Passed: C = Character to display

Returned: None

Side effects: This routine makes use of the IOBYTE for the specific task that is doing the call. Therefore, the character may also be printed if the IOBYTE specifies echo to printer; also the task may be suspended until the console is reattached. If the console has been defined by the IOBYTE to be background null, then the character is ignored.

LIST - Print Character

The Print character routine sends the character passed to the list device specified by the IOBYTE for the current task.

Passed: C = Character to print

Returned: None

Side effects: This routine uses the IOBYTE for the current task, so the character may be sent to the console, printer 1, or printer 2, or it may simply be thrown away. If the character is sent to the console and the task does not have a console then the task will be suspended until a console is attached to the task.

PUNCH - Punch character

The Punch Character routine sends the character passed to the punch device (Port 4).

Passed: C = Character to punch

Returned: None

Side effects: Same as LIST

READER - Get a character from the Reader

The Reader routine waits for a character to become available from the reader (Port 4) and returns it to the calling program.

Passed: None

Returned: A = Character received

Side effects: If the input is from the reader, bit 7 is stripped off.

HOME - Home unit to track zero

The Home routine simply moves the currently selected unit to track zero.

Passed: None

Returned: A - Error code returned from physical driver.

Side effects: None

SELDSK - Select current Disk

The Select Disk routine selects the specified unit for future disk operations.

Passed: C = Unit number (Ø - 15, corresponding to Drives A - P)

Returned: DE = Pointer to the system's copy of the disk parameter header for the selected unit.
HL = Pointer to the user's copy of the disk parameter header for the selected unit.

Side effects: If there is an attempt to select a nonexistent unit, the HL and DE registers are returned with zero as an error indicator.

SETTRK - Set current Track

The Set Track routine sets the current track value for the currently selected unit for future disk operations.

Passed: BC= Physical track number

Returned: None

Side effects: None

SETSEC - Set current Sector

The Set Sector routine sets the current sector value for the currently selected unit for future disk operations.

Passed: C = Logical sector number

Returned: None

Side effects: None

SETDMA - Set current DMA address

The Set DMA address routine sets the starting address of the 128-byte record for future disk reads and writes.

Passed: BC = DMA address

Returned: None

Side effects: None

READ - Read logical record from disk

The Read routine reads a logical record in from the currently selected unit, track, and sector. The record is then placed at the starting address specified by the DMA address and task.

Passed: C = Ø - Normal read
= 1 - Directory read (buffered)

B = Ø - Sequential read
= ØFF - Random read

Returned: A - Error code returned from physical driver.

Side effects: None

WRITE - Write logical record to disk

The Write routine retrieves a 128-byte logical record, pointed to by the DMA address and from the DMA task, and writes it out to the currently selected unit, track, and sector.

Passed: C = Ø - Allocated write
= 1 - Directory write
= 2 - Unallocated write

Returned: A - Error code returned from physical driver.

Side Effects: The actual write may not be performed immediately, therefore, the error code may be postponed until a future read, write, warm boot, or reset buffer system call is made.

LISTST - List Output Status

The List Output Status routine returns the output status of the list device.

Passed: None

Returned: A - FF (ready for next character)
- Ø (printer not ready)

Side Effects: See IOBYTE

SECTRAN - Sector Translation

The Sector Translation routine converts logical sector numbers into physical sector numbers to be passed to the SET SECTOR routine.

Passed: BC - Logical sector number

Returned: HL - Physical sector number

Side Effects: None

DISKCMND - Disk Command

The Disk Command routine will save the state of the interrupt system prior to entering the physical disk drivers and re-establishes the interrupt system to its original state upon return. Note: this is the main entry into the physical disk drivers.

Passed: IX - Pointer to Device Control Block (DCB)

Returned: A - Physical disk driver error code

Side Effects: None

DMATASKSET - Set DMA Task

The DMATASKSET routine will set the current DMA task number used by the disk system for future disk operations.

Passed: C - Task number

Returned: None

Side Effects: This routine is automatically called by the CCP before control is passed to a transient program; therefore, there is no need for the user to set the DMA Task to the current task.

SETIOBYTE - Set IOBYTE

The Set IOBYTE routine will set the user's current IOBYTE to the specified passed value.

Passed: C - IOBYTE value to be set

Returned: A - Ø (IOBYTE successfully set)
- FF (IOBYTE not successfully set)

Side Effects: Since the IOBYTE is a reflection of shared system resources, if a specific resource has already been reserved by another user, this routine will not set the IOBYTE and will return an error of 'FF'. If the IOBYTE is successfully set, the user's IOBYTE at location 3 is updated to the value passed.

GETIOBYTE - Get IOBYTE

The Get IOBYTE routine will return the task's current IOBYTE setting.

Passed: None

Returned: A - IOBYTE

Side Effects: None

ATTACH - Attach/Detach console from task

The ATTACH routine will either attach or detach a specific console to or from the current task.

SLICERELEASE - Release remaining Time Slice

The SLICERELEASE routine will release the remaining time allocated to the current task to the next task to be executed.

Passed: None

Returned: None

Side Effects: The next task will receive the rest of the remaining time given up by the current task in addition to its normal time allocation.

TASKRELEASE - Put current task to sleep

The TASKRELEASE routine will put the specified current task to sleep which can be awakened later by an outside source, if necessary.

Passed: C = Ø - Current Task
=1-8 - Task 1-8

Returned: None

Side Effects: The system response time will increase because the sleeping task is no longer receiving a time slice.

CRNTTASK - Return Current Task Number

The CRNTASK routine will return the number and the bank of the currently executing user task.

Passed: None

Returned: A - Task number
E - Bank number

Side Effects: None

TASKTABLE - Returns Task Table Pointer

The TASKTABLE routine will calculate the address of the BIOS Task Control Block (BTCB) for a specific task.

Passed: C - Task number

Returned: HL - Pointer to the BTCB in the system bank

Side Effects: None

ALLOCLOCK - Lockout Task Reallocation

The ALLOCLOCK routine will perform a system-wide lockout of the possible bank-switching movement of a current user while the system is fulfilling that user's processing request.

Passed: None

Returned: None

Side Effects: None

ALLOCUNLOCK - Unlock Task Reallocation

The ALLOCUNLOCK routine will unlock the system-wide bank-switching movement of the current user's task.

Passed: None

Returned: None

Side Effects: Since the lockout is an incremental type lock (i.e., more than one user may perform a lockout at any one time), the actual unlocking of the system will not occur until the lock is returned to zero (until all users have unlocked the system).

SOFTERROR - Set/Reset Soft Error Report Flag

The SOFTERROR routine will enable or disable the reporting of sectors that have been successfully corrected by the Error Correction Code (ECC) routines. Whenever a sector has been successfully corrected, the message 'SOFT ERROR- ' will display at the user's console.

Passed: C - Ø (disable reporting)
- FF (enable reporting)

Returned: None

Side Effects: None

SUBNKMOV - Set Up Bank Move

The SUBNKMOV routine will move Ø to 128 bytes of data from one specific task to another.

Passed:
A - Number of bytes to move (1-128)
B - Destination task number
C - Source task number
DE - Destination starting address
HL - Source starting address

Returned: None

Side Effects: None

SUBNKCAL - Set Up Bank Call

The SUBNKCAL routine will call a specified (sub)routine in another task, and return control to the calling program upon completion.

Passed: A - Destination task number
HL - Subroutine address

Returned: None

Side Effects: All registers are preserved on both the call and the return from the subroutine.

USERDCOM - User's Disk Command Entry

The USERDCOM routine will perform the same function as the DISKCMND function except that it will transfer the user's DCB into the system bank and either get or return the user's buffer.

Passed: IX - Pointer to DCB

Returned: A - Physical Driver Error Code

Side Effects: None

CONFIG.SR - Configuration Set/Return

The CONFIG.SR routine will either set or return the current system configuration variables (see BIOS Configuration Block for a complete description of the system configuration variables).

Passed: C - Ø = Return Variables
- FF = Set Variables
DE - Pointer to Configuration Block buffer

Returned: None

Side Effects: Any changes made to the configuration block will be in effect upon completion of the set call.

LINSTAT - List Device Input Status

The LINSTAT routine will return the appropriate list device input status.

Passed: None

Returned: A - Ø = Character Not Ready
- FF = Character Waiting

Side Effects: See IOBYTE

LINCHAR - List Device Input Character

The LINCHAR routine will return a character received from the list device.

Passed: None

Returned: A - Character Received

Side Effects: The input status is not checked prior to acquisition of the character; it is therefore up to the user to ensure that a character is available before calling this routine (see IOBYTE).

LTYPE - Return List Device Type

The LTYPE routine will return the specific list device type byte.

Passed: C - Ø = Return currently selected list device type byte
- 1 = Return type bytes of list device #1
- 2 = Return type bytes of list device #2

Returned: HL - Type bytes

Side Effects: See LIST DEVICES

LINIT - Initialize List Device

The LINIT routine will call the Initialization routine of the currently selected list device.

Passed: None

Returned: Currently None

Side Effects: The current list devices return no values; however, since the user may write his own list device drivers, it is possible that values may be returned through the registers.

HDVECTR - Return Removable Media Vector

The HDVECTR routine will return a one word, bit oriented vector containing information on which drives are removable media drives (i.e., floppy drives). Bit Ø of the word corresponds to Drive A, bit 15 corresponds to Drive P.

Passed: None

Returned: HL - Unit Vector
Bit Ø = Drive A, ..., Bit 15 = Drive P
Ø = Fixed Media, 1 = Removable Media

Side Effects: None

E.DKYBD - Enable/Disable Keyboard Circular Buffering

The E.DKYBD routine will either enable or disable the circular buffering of characters received from the keyboard for the current user.

Passed: C - Ø = Disable Buffering
- FF = Enable Buffering

Returned: None

Side Effects: When the circular buffer is disabled, all characters in the buffer are lost.

E.DCONV - Enable/Disable Keyboard Character Conversion by the Monitor

The E.DCONV routine will either enable or disable the conversion of characters received from the keyboard by the Monitor.

Passed: C - Ø = Disable Conversion
- FF = Enable Conversion

Returned: None

Side Effects: None

LPAGE - Set List Device Auto-Paging Flag

The LPAGE routine will set the currently selected list device's auto-paging flag with the value passed in the C register.

Passed: C - Auto-Paging Value

Returned: None

Side Effects: None

MOVBDOS - Move BDOS Base Address

The MOVBDOS routine will move the BDOS base address to any specified page location in the current user's RAM whenever a warm or cold boot is performed.

Passed: BC - Page Address
Ø = Default Value (ØDEH)

Returned: None

Side Effects: On either a warm or cold boot, the BDOS entry jump at location 5 and the user's copy of the BDOS are updated to reflect the specified location.

RBUFSYS - Reset Buffer System

The RBUFSYS routine will flush and reset any buffers being maintained for the unit specified. This includes the writing of the WRITE buffers and the resetting of either the directory or data buffers associated with the unit.

Passed: BC - Unit Vector:
Bit Ø = Drive A, ... Bit 15 = Drive P
1 = Reset, Ø = No Reset

Returned: A - Physical Driver Error Code

TASKWAIT - Place Task into a Wait State

The Taskwait routine will place a specified task into a wait state until a specified condition occurs. This routine is useful in improving system resopnse time when waiting for a condition to occur.

Passed: C - Task Number
0 = Current task
1 = Despooler
2 = User 1
3 = User 2
.
.
B - Wait Expression
0 = Byte Constant
1 = Word Constant
2 = Not Byte
3 = Not Word
4 = Bit 0 to become Reset
5 = Bit 1 to become constant
.
.
11 = Bit 7 to become Reset
12 = Bit 0 to become set
13 = Bit 1 to become set
.
.
19 = Bit 7 to become set
DE - Wait Expression Equality (WEE)
HL - Wait Expression Address (WEA)

In order to use this routine the user must specify the task number to place into a wait state and the type of wait state expression. i.e. wait for a byte/or word at the WEA to equal the WEE, or a bit to at the WEA to become set on Reset. If the task number is the current task, execution will resume once the wait condition has been met at the instruction immediately after the call to this routine.

5.5 CP/M TABLES AND VARIABLES

5.5.1 Reserved Locations in Page Zero

Main memory, page zero, between locations ØØH and ØFFH contains various segments of code and data which are used during processing. This information is set up for normal operation under VECTOR 4 CP/M, but can be overwritten by a transient program if the BDOS facilities are not required by the transient. For example, if a particular program performs only simple I/O and must begin execution at location Ø, it can be first loaded into the TPA using a small memory move program which will get control when loaded (the 'move' program must get control from location Ø1ØØH, the assumed beginning of all transient programs). The move program will then proceed to move the entire memory image down to location Ø and pass control to the starting address of the memory load. If the BIOS is overwritten, the programmer must bring the system back into memory with a warm boot sequence. The code and data areas for main memory are shown for reference in the following:

ØØØØH - ØØØ2H Contains a jump instruction to the warm boot entry point at location F8Ø3H to allow a simple programmed restart.

ØØØ3H - ØØØ3H Contains the Vector standard IOBYTE.

ØØØ4H - ØØØ4H Contains the current default drive number and user number; the high nybl equates to the user number, the low nybl equates to the drive code (User numbers: Ø to 15, Drives: Ø = A,..., 15 = P).

ØØØ5H - ØØØ7H Contains a dual-purpose jump instruction to the BDOS: JMP ØØØ5H provides the primary entry point to the BDOS, and LHLD ØØØ6H brings the address field of the instruction to the HL register pair; this value is the lowest address in memory used by CP/M. NOTE: the DDT program will change the address field to reflect the reduced memory size in debug mode.

ØØ8H - ØØ27H Restart locations 1 through 4 not currently used.

ØØ3ØH - ØØ37H Restart location 6 not currently used.

ØØ38H - ØØ3AH Contains restart 7, a jump instruction into the DDT or RAID program when running in debug mode for programmed breakpoints; however, it is not used by CP/M for any other reason.

ØØ3BH - ØØ5BH Not currently used - reserved.

ØØ5CH - ØØ7CH Contains the default FCB produced for a transient program by the CCP.

ØØ7DH - ØØ7FH Contains the optional default random record position.

ØØ8ØH - ØØFFH Contains the default 128-byte disk buffer; also filled with the command line when a transient is loaded under the CCP.

5.5.2 IOBYTE

VECTOR 4 CP/M has an implementation of the IOBYTE (VECTOR 4 CP/M does not used Intel standard) which defines the logical to physical device mapping for Character I/O devices which is in effect at a particular time for an individual user. The mapping is performed by splitting the IOBYTE into four distinct fields of two bits each, called the CONSOLE, READER, PUNCH, and LIST fields, as shown below:

	Most Significant		Least Significant	
IOBYTE at ØØØ3H	LIST	PUNCH	READER	CONSOLE
	bits 6,7	bits 4,5	bits 2,3	bits Ø,1

The value in each field (the definitions of which have been changed from the standard Intel definitions) can be in the range Ø-3, defining the assigned source or destination of each logical device. The values which can be assigned to each field are given as follows:

CONSOLE field (bits Ø,1)

Ø - Console is assigned to the console device
1 - Console output is echoed to the print device
2 - No console - bit bucket
3 - No console - suspend until console available

CONSOLE

MODE	0	1	2	3
ROUTINE				
CONST	Normal Returns status in A. Ready = 0FFH Not Ready = 00H	Normal See Mode 0	Not Ready = 00H	Program is suspend until console is attached to prog operation contin as Normal.
CONIN	Normal Character returned in A	Normal See Mode 0	No Character (Null) 00H	Same as above
CONOUT	Normal Character placed onto screen	Normal See Mode 0	Character Discarded	Same as above

READER field (bits 2,3)

0 - No READER
1 - READER input from console
2 - User defined reader #1
3 - User defined reader #2

READER

MODE	No character (Null) 00H	See CONSOLE 'CONIN'	Normal Character returned in A	Same as Mode

PUNCH field (bits 4,5)

Ø - No PUNCH
1 - PUNCH is assigned to console
2 - PUNCH is assigned to list device
3 - User defined punch

PUNCH

MODE	Character Discarded	See CONSOLE 'CONOUT'	See LIST 'LIST'	Normal Character returned in A.

LIST field (bits 6,7)

Ø - No LIST
1 - LIST is assigned to console
2 - LIST is list device #1
3 - LIST is list device #2

LIST

MODE	Ø	1	2	3
LIST	Character discarded	See CONSOLE 'CONOUT'	Normal Character printed on printe	
LISTST	Not ready ØØH	Always ready ØFFH	Normal Output status of printer 1 returned in register A	Normal
LINCHAR	No character (Null) ØØH	See CONSOLE 'CONIN'	Normal Character received from printer 1 in A	Normal Character received f printer 2
LINSTAT	Not ready ØØH	See CONSOLE 'CONST'	Normal Input status of printer 1 returned in A	Normal Input stat of printer returned i
LTYPE	No printer HL = Ø	See Mode Ø	Normal Returns type byte in HL of printer 1	Normal Returns ty byte in HL printer 2
LPAGE	No action	No action	Normal Sets page byte for printer 1	Normal Sets page for printe
LINIT	No action	No action	Normal Printer 1 initialized	Normal Printer 2 initialize

If the console output is to be echoed to the list device, and the list device is set to be routed to the console, an endless loop does not occur. The BIOS will always prevent this kind of endless loop condition from occurring, and will send the character to each device only once. The IOBYTE value must be changed through BDOS functions 7 and 8 , otherwise, the system will not recognize the user's modification of the IOBYTE.

The VECTOR 4 CP/M defaults for the IOBYTE are set at Ø and correspond to the following:

- Normal console
- No LIST
- No READER
- No PUNCH

5.5.3 Disk Parameter Tables

Tables are included in the BIOS which describe the particular characteristics of the disk subsystem used with VECTOR 4 CP/M. These tables are automatically generated by the GENSYS and CONFIG programs. Each disk drive has an associated, 18-byte Disk Parameter Header (DPH), which contains information about the disk drive and provides a 'scratchpad' area for certain BDOS operations. The format of each 18-byte DPH element for each drive is shown in the following:

XLT | ØØØØ | ØØØØ | ØØØØ | DIRBUF | DPB | CSV | ALV | SALV

XLT — Address of the logical to physical translation vector, if used for this particular drive, or the value 'ØØØØH' if no sector translation takes place; i.e., the physical and logical sector numbers are the same. Disk drives with identical sector skew factors share the same translate tables.

ØØØØ — Scratchpad values for use within the BDOS; initial value is not important.

DIRBUF — Address of a 128-byte scratchpad area for directory operations within BDOS; all DPHs address the same scratchpad area.

DPB Address of a Disk Parameter Block (DPB) for this particular drive; drives with identical disk characteristics address the same disk parameter block.

CSV Address of a scratchpad area used for software check for changed disks; this address is different for each DPH.

UALV Address of a scratchpad area in the user's bank used by the BDOS to keep disk storage allocation information; this address is different for each DPH.

SALV Address of a scratchpad area in the system bank used by the BDOS to keep disk storage allocation information; this address is different for each DPH.

Given 16 disk drives, the DPHs are arranged in a table where the first row of 16 bytes corresponds to Drive Ø (Drive A), with the last row corresponding to Drive 15 (Drive P). The table therefore appears as:

DPBASE (DPH Base Address)

ØØ	XLT ØØ	ØØØØ	ØØØØ	ØØØØ	DIRBUF	DPB ØØ	CSV ØØ	ALV ØØ
Ø1	XLT Ø1	ØØØØ	ØØØØ	ØØØØ	DIRBUF	DPB Ø1	CSV Ø1	ALV Ø1
Ø2	XLT Ø2	ØØØØ	ØØØØ	ØØØØ	DIRBUF	DPB Ø2	CSV Ø2	ALV Ø2
•								
•								
•								
15	XLT 15	ØØØØ	ØØØØ	ØØØØ	DIRBUF	DBP 15	CSV 15	ALV 15

The SELDSK subroutine has the responsibility to return the base address of the DPH for the selected drive. The translation vectors (XLT ØØ through XLT 15) are located elsewhere in the BIOS and correspond one-to-one with the logical sector numbers zero through 15. The DPB for each drive is more complex; a particular DPB addressed by one or more DPHs takes the general form (each is a byte (8b) or word value (16b)) as shown in the following:

SPT	BSH	BLM	EXM	DSM	DRM	ALØ	AL1	CKS	OFF	PSPT
16b	8b	8b	8b	16b	16b	8b	8b	16b	16b	8b

SPT Corresponds to the total number of logical sectors per track.

BSH	Corresponds to the data allocation block shift factor; determined by the data block allocation size.
EXM	Corresponds to the extent mask; determined by the data block allocation size and the number of disk blocks.
DSM	Determines the total storage capacity of the disk drive (in blocks).
DRM	Determines the total number of directory entries which can be stored on this drive minus 1.
AL0, AL1	Determine reserved directory blocks.
CKS	Corresponds to the size of the directory check vector.
OFF	Corresponds to the number of reserved tracks at the beginning of the logical disk.
PSPT	Corresponds to the number of physical sectors per track.

The values of BSH and BLM implicitly determine the data allocation size BLS, which is not an entry in the disk parameter block. For example, if the selected value for the BLS is given as shown below, the values of BSH and BLM can then be determined (all values are in decimal):

BLS	BSH	BLM
1,024	3	7
2,048	4	15
4,096	5	31
8,192	6	63
16,384	7	127

The value of EXM depends upon both the BLS and whether the DSM value is less than 256 or greater than 255, as shown in the following:

BLS	DSM < 256	DSM > 255
1,024	0	N/A
2,048	1	0
4,096	3	1
8,192	7	3
16,384	15	7

The value of the DSM is the maximum data block number supported by this particular drive, measured in BLS units. The product of 'BLS(DSM+1)' is the total number of bytes held by the drive; this must be within the capacity of the physical drive not counting the reserved operating system tracks.

The DRM entry is one less than the total number of directory entries, which can take on a 16-bit value; however, the values of ALØ and AL1 are determined by the DRM. The two values, ALØ and AL1, can together be considered a string of 16 bits, as shown in the following:

ALØ								AL1							
ØØ	Ø1	Ø2	Ø3	Ø4	Ø5	Ø6	Ø7	Ø8	Ø9	1Ø	11	12	13	14	15

Position ØØ corresponds to the high-order bit of the byte labeled ALØ, and position 15 corresponds to the low-order bit of the byte labeled AL1. Each bit position reserves a data block for a number of directory entries, thereby allowing a total of 16 data blocks to be assigned for directory entries (bits are assigned beginning with 'ØØ' and filled in to the right until position 15). Each directory entry occupies 32 bytes, which results in the following table:

BLS	Directory Entries
1,Ø24	32(# of bits)
2,Ø48	64(# of bits)
4,Ø96	128(# of bits)
8,192	256(# of bits)
16,384	512(# of bits)

Therefore, if DRM = 127 (128 directory entries) and BLS = 1Ø24, there are 32 directory entries per block, requiring four reserved blocks; here, the four high-order bits of ALØ are set, resulting in the he values ALØ = ØFØH and AL1 = ØØH.

To determine the CKS value, if the disk drive media is removable, then CKS = (DRM+1)/4, where DRM is the last directory entry number; if the media is fixed, then CKS = Ø (no directory records are checked in this case).

The OFF field determines the number of tracks which are skipped at the beginning of the physical disk. This value is automatically added whenever SETTRK is called, and can be used as a mechanism for skipping reserved operating system tracks, or for partitioning a large disk into smaller segmented sections.

Several DPHs can address the same DPB if their drive characteristics are identical; furthermore, the DPB can be dynamically changed when a new drive is addressed by changing the pointer in the DPH since the BDOS copies the DPB values to a local area whenever the SELDSK function is called. Note that the three address values CSV, UALV, and SALV remain when returning back to the DPH for a particular drive. Both addresses reference an area of uninitialized memory following the BIOS. The areas must be unique for each drive, and the size of each area is determined by the values in the DPB.

The size of the area addressed by the CSV is CKS bytes; this is sufficient to hold the directory check information for this particular drive. If CKS = (CRM+1)/4, then (DRM+1)/4 bytes just be reserved for directory check use; if CKS = Ø, no storage is reserved.

The size of the area addressed by UALV and SALV is determined by the maximum number of data blocks allowed for this particular disk and is computed as (DSM/8)+1. The UALV and SALV values should never be changed by any program.

5.6 VIDEO DRIVER

The VECTOR 4 CP/M contains a powerful video driver, it's purpose being to accept a stream of ASCII codes and to write them into the screen memory in the proper place, interpreting certain non-printing control codes in a special way. Access to the video driver can be obtained in several ways; a BIOS console output call or BDOS direct console I/O call, function 6, with the control or ASCII character in the appropriate register will send the character directly to the video driver within CP/M.

The video driver has been modified to sound the speaker contained in the Vector 4. The bell code (Ø7H) will be trapped by the video driver and BDOS function 222 will be called to produce a 75Ø Hz tone for 1/4 of a second period. If a longer bell is required, repeated bell codes (output using the video driver) will cause the timer to be reset.

The following control codes are interpreted as special functions, while all others are ignored:

DECIMAL VALUE	HEX VALUE	CONTROL CODE	CODE DESCRIPTION
2	2	[CTRL B]	HOME CURSOR
4	4	[CTRL D]	CLEAR SCREEN AND HOME CURSOR
5	5	[CTRL E]	DISPLAY CODE IN B REGISTER
7	7	[CTRL G]	BELL
8	8	[CTRL H]	DESTRUCTIVE BACKSPACE (BACKSPACE KEY)
9	9	[CTRL I]	TAB TO NEXT 8 MULTIPLE (TAB KEY)
10	A	[CTRL J]	LINEFEED ([LF] KEY)
13	D	[CTRL M]	CARRIAGE RETURN
14	E	[CTRL N]	TOGGLE CURSOR
16	10	[CTRL P]	CLEAR TO END OF SCREEN
17	11	[CTRL Q]	CLEAR TO END OF LINE
18	12	[CTRL R]	CURSOR DOWN
20	14	[CTRL T]	TOGGLE REVERSE VIDEO
21	15	[CTRL U]	CURSOR UP
23	17	[CTRL W]	CURSOR LEFT
24	18	[CTRL X]	CLEAR TO START OF LINE
26	1A	[CTRL Z]	CURSOR RIGHT
27	1B	[ESC]	CURSOR XY POSITION LEAD-IN or TOP OF SCREEN LEAD-IN

There are special keys on the keyboard to generate some of the codes, e.g., [RETURN], [TAB], [LF] (linefeed), [BACKSPACE], [ESC], [DEL], and cursor arrows. A few of the control codes are not too self-explanatory; a [CTRL D] will set the reverse video flag to normal in addition to clearing the screen and homing the cursor. A [CTRL T] will then toggle the reverse video flag from normal to reverse and back without printing on the screen.

CURSOR X-Y POSITIONING

Many programs utilize random X-Y positioning of the cursor. This is done by outputting a three-byte sequence to the video driver. The first code is an [ESC] (1BH) followed by the desired X and Y positions in hex. The top left corner of the screen is Ø, Ø. The hex sequence '1B 4Ø Ø8' would cause the cursor to move to line 8, character position 64 on the screen.

To send the same sequence to the video driver via Microsoft Basic, the following statement would be used:

PRINT CHR$(27);CHR$(X+128);CHR$(Y+128);

where 'X' equals 64 (4ØH) and 'Y' equals Ø8 (Ø8H). Adding the value of 128 to X and Y in this example sets the eighth bit high. This is done to avoid Microsoft Basic from confusing the values as control codes.

The video driver provides an extensive range of special controls; however, they must be incorporated into the software generating the video stream to be meaningful. For instance, a piece of software that merely echoes all characters as they go into its input buffer will allow cursor motion on the screen, but this will probaly be meaningless to the software.

SETTING TOP OF SCREEN

The logical top of screen can be set by sending the appropriate codes (escape sequences) to the video driver. To set the top of screen, send the following to the video driver:

1B 7F <line number>

The line number must be expressed in hex; in the range of ØH to 16H (Ø to 22 decimal).

APPENDIX A - VECTOR 4 CP/M FILE SYSTEM ERRORS

The BDOS file system has an extensive error handling capability, which, when an error is detected, can respond in one of two ways:

1. It can display an error message on the console and abort the process.

2. It can return to the calling program with return codes in register pair HL identifying the error (as previously described: register A = L and register B = H).

There are two types of errors the file system handles: 'physical' and 'logical' errors. Physical errors pertain directly to hardware limitations and/or malfunctions; logical errors pertain to system limitations or illegal usage of a function.

The SET BDOS ERROR MODE function (function 45) determines how the file system handles physical and logical errors. The BDOS Error Mode can exist in two states: Default and Return Error modes. In the Default state, the BDOS displays the error message on the user's console and terminates the calling process (method 1). In the Return Error mode (method 2), the BDOS returns control to the calling process with the error identified in registers H and L. The Return mode ensures that CP/M does not terminate the process because of a physical or logical error.

A. Method 1

During a BDOS function call, if a physical error is encountered during disk I/O, a message is displayed on the console and program execution is terminated and control is transferred to the CCP. Physical and logical errors are displayed on the console in the following format:

ERROR - drive:<error>

where <error> identifies the error. Following is a list of the error messages displayed as a result of a physical error:

DISK READ/WRITE(xxyy)

The DISK READ/WRITE message will occur when the disk controller electronics has detected an error condition in reading or writing to the diskette ('xx' is a hex code denoting the particular physical BIOS error detected by the physical disk driver; 'yy' corresponds to the BDOS physical

and logical error codes). This condition is generally due to a malfunctioning in the disk controller or the result of an extremely worn diskette. If your system reports this error more than once a month, it may require servicing or your media may need replacement. Recovery from this condition may be accomplished by responding to the next prompt:

PRESS [CTRL C] TO REBOOT OR [RETURN] TO CONTINUE

If a [CTRL C] is entered, the current program will be aborted and control returned to the CCP; if a [RETURN] is entered, the error will be ignored by the program and execution will continue normally. Pressing [RETURN] would only be used when copying a file (using PIP), when it is mandatory that as much of the file is recovered as possible. Note, however, that entering a [RETURN] may destroy diskette integrity if the operation is a directory write; therefore, make sure an adequate backup has been made.

NOTE: The following BDOS errors will display the particular error message and simply wait for user input to reboot; entering any key will cause a warm boot.

DRIVE SELECT

The DRIVE SELECT message will occur when there is an attempt to access a drive this is not supported by the current hardware/software configuration. In this case, the drive which is out of range is displayed along with the error message. The system reboots following any input from the terminal.

DRIVE NOT READY

The DRIVE NOT READY message will occur when an attempt is made to read from a drive and the disk is either not properly seated or there is no disk in the drive.

DISK IS WRITE PROTECTED

The DISK IS WRITE PROTECTED message will occur when there is an attempt to write to a disk after the disk has been write-protected by use of a write-protect tab on the diskette.

DISK IS READ-ONLY

The DISK READ-ONLY message will occur when there is an attempt to write to a diskette which has been designated as 'READ/ONLY' using the STAT command, or has been set to 'READ-ONLY' by the BDOS. In general, the user should warm boot the system, using [CTRL C], whenever physically changing a diskette. If a changed diskette is to be read but not written to,

the system allows the diskette to be changed without requiring a warm boot, but internally marks the drive as Read-Only. The status of the drive is subsequently changed back to 'Read/Write' following a warm boot. Upon displaying the READ-ONLY message, CP/M waits for an input from the console; an automatic warm boot will occur following any user input.

FILE IS READ-ONLY

The FILE READ-ONLY message will occur when there is an attempt to write to a file which has been declared as 'R/O' using the STAT command or by an applications program, or if the file is opened in READ-ONLY mode. The file may be returned to READ/WRITE status using the STAT command, if necessary.

Following is a list of the error messages displayed as a result of a logical error:

FILE CURRENTLY OPEN

The FILE CURRENTLY OPEN message will occur if an attempt is made to access a file which is in use and locked by another user.

ILLEGAL FILE ACCESS

The ILLEGAL FILE ACCESS message will occur when an attempt is made to access a file that is not open.

FILE ALREADY EXISTS

The FILE ALREADY EXISTS message will occur when a user or program is trying to create, rename, or spool a specific file which already exists on disk.

ILLEGAL '?' IN FCB

The ILLEGAL '?' IN FCB message will occur when a question mark '?' is used within the FCB in any file function other than SEARCH FIRST (function 17), SEARCH NEXT (function 18), and DELETE FILE (function 19).

FILE OPEN LIST FULL .

The FILE OPEN LIST FULL message will occur when an attempt is made to OPEN (function 15) or CREATE (function 22) a file and the System File List is full (32 files maximum - list is maintained by the operating system of open files in the system, the task(s) which have the files open, and the mode in

which the file is open).

RECORD LOCK LIST FULL

The RECORD LOCK LIST FULL message will occur when an attempt is made to LOCK a record (function 42) and the Record Lock List is full (512 maximum - series of lists linked to the open file entries of the System File List describing which records are locked, and which task has them locked). NOTE: records can only be locked in unlocked mode.

B. Method 2

By using SET BDOS ERROR MODE (function 45), it is possible to inhibit error messages and to cause execution to return to the calling program with the physical or logical error codes in register H and L.

The BDOS will return the following logical error codes in register L:

ØØ - FUNCTION SUCCESSFUL

Ø1 - (Read mode): **READING UNWRITTEN DATA**

Read past end of file or attempt to access unallocated data block.

(Write mode): **NO DIRECTORY SPACE**

Cannot open new extent.

Ø2 - NO AVAILABLE DATA BLOCKS (Not returned in random mode)

Occurs when the write command attempts to allocate a new data block to the file and no unallocated data blocks exist on the selected drive.

Ø3 - CANNOT CLOSE CURRENT EXTENT

Occurs when the current extent cannot be closed prior to moving to a new extent.

Ø4 - SEEK TO UNWRITTEN EXTENT

Attempted to access an extent which has not been created.

Ø5 - NO AVAILABLE DIRECTORY SPACE (Write mode only)

Occurs when the write function attempts to create a new extent that requires a new directory entry and no available directory entries exist on the selected drive.

06 - SEEK PAST PHYSICAL END OF DISK (random record number out of range)

Occurs when byte 35 (r2) of the referenced FCB is greater than three.

07 - RECORD DID NOT MATCH

Used by TEST AND WRITE RECORD function to indicate non-identical records.

08 - RECORD LOCKED BY ANOTHER TASK

Occurs when the WRITE FUNCTION attempts to write to a record locked by another task. This error is only returned for files open in unlocked mode.

09 - INVALID FCB

Occurs when the previous BDOS Read or Write call returned an error code and invalidated the FCB.

10 - INVALID FILE ACCESS

Occurs when an attempt has been made to access a file that has not been opened (is not contained within the Open File list).

11 - UNLOCKED FILE VERIFICATION ERROR

Occurs if the BDOS cannot locate the FCB's directory entry when attempting to verify that the referenced FCB contains current information. This error is only returned for files open in unlocked mode.

12 - NO ROOM IN RECORD LOCK LIST

Occurs when the sum of the number of records currently locked by the calling program and the number of records to be locked by the LOCK RECORD call exceeds the maximum allowed value.

14 - NO ROOM IN THE OPEN FILE LIST

Occurs when the Open File list is full.

255 - PHYSICAL ERROR (Refer to register H)

Occurs when a physical error is encountered. For further information about the error, check the H register or call function 225, RETURN BIOS ERROR CODE; the low-order nybl of the H register will contain the physical or logical error code.

REGISTER L RETURN CODES

For functions that return error codes, the BDOS sets the high-order nybl to the number of sectors successfully read or written before the error is encountered. This information is returned only when a process uses the SET MULTI-SECTOR COUNT function to set the BDOS logical sector count to a value other than '1'; otherwise, the BDOS sets N1 to zero. Successful read and write functions also set the high-order nybl to zero; functions that return a Directory code or an Error Flag set it to zero.

The BDOS returns logical or physical error codes in the low-order nybl only if it is in the return error mode; otherwise, it will be set to zero. Following is a list of the error codes that are returned in the low-order nybl of register L:

00 NO ERROR OR NOT A REGISTER H ERROR
01 BAD SECTOR - PERMANENT ERROR
02 READ-ONLY DISK
03 READ-ONLY FILE
04 DRIVE SELECT ERROR
05 FILE CURRENTLY OPEN
06 CLOSE CHECKSUM ERROR
08 FILE ALREADY EXISTS
09 ILLEGAL '?' IN FCB
10 NO ROOM IN OPEN FILE LIST
11 NO ROOM IN RECORD LOCK LIST

NOTE: Register L is equal to zero if the called function is successful. In addition, the BDOS sets the low-order nybl of register L to zero when register L returns a value other than '255'. Except for functions that return Directory codes, if register L contains a value of '255' upon return, the low-order nybl identifies the error when the BDOS is in return error mode.

BDOS DIRECTORY CODES

00 - 03 SUCCESSFUL FUNCTION
255 UNSUCCESSFUL FUNCTION

With the exception of the BDOS Search functions, Directory code values (Ø-3) have no significance other than to indicate a successful result. However, for the SEARCH functions, a successful Directory code identifies the relative starting position of the directory element in the calling program's current DMA buffer. Following is a list of the BDOS functions that return a Directory code value:

Function 15 - OPEN FILE
Function 16 - CLOSE FILE
Function 17 - SEARCH FOR FIRST
Function 18 - SEARCH FOR NEXT
Function 19 - DELETE FILE
Function 22 - MAKE FILE
Function 23 - RENAME FILE
Function 30 - SET FILE ATTRIBUTES

NOTE: Because the Directory code values conflict with the BDOS error code values, if a value between 1 and 3 is returned it can be assumed that it is a Directory code value and not a BDOS error code value.

There is one last alternate method for handling BDOS error codes that is implemented in VECTOR 4 CP/M in order to maintain compatibility with previous versions of Vector's CP/M. This method uses the BDOS functions 225, RETURN BIOS ERROR CODE, and 226, INHIBIT/ENABLE BDOS ERRORS; by disabling the BDOS error reporting with function 226, the programmer should then call function 225 after every access to the BDOS file system for error reporting. Function 225 will return either a BIOS disk error code or one of the following:

80 - BDOS file access error
81 - BDOS file or record list error

BIOS DISK DRIVER ERRORS - REGISTER H

HD = Hard Disk
FD = Floppy Disk

11 **DRIVE NOT READY** - HD - Internal mechanical problem.
12 **WRITE PROTECTED** - FD - Write-Protect tab is being used; diskette is in wrong.
13 **WRITE FAULT** - HD - internal but no external write.
14 **TRACK Ø NOT FOUND** - HD, FD - Hardware problem.

21 **CONTROLLER BUSY TIMEOUT** - HD, FD - Floppy door is open; disk drive or controller board problems; fluctuation speed of hard disk.
22 **SEEK COMPLETE TIMEOUT** - HD - Not able to step to the next track within certain amount of time.
23 **LOSS OF SYNC TIMEOUT** - HD - Drive spinning either too fast or too slow.

31 **BAD COMMAND CODE** - HD, FD - Bad command format.
32 **BAD UNIT VALUE** - HD, FD - Bad unit number.
33 **BAD SECTOR VALUE** - HD, FD - Bad sector number.
34 **BAD TRACK VALUE** - HD, FD - Bad track number.

41 **TRACKS DON'T MATCH** - 41-44 HD - Bad matches.
42 **SECTORS DON'T MATCH**
43 **HEADS DON'T MATCH**
44 **DATA DOESN'T MATCH**

51-54 must be set using BIOS SOFTERROR routine.

51 **READ CKSUMS DON'T MATCH** - FD - Checksum values don't match.
52 **READ/WRITE CKSUMS DON'T MATCH** - FD - Checksum values don't match.
53 **ECC ERROR** - HD, FD - ECC or disk not the same as physical driver calculation.
54 **UNCORRECTABLE ECC ERROR** - HD, FD - Needs 2 out of 5 retrys to correct.
55 **UNCORRECTABLE SPASM ERROR** - HD, FD -
56 **SYNC BYTE ERROR** - HD, FD - First byte of each sector is not 'FF' and must be.

E2 **DISABLED AND BUSY** - Physical driver called for second time and is currently busy; results from system crash, FORMAT, or DISKCOPY.

APPENDIX B - OCTAVE SETTINGS FOR TONE GENERATORS

NOTE	FREQUENCY	SETTING	HEX	ACTUAL FREQUENCY
OCTAVE 1				
C	65.4	956	3BC	65.4
C#	69.3	9Ø2	386	69.3
D	73.4	851	353	73.4
D#	77.8	8Ø4	324	77.7
E	82.4	758	2F6	82.5
F	87.3	716	2CC	87.3
F#	92.5	676	2A4	92.5
G	98.Ø	638	27E	98.Ø
G#	1Ø3.8	6Ø2	25A	1Ø3.8
A	11Ø.Ø	568	238	11Ø.Ø
A#	116.5	536	218	116.6
B	123.5	5Ø6	1FA	123.5
OCTAVE 2				
C	13Ø.8	478	1DE	13Ø.8
C#	138.6	451	1C3	138.6
D	146.8	426	1AA	146.7
D#	155.6	4Ø2	192	155.5
E	164.8	379	17B	164.9
F	174.6	356	166	174.6
F#	185.Ø	338	152	184.9
G	196.Ø	319	13F	195.9
G#	2Ø7.7	3Ø1	12D	2Ø7.6
A	22Ø.Ø	284	11C	22Ø.1
A#	233.1	268	1ØC	233.2
B	246.9	253	FD	247.Ø

OCTAVE 3

C	261.6	239	EF	261.5
C#	277.2	225	E1	277.8
D	293.7	213	D5	293.4
D#	311.1	201	C9	310.9
E	329.6	190	BE	328.9
F	349.2	179	B3	349.2
F#	370.0	169	A9	369.8
G	392.0	159	9F	393.1
G#	415.3	150	96	416.7
A	440.0	142	8E	440.1
A#	466.2	134	86	466.4
B	493.9	127	7F	492.1

OCTAVE 4

C	523.3	119	77	525.2
C#	554.4	113	71	553.1
D	587.3	106	6A	589.6
D#	622.3	100	64	625.0
E	659.3	95	5F	657.9
F	698.5	89	59	702.2
F#	740.0	84	54	744.0
G	784.0	80	50	781.3
G#	830.6	75	4B	833.3
A	880.0	71	47	880.3
A#	932.3	67	43	932.8
B	987.8	63	3F	992.1

OCTAVE 5

C	1046.5	60	3C	1041.7
C#	1108.7	56	38	1116.1
D	1174.7	53	35	1179.2
D#	1244.5	50	32	1250.0
E	1318.5	47	2F	1329.8
F	1396.9	45	2D	1388.9

F#	148Ø.Ø	42	2A	1488.1
G	1568.Ø	4Ø	28	1562.5
G#	1661.2	38	26	1644.7
A	166Ø.Ø	38	26	1644.7
A#	1758.7	36	24	1736.1
B	1863.3	34	22	1838.2

OCTAVE 6

C	1974.1	32	2Ø	1953.1
C#	2Ø91.5	3Ø	1E	2Ø83.3
D	2215.8	28	1C	2232.1
D#	2347.6	27	1B	2314.8
E	2487.2	25	19	25ØØ.Ø
F	2635.1	24	18	26Ø4.2
F#	2791.8	22	16	284Ø.9
G	2957.8	21	15	2976.2
G#	3133.7	2Ø	14	3125.Ø
A	332Ø.Ø	19	13	3289.5
A#	3517.4	18	12	3472.2
B	3726.6	17	11	3676.5

OCTAVE 7

C	3948.2	16	1Ø	39Ø6.3
C#	4182.9	15	F	4166.7
D	4431.7	14	E	4464.3
D#	4695.2	13	D	48Ø7.7
E	4974.4	13	D	48Ø7.7
F	527Ø.2	12	C	52Ø8.3
F#	5583.6	11	B	5681.8
G	5915.6	11	B	5681.8
G#	6267.3	1Ø	A	625Ø.Ø