VECTOR

CP/M-86

Programmer's Guide

CP/M-86

PROGRAMMER'S GUIDE

Revision 00

May 20, 1983

Manual P/N 7100-0023

Revision Numbers and Updates

The date, revision and part number of each page appears on its bottom line. The revision number (which may range from 00 to 99) is appended to the 8 digit Vector part number by a hyphen. The date and revision on the Title Page corresponds to that of the page most recently revised. In addition a page listing the latest revision level of each page is included before the Table of Contents. THIS MANUAL SHOULD ONLY BE USED WITH THE PRODUCT(S) IDENTIFIED ON THE TITLE PAGE.

WARRANTY AGREEMENT

Vector warrants to the authorized dealer that products manufactured by Vector will be free from defects in material and workmanship for a period of ninety (90) days following delivery to the end-user.

Vector's obligation under the warranty is limited to replacing or repairing, at its option, at its factory, products that, within the warranty period, are returned prepaid and insured to Vector and that are found by Vector to be defective. Return authorization must be obtained from Vector Customer Support before returning products. The repaired or replacement product will be returned prepaid to the dealer.

This warranty shall immediately be null and void if, in Vector's sole judgement, the product has been subjected to misuse, abuse, neglect, accident, improper installation, alterations, modifications, including failure to maintain environmental conditions, or use supplies that do not meet specifications recommended by Vector; or external causes such as electrical power fluctuations and failures, floods, windstorms and other acts of God, or if the serial number and/or product markings have been removed, defaced, or altered.

This warranty agreement is void if the warranty form is not returned to Vector within ten (10) days of end-user purchase. In such event, repair or alterations will be rendered only on special order by the customer and after approval by the customer of the estimated additional charge.

THE FOREGOING WARRANTY IS EXCLUSIVE AND IN LIEU OF ALL OTHER WARRANTIES EITHER EXPRESSED OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

In no event shall Vector be liable for incidental or consequential damages or economic loss arising out of or related to the product or services provided.

MANUAL REVISIONS

Page	Revision	Date	Page	Revision	Date
i	00	05-20-83	4-1	00	05-20-83
ii	00	05-20-83	4-2	00	05-20-83
iii	00	05-20-83	4-3	00	05-20-83
iv	00	05-20-83	4-4	00	05-20-83
v	00	05-20-83	4-5	00	05-20-83
vi	Blank		4-6	00	05-20-83
vii	00	05-20-83	4-7	00	05-20-83
viii	Blank		4-8	00	05-20-83
ix	00	05-20-83	4-9	00	05-20-83
x	00	05-20-83	4-10	00	05-20-83
			4-11	00	05-20-83
			4-12	00	05-20-83
			4-13	00	05-20-83
1-1	00	05-20-83	4-14	00	05-20-83
1-2	00	05-20-83	4-15	00	05-20-83
1-3	00	05-20-83	4-16	00	05-20-83
1-4	00	05-20-83	4-17	00	05-20-83
1-5	00	05-20-83	4-18	00	05-20-83
1-6	Blank		4-19	00	05-20-83
			4-20	00	05-20-83
			4-21	00	05-20-83
2-1	00	05-20-83	4-22	00	05-20-83
2-2	00	05-20-83	4-23	00	05-20-83
2-3	00	05-20-83	4-24	00	05-20-83
2-4	00	05-20-83	4-25	00	05-20-83
2-5	00	05-20-83	4-26	00	05-20-83
2-6	00	05-20-83	4-27	00	05-20-83
2-7	00	05-20-83	4-28	00	05-20-83
2-8	00	05-20-83	4-29	00	05-20-83
2-9	00	05-20-83	4-30	00	05-20-83
2-10	00	05-20-83	4-31	00	05-20-83
2-11	00	05-20-83	4-32	00	05-20-83
2-12	00	05-20-83	4-33	00	05-20-83
2-13	00	05-20-83	4-34	00	05-20-83
2-14	00	05-20-83	4-35	00	05-20-83
			4-36	00	05-20-83
3-1	00	05-20-83	4-37	00	05-20-83
3-2	00	05-20-83	4-38	00	05-20-83
3-3	00	05-20-83	4-39	00	05-20-83
3-4	00	05-20-83	4-40	00	05-20-83
3-5	00	05-20-83	4-41	00	05-20-83
3-6	00	05-20-83	4-42	00	05-20-83
3-7	00	05-20-83	4-43	00	05-20-83
3-8	00	05-20-83	4-44	00	05-20-83
3-9	00	05-20-83	4-45	00	05-20-83
3-10	Blank		4-46	00	05-20-83

MANUAL REVISIONS (Cont.)

Page	Revision	Date
4-47	00	05-20-83
4-48	00	05-20-83
4-49	00	05-20-83
4-50	00	05-20-83
4-51	00	05-20-83
4-52	00	05-20-83
4-53	00	05-20-83
4-54	00	05-20-83
4-55	00	05-20-83
4-56	00	05-20-83
4-57	00	05-20-83
4-58	00	05-20-83
4-59	00	05-20-83

FOREWORD

Audience	The CP/M-86 Programmer's Guide is designed for System Programmers.
Scope	The CP/M-86 Programmer's Guide describes all aspects of the CP/M-86 Operating System. This Guide also discusses the Vector 4 Simulator (for 8-bit software) and specific technical information about the Video and Device Drivers.
Organization	The CP/M-86 Programmer's Guide is organized in four sections. The first section gives a general overview of supplementary documentation and how it can be used with this manual. The next two sections describe the Vector 4, 16-bit Video Driver/Console Input and Device Drivers. The last section describes the Vector 4 CP/M Simulator.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

SECTION I - OVERVIEW OF OPERATING SYSTEM DOCUMENTATION

1.1 Vector 4 Operating Systems

The Vector 4 can use either Vector 4 CP/M or CP/M-86. Vector 4 CP/M is designed to use the 8-bit, Z-80 microprocessor while CP/M-86 will only work with the 16-bit 8088 CPU.

1.2 Types of Operating System Documentation

The Operating System Documentation can be divided into two general areas:

-Documentation which describes system software for Vector 4 CP/M
-Documentation which describes system software for CP/M-86

These two general areas can each be divided into User's and Programmer's Documentation. The User's Documentation consists of Vector's CP/M-86 User's Guide, Vector's Vector 4 CP/M Programmer's Guide and Digital Research's CP/M-86 Operating System User's Guide. These Guides describe several features of the CP/M-86 Operating System and the Vector 4 CP/M Operating System. This includes a complete description of the resident and transient commands, loading the Operating System and an explanation of Vector's Disk Partitioning System. Note: The Disk Partitioning system only applies to 16-bit operating system.

The Programmer's Documentation consists of this manual along with several other manuals:

-Digital Research's CP/M-86 Operating System- Programmer's Guide
-Digital Research's CP/M-86 Operating System- System Guide
-Vector's Vector 4 CP/M Programmer's Guide
-Vector's SCOPE Reference Manual
-Vector's ZSM Assembler For CP/M Manual
-Digital Research's CP/M Dynamic Debugging Tool User's Guide
-Microsoft BASIC Interpreter Reference Manual

The first three manuals (including this manual) are included in the set of programmer's documentation for the CP/M-86 Operating System. The next five manuals are arranged in a second set of documentation. This second set of documentation covers programmer's information for the Vector 4 CP/M Operating System.

1.3 How to Use User's and Programmer's Documentation

All operating system documentation is included with your Vector 4. If you are using Vector 4 CP/M then you would want to reference the users and programmers guides for that operating system:

-Vector 4 CP/M User's Guide
-Vector 4 CP/M Programmer's Guide

Also, you could reference other manuals for specific information on system programs:

-Vector's SCOPE Reference Manual
-Vector's ZSM Assembler For CP/M Manual
-Digital Research's CP/M Dynamic Debugging Tool User's Guide
-Microsoft BASIC Interpreter Reference Manual

If you are using the CP/M-86 Operating System you would want to reference those manuals which apply to the CP/M-86 Operating System. These include:

-Vector's CP/M-86 User's Guide
-Digital Research's CP/M-86 Operating System- Programmer's Guide
-Digital Research's CP/M-86 Operating System- System Guide
-Vector's CP/M-86 Programmer's Guide

The CP/M-86 Operating System uses the same resident (built-in) and transient commands that are used by the CP/M-80 Operating System. If you are familiar with CP/M-80's commands, you should be able to easily adapt to the CP/M-86 environment. If you aren't familiar with CP/M-80 Commands, see the User's Guide for a complete discussion of these commands.

As explained in the previous section, Vector's CP/M-86 User's Guide contains an explanation of Operating System loading procedures and Vector's Disk Partitioning System. All CP/M-86 users and programmers should review these sections.

It should be noted that Digital Research's CP/M-86 Programmer's Guide describes system software that can only be used with CP/M-86. i.e., ASM-86 and DDT-86. Digital Research's CP/M-86 System Guide includes an extensive discussion of BDOS Functions and BIOS Jump Routines for the CP/M-86 Operating System.

The next section describes the contents of this manual.

1.4 How to Use Vector's CP/M-86 Programmer's Guide

This manual describes specific operating system modules which have been designed to provide an interface between Vector 4, 16-bit hardware (8088) and a 16-bit Operating System (CP/M-86).

These modules are unique to the Vector 4 hardware environment and must be used when creating or modifying 16-bit software. They include:

-Device Drivers
-Video and Console Input

As a system programmer you can refer to this information along with the "Standard" BDOS Functions (found in CP/M-86 Systems Guide) and BIOS Jump Routines (found in CP/M-86 Systems Guide).[1] These sources should provide you with all the system information you need to design and create 16-bit programs.

The Vector CP/M-86 Programmer's Guide also describes a Vector 4 CP/M Simulator which can be used to run 8-bit programs *within* a 16-bit operating system.

1.5 System and Programmer's Disk

The Vector 4 comes with a System and Programmer's Disk. Both these disks have files which are described in different types of documentation which comes with your Vector 4. The following chart gives the file names, file description and the specific documentation which describes those files.

KEY

U1= Vector 4 CP/M User's Guide
U2= Vector 4 CP/M-86 User's Guide
U3= Digital Research's CP/M-86 Operating System—User's Guide

T1= Vector 4 Technical Information Manual
T2= Disktest User's Manual

P1= Vector 4 CP/M Programmer's Guide
P2= Vector's SCOPE Reference Manual
P3= Vector's ZSM Assembler For CP/M Manual
P4= Digital Research's CP/M Dynamic Debugging Tool User's Guide
P5= Vector 4 CP/M-86 Programmer's Guide
P6= Digital Research's CP/M-86 Operating System—Programmer's Guide
P7= Digital Research's CP/M-86 Operating System—System Guide
P8= Microsoft BASIC Interpreter Reference Manual

[1] Vector does not support Table Generation using GENDEF or Bootstrap Adaptation Procedures (Pages 72 to 86).

System Disk

File Name	Documentation Location	File Description
(User 0)		
UTILITY.CMD	U2	Utility to FORMAT, GENSYS,.....
CONFIG.CMD	U2	System Configuration Program
SERIAL.DEV	U2	Serial Printer Driver
PARALLEL.DEV	U2	Parallel Printer Driver
NEC.DEV	U2	NEC Printer Driver
PIP.CMD	U2, U3	Peripheral Interchange Program
STORE.COM	U1, U2	Multi-Floppy Harddisk Storage Program
RESTORE.COM	U1, U2	Multi-Floppy Harddisk Restore Program
STAT.CMD	U2, U3	File and disk status utility
HELP.CMD	U2, U3	Help utility
HELP.HLP	U2	Data file for Help Utility
ED.CMD	U2, U3,	CP/M-86 Text Editor
DDT86.CMD	U3, P6	CP/M Debugger
ASM86.CMD	P6	8086 Assembler
GENCMD.CMD	U3, P7	CMD file generation utility
RUN8.CMD	U2, P5	Vector 4 CP/M Simulator
TOD.CMD	U3	Display and set time of day utility
SUBMIT.CMD	U3	Batch processing utility
(User 1)		
PRINTEST.COM	U1, U2, T1	Printer Test Program
MEMTEST.CMD	U1, U2, T1	Memory Test Program
KYBDTEST.COM	U1, U2, T1	Keyboard Test Program
SCRNTEST.COM	U1, U2, T1	Screen Test Program
PORTEST.COM	U1, U2, T1	Port Test Program
CPUTEST.COM	U1, U2, T1	Central Processing Unit Test Program
CRC.COM	U1, U2, T1	Cyclic Redundancy Check
DISKTEST.COM	U1, U2, T1, T2	Disk Test Program
RECLAIM.COM	U1, U2	Disk check and re-write program

Programmer's Disk

File Name	Documentation Location	File Description
(User 0)		
ED.CMD	U2, U3	CP/M-86 Text Editor
SC.COM	P2	SCOPE program and text editor
ASM86.CMD	U3, P6	8086 Assembler
ZSM.COM	P3	8080 Assembler
GENCMD.CMD	U3, P7	CMD file generation utility
LMCMD.CMD	P7	CMD file generation utility
LOAD.COM	P1	COM file generation utility
DDT86.CMD	U3, P6	CP/M Debugger
SERIAL.DEV	U2	Serial Printer Driver
SERIAL.A86	P5	Serial Printer Driver source code[1]
PARALLEL.DEV	U2	Parallel Printer Driver
PARALLEL.A86	P5	Parallel Printer Driver source code[1]
NEC.DEV	U2	NEC Printer Driver
RANDOM.A86	P5, P7	Sample A86 program using BDOS calls
DUMP.ASM	P1, P3	Sample ASM program using BDOS calls
DUMP.COM	P1	Program DUMP Utility
MBASIC5.COM	P8	Microsoft BASIC
EDIR.COM	U1, P1	Extended DIR Utility
ERAX.COM	U1, P1	Extended ERA Utility
SORTDIR.COM	U1, P1	Sort DIR Utility
ACOM.DEV	U2	Asynchronous Communications Driver
ACOM.A86	P5	Asynchronous Communications Driver Source

1 The reference (documentation location) describes how to write drivers.

SECTION II - VIDEO DRIVER AND CONSOLE INPUT MODULE

2.1 Overview of Video Driver and Console Input Module

The Operating System Independent Video Driver/Console Input Module is designed to provide all necessary operating system video/console functions within a standard module of code. The first responsibility of the module is to support a full set of programming commands that control a memory mapped video screen, needed to satisfy the console output requirement. The second responsibility of the module is to handle all keyboard input, conversion and buffering functions needed to satisfy the console input requirement.

Certain Console Functions will not properly pass through the escape character value "1BH". In these cases a "9BH" may be substituted for a "1BH". All requests for information that are made using a "9BH" will return information using the lead in control code of "9BH".

2.2 Video Driver

The video driver will operate in one of two modes. The first mode recognizes a subset of the ANSI 3.64 - 1979 Standards specification. This subset has been implemented within the practical and functional limitations of the Vector 4 computer system. The second mode (compatible mode), recognizes the Vector 4 CP/M video driver functions.

The mode control command is recognized regardless of the current mode.

VIDEO DRIVER ANSI 3.64 COMMANDS

The following control sequences will produce the indicated action. In all cases the entire sequence must be sent sequentially (without intervening characters). If an improper character is detected, within the control sequence, then the entire sequence is aborted (i.e. reset to STATE1) and the improper character is not printed. However, subsequent printable characters will be printed until an "ESC" is encountered.

Parameter Description Table

1. The parameters n and f# must consist of ASCII digits 1 thru 9 (30H-39H).
2. Two or more digit, decimal parameters (e.g. 37, 187) are sent as two or more consecutive ASCII digits.
3. If more than one parameter required then the character ";" (3BH) is included between them.
4. The maximum value for a parameter is 255.

5. If n equals zero or is not present, the default value of one (31H) is used. If f# is omitted the default value is 30H.

Whenever an attempt is made to move the cursor beyond the physical limit of the screen, the cursor will be stopped at the last possible position (i.e. just before leaving the screen). All lines are numbered beginning with one at the top of the screen. The column numbers also begin with one at the left side of the screen.

The physical screen size is defined by two parameters: Lines and columns. A provision is included for up to 136 horizontal and 40 vertical tab stops. Attempted cursor movement to any non-existing tab position (or positions beyond the screen limit) results in movement to the appropriate edge of the screen. Upon screen initialization horizontal tab stops are set at every eighth position (default). There are no default settings for vertical tab stops.

CURSOR CONTROL COMMANDS

"ESC","[",n,"A"	CUU	Cursor Up - Move cursor upward n lines. An attempt to move the cursor above the top of the screen results in the cursor positioned at the same column of the first line.
"ESC","[",n,"B"	CUD	Cursor Down - Move cursor downward n lines. An attempt to move the cursor beyond the bottom of the screen results in the cursor positioned at the same column of the last line.
"ESC","[",n,"C"	CUF	Cursor Forward - Move cursor forward n positions. An attempt to move the cursor beyond the right side of the screen results in the cursor stopping at the last column of the current line.
"ESC","[",n,"D"	CUB	Cursor Backward - Move cursor backward n positions. An attempt to move the cursor beyond the left side of the screen results in the cursor stopping at the first column of the current line.
"ESC","[",n,"E"	CNL	Cursor Next Line - Move cursor to the first column of the n-th following line. An attempt to move the cursor beyond the bottom of the screen results in the cursor stopping at the first column of the last line.

"ESC","[",n,"F"	CPL	Cursor Preceding Line - Move cursor to the first column of the n-th preceding line. An attempt to move the cursor beyond the top of the screen results in the cursor stopping at the first column of the first line.
"ESC","[",n,"G"	CHA	Cursor Horizontal Absolute - Move cursor to absolute horizontal position n. An attempt to move the cursor beyond the right side of the screen results in the cursor stopping at the last column of the current line.
"ESC","[",v,";",h,"H"	CUP	Cursor Position - Move cursor to position specified by v (vertical) and h (horizontal): Defaults- v,h = 1. An attempt to move the cursor beyond the right side of the screen results in the cursor stopping at the last column. An attempt to move the cursor beyond the bottom of the screen results in the cursor stopping at the last line.
"ESC","[",n,"I"	CHT	Cursor Horizontal Tab - Advance Cursor to the n-th following horizontal tab stop (set by CTC,HTS, or default). An attempt to move the cursor beyond the right side of the screen or to a non-existing tab stop results in the cursor stopping at the last column of the current line.
"ESC","[","6","n"	CPR	Cursor Position Report - Returns the current cursor position into the keyboard buffer with the following sequence : "ESC","[",v,";",h,"R" where v = vertical position, h = horizontal.
"ESC","[",n,"Y"	CVT	Cursor Vertical Tab - Advance cursor to the n-th following vertical tab stop (set by CTC or VTS). An attempt to move the cursor beyond the bottom of the screen or to a non-existent tab stop results in the cursor stopping at the same column of the last line.
"ESC","[",n,"Z"	CBT	Cursor Backward Tab - Move cursor backward to the n-th preceeding horizontal tab. An attempt to move the cursor beyond the left side of the screen or to a non-existing tab stop results in the cursor stopping at the first column of the current line.
"ESC","[",n,"`"	HPA	Horizontal Position Absolute - Same as CHA.

"ESC","[",n,"a"	HPR	Horizontal Position Relative - Same as CUF.
"ESC","[",n,"d"	VPA	Vertical Position Absolute - Move cursor to absolute line n with no horizontal movement. An attempt to move the cursor to a non-existent screen position results in the cursor stopping at the last line of the same column.
"ESC","[",n,"e"	VPR	Vertical Position Relative - Same as CUD.
"ESC","[",v,";",h,"f"	HVP	Horizontal and Vertical Position - Moves cursor to the specified vertical (v) and horizontal (h) position (same as CUP).
"ESC","D"	IND	Index - Move cursor down one line with no horizontal movement. An attempt to move the cursor beyond the last line results in the cursor stopping at the same column of the last line.
"ESC","E"	NEL	Next line - Move cursor to start of the following line. An attempt to move the cursor beyond the last line results in the cursor stopping at the start of the last line.
"ESC","M"	RI	Reverse index - Move cursor up one line. An attempt to move the cursor beyond the first line results in the cursor stopping at the same column of the first line.

SCREEN SCROLLING COMMANDS

"ESC","[",n,"S"	SU	Scroll Up - Move entire screen contents up n lines. All lines moved off the top of the screen are permanently erased while n erased lines are inserted at the bottom of the screen.

FORMAT EFFECTOR COMMANDS (TABULATION / SPECIAL AREAS)

"ESC","[",f#,"g"	TBC	Tabulation Clear - Clears tab stops according to f#:

30 Clear horizontal tab at cursor (default).

31 Clear vertical tab at current line.

		33 Clear all horizontal tab stops.
		34 Clear all vertical tab stops.
"ESC","[",f#,"W"	CTC	Cursor Tab Control - Sets or clears tab stops according to f# .
		30 Set horizontal tab stop at the current cursor position (default).
		31 Set vertical tab stop at the current line.
		32 Clear horizontal tab stop at the current cursor position.
		33 Clear vertical tab stop at the current cursor position.
		35 Clear all horizontal tab stops.
		36 Clear all vertical tab stops.
"ESC","H"	HTS	Horizontal Tab Set - Set horizontal tab stop at current position.
"ESC","J"	VTS	Vertical Tab Set - Sets vertical tab at active line.

VISUAL DISPLAY EDITING

"ESC","[",n,"@"	ICH	Insert Characters - Insert n spaces from the current cursor position. Existing characters are shifted away from cursor (forward or backward depending on HEM (defined on page 2-10). Characters that are shifted off the screen are lost.
"ESC","[",f#,"J"	ED	Erase Display - Erases some or all of the displayed characters depending on f#:
		30 From cursor position to end inclusive.
		31 From start to cursor position.
		32 All of display.

"ESC","[",f#,"K"	EL	Erase In Line - Erases some or all of the characters in the current line depending on f#:
		30 From cursor position to end inclusive.
		31 From start to cursor position inclusive.
		32 All of the line inclusive.
"ESC","[",n,"L"	IL	Insert Line - Inserts n blank lines at the current cursor line. Previous lines shifted up. All lines shifted beyond the first line are permanently lost.
"ESC","[",n,"M"	DL	Delete Line - Deletes n lines from the current cursor line inclusive. Adjacent lines are shifted upward toward current line. Erased lines (n) are added to the bottom of the display.
"ESC","[",n,"P"	DCH	Delete Characters - Deletes n characters from the current cursor inclusive, adjacent characters are shifted toward cursor (forward or backward depends on HEM (defined on page 2-10). Erased characters (n) are added to the appropriate end of the shifted characters.
"ESC","[",n,"X"	ECH	Erase Characters - Erases n characters after the current cursor position inclusive. Cursor does not move.
"ESC","[",f#,"m"	SGR	Select Graphic Rendition - Selects graphic mode according to function # :
		30 Normal Video (light on dark background)
		37 Reverse Video (dark on light background)
		31";"30 Standard Character Set (default)
		31";"31 Alternate Character Set
		31";"32 Foreign Video Conversion Enabled

31";"33	Foreign Video Conversion Disabled
31";"34	Keyboard Standard (Vector 3 compatible)
31";"35	Keyboard Physical (No conversion)
31";"36	Keyboard Foreign Conversion Enabled
31";"37	Keyboard Foreign Conversion Disabled

MISCELLANEOUS FUNCTIONS

"ESC","[",n,"b"	REP	Repeat - The previous single character is repeated n times.
"ESC","c"	RIS	Reset to Initial State - Reset screen to the initial condition (Resets modes, keyboard reassignment buffer and sets horizontal tab stops every 8 positions) note: This command has no visible effect on the screen.
"ESC","[",n,22H, "chars",22H,"p"	KRA	Keyboard Reassignment - (non ANSI standard implementation). Allows the user to reassign the keyboard hex codes to another representation in the form of one or more ASCII characters (including command strings as interpreted here). The first numeric parameter is the value which is replaced (by the subsequent parameters) when it appears in the console input buffer. The replacement values may be either numeric parameters as previously discussed or an ASCII string beginning and ending with the character 22H (") or any combination of both. Example:

```
 1B 5B  31 39 33   22 STAT *.* 0D  22 70
esc  [  ---n----    " -string----   "  p
```

Whenever the F1 key is depressed the CP/M-86 command STAT *.* will be executed automatically. Note: The carriage return must be included if you want the command string to be executed. When adding a new reassignment, the following sequence is returned in the input buffer:

		"ESC","[","0","w" =successful "ESC","[","1","w" =not successful (buffer full)
"ESC","[",n1,";",n2,"r"	SSC	Set Screen Columns - (non ANSI standard) Allows the user to define an active screen which is within the first and last physical screen columns: n1 = starting column, n2 = ending column.
"ESC","[",n1,";",n2,"q"	SSL	Set Screen Lines - (non ANSI standard) Allows the user to define an active screen which is within the first and last physical screen lines: n1 = starting line, n2 = ending line.
"ESC","[","x"	RRS	Return Reassignment Space - returns the following sequence: "ESC","[",n1,";",n2,"[" where: n1 = size in bytes of the reassignment buffer and n2 = # of empty reassignment buffer bytes.
"ESC","[",f#,"chars","o"	SSD	Set Status Display - (non ANSI standard) Allows the user to print a message on the bottom (25th) line (cols. 1-70) and turn the clock display (cols. 71-78) on/off according to f#: 0 Clock off - Normal Video 1 Clock on - Normal Video 2 Clock off - Reverse Video 3 Clock on - Reverse Video

MODE MODIFICATION COMMANDS

"ESC","[",f#,"s"	SCM	Set Video Command Mode - Sets command interpretation for independent type commands (i.e. "ESC",command) depending on f#. This mode eliminates conflict between the ANSI standard commands with that format and the Vector ESC sequences. 30 ANSI standard mode for ESC commands. 31 Vector compatible mode for ESC commands.

"ESC","[","t"	RSM	Return System Mode - Current system mode is returned with the following format into the input buffer: "ESC","[",n0,n1,n2,n3,n4,n5, "y" where:	
	n0:	30	List device mode 0 - ignore list output
		31	List device mode 1 - Printer driver #1
		32	List device mode 2 - Printer driver #2
		33	List device mode 3 - Output to video driver
	n1:	30	Standard Character Set
		31	Alternate Character Set (foreign)
	n2:	30	Standard (V3) Keyboard Conversion
		31	Physical Keyboard (no conversion)
	n3:	30	Foreign Keyboard Conversion Disabled
		31	" " " Enabled
	n4:	30	Foreign Video Conversion Disabled
		31	" " " Enabled
	n5:	30	ANSI Standard command mode
		31	Vector compatible command mode
"ESC","[",f#,"z"	SLD	Set List Device - according to f#:	
		30	Mode 0 - Ignore List Output
		31	Mode 1 - Printer Driver #1
		32	Mode 2 - Printer Driver #2
		33	Mode 3 - List Output To Video Driver
		Returns the following sequence into the input buffer: "ESC","[",n,"u"	
		30	Successful
		31	List Device Busy
		32	Driver Not Connected
"ESC","[","v"	RRT	Reset Reassignment Table - Clears all entries in the reassignment table so that no reassignments are in effect.	
"ESC","[",f#,"h"	SM	Set Mode - Sets mode according to f#:	
"ESC","[",f#,"l"	RM	Reset Mode - Resets mode according to f#:	

The following modes are set or reset by SM and RM:

f#		
34	IRM	Insertion/Replacement Mode
		set - characters at cursor are shifted left or right (depending on HEM) and new characters inserted.
		reset- characters at cursor are replaced by new characters (default).
31,";",30	HEM	Horizontal Editing Mode
		set - causes DCH ,ICH and IRM to effect characters preceeding the cursor.
		reset- causes DCH ,ICH and IRM to affect characters after the cursor (default).
32,";",30	LNM	Line Feed New Line Mode -
		set - causes line feed (LF) to move cursor to beginning of next line.
		reset- causes line feed (LF) to move cursor down one line with no horizontal movement (default).

The following control sequences are also defined in ANSI 3.64 but are not included in this implementation:

APC	Application Program Command
CCH	Cancel Character
CRM	Control Representation Mode
DA	Device Attributes
DAQ	Define Area Qualification
DCS	Device Control String
DMI	Disable Manual Input
DSR	Device Status Report
EA	Erase in Area
EBM	Editing Boundary Mode

EF	Erase in Field
EMI	Enable Manual Input
EPA	End Protected Area
ESA	End Selected Area
FEAM	Format Effector Action Mode
FETM	Format Effector Transfer Mode
FNT	Font Selection
GATM	Guarded Area Transfer Mode
GSM	Graphic Size Modification
GSS	Graphic Size Selection
HTJ	Horizontal Tab with Justify
JFY	Justify
KAM	Keyboard Action Mode
MATM	Multiple Area Transfer Mode
MC	Media Copy
MW	Message Waiting
NP	Next Page
OSC	Operating System Command
PLD	Partial Line Down
PLU	Partial Line Up
PM	Privacy Message
PP	Preceding Page
PU1	Privacy Use One
PU2	Privacy Use Two
PUM	Positioning Unit Mode
QUAD	Quad
SATM	Area Transfer Mode
SD	Scroll Down
SEM	Select Editing Extent Mode
SL	Scroll Left
SPA	Start Protected Area
SPI	Spacing Increment
SR	Scroll Right
SRM	Send-Receive Mode
SRTM	Status Report Transfer Mode
SSA	Start Selected Area
SS2	Single Shift 2
SS3	Single Shift 3
ST	String Terminator
STS	Set Transmit State
TSS	Thin Space Specification
TTM	Transfer Termination Mode
VEM	Vertical Editing Mode

VIDEO DRIVER VECTOR COMPATIBLE COMMANDS

A (01H)	Initialize	- Initializes screen
B (02H)	Home	- Home cursor
D (04H)	Clearscreen	- Clear screen and home cursor
G (07H)	Bell	- Audible bell
H (08H)	Backspace	- Backspace
I (09H)	Tab	- Tab
J (0AH)	Linefeed	- Linefeed
M (0DH)	Carriage ret	- Carriage Return
N (0EH)	Toggle	- Toggle reverse video cursor
P (10H)	Clrscreen	- Clear to end of screen
Q (11H)	Clrline	- Clear to end of line
R (12H)	Linefeed	- Same as CRTL J
T (14H)	Toggle	- Toggle reverse video character
U (15H)	Upcursor	- Move cursor up
W (17H)	Curleft	- Move cursor left
X (18H)	Clrstart	- Clear to start of line
Z (1AH)	Curight	- Move cursor right
ESC x y	Curpos	- x,y cursor positioning
ESC ^ A	SCS	- Standard Character Set
ESC ^ B	ACS	- Alternate Character Set
ESC ^ C	KPM	- Keyboard Physical Mode
ESC ^ D	KSM	- Keyboard Standard Mode
ESC ^ E	KFM	- Keyboard Foreign Mode
ESC ^ F	GME	- Graphics Mode Enabled
ESC ^ G	GMD	- Graphics Mode Disabled
ESC ^ H	FVCE	- Foreign Video Conversion Enabled
ESC ^ I	FVCD	- Foreign Video Conversion Disabled

FOREIGN VIDEO CONVERSION

When the video driver is in the Foreign Video Conversion Enabled mode, the following conversions take place between the code sent to the video driver and the code displayed in the screen.

ACTUAL CODE	CONVERTED VIDEO CODE
40	16
5B	17
5C	18
5D	19
5E	1A
60	1B
7B	1C
7C	1D
7D	1E
7E	1F

2.3 Console Input

The console input routine is responsible for handling the physical interface with the keyboard. It performs the actual retrieval, buffering and conversion functions. The character input and buffer fill functions are performed as part of the interrupt service procedure. The character is buffered in its raw physical state. All character conversions are done on the character when it is read from the buffer.

USER PROGRAMMABLE KEYCODE TRANSLATION

Another responsiblity of the console input function is to handle the keycode translation function on the input side. Translation is defined as the conversion of one input code to a different output code or number of output codes. This is done by searching through the translation table for a matching input code. If a matching input code is found, the output codes are sent through the console channel instead of the original input code. The original implementation uses a 512 byte table to store the original code, number of output translation codes and actual output translation code or codes. The output string may not exceed 255 codes in length. The table is maintained so that no two identical input codes exist concurrently. If a command is received specifying a translation with an input code that already exists in the table, the existing entry is replaced with the new entry. If the translation is specified with a length of zero the matching entry would be removed with no new entry added. The number of entries in the table is dependent on the length of each entry. The sum of the lengths may not exceed the total table length.

FOREIGN KEYBOARD CONVERSION

When the video driver is in the Keyboard Foreign Mode, the following conversions take place between the physical code returned by the keyboard and the logical code returned by the system. This function is implemented by adding these conversions to the keyboard conversion table. If one of the physical codes required by this function was mapped before this function was enabled, the old mapping of that physical code will be destroyed.

After this function has been enabled, the physical codes may be selectively mapped to other logical values.

PHYSICAL	LOGICAL
B1	40
B2	5B
B3	5C
B4	5D
B5	5E
B6	60
B7	7B
B8	7C
B9	7D
FF	7E

SECTION III - DEVICE DRIVERS

3.1 Overview of Device Drivers

The following description of the printer/communications interface is intended for CP/M-86. Each printer/communications driver is a stand alone program structured as a CP/M-86 8080 model where the code, data, stack and extra segment all have the same base address. All data storage and code must be within the printer driver. All segment registers must be returned to original value on exiting. All other registers may be modified. Two printer drivers and one communications driver are supported within each system. The initial implementation will statically preallocate 3K of memory for each printer driver and 2K of memory for the communications driver.

Control is transferred to specific functions within the printer/communications device using a jump table that begins at the start of the segment. Immediately following the jump table is a table of parameters to be used within the printer/communications driver.

If the function is returned incomplete, all required input values should be preserved for a subsequent recall.

NOTE: Examples of device drivers (Communication and Printer) exist on the "system" disk which came with your Vector 4. The "Config" Program (described in User's Guide) is used to integrate the drivers into your system configuration.

3.2 Printer Driver Entry Point and Data Block Descriptions

The following entry points are located at the beginning of the driver. The entry points are used to pass control to the various routines within the driver.

```
driver+0      JMP     INITIALIZE
driver+3      JMP     STATUS
driver+6      JMP     DATA_OUTPUT
driver+9      JMP     DATA_INPUT
.
.       future expansion of table
.
```

Data storage begins at driver + 24

driver+24	DRIVER NAME	byte * 16
driver+40	DRIVER CL/TYPE	byte
driver+41	DRIVER INT.DES	byte
driver+42	SERIAL INIT	byte
driver+43	FORMLENGTH	byte
driver+44	TOP MARGIN	byte
driver+45	LAST LINE	byte
driver+46	AUTOPAGE	byte

DESCRIPTION OF ENTRY POINTS

INITIALIZE - This routine should initialize the communicating hardware. Serial drivers should be initialized according to the data byte describing baud rate, parity, etc. For parallel devices, any special sequences should be sent, clearing ports. The connected device, when applicable, should be set to a known state. Initialization of special functions, within the printer, should also be done at this time (e.g. horizontal, vertical tab settings, etc.) Reset any line counting functions supported by the driver. Also any character buffering done by the driver should be cleared.

```
Required information - none
Returned information - (AL) completion status
                       (DX) printer condition information
```

STATUS - This routine should return general printer I/O status.

```
Required information - none
Returned information - (AL) completion status
                       (DX) printer condition information
```

DATA_OUTPUT - This routine sends a data character to the printer. An incomplete status exists until the printer has successfully transmitted the character or a fault has occurred.

```
Required information - (BL) character to be printed
Returned information - (AL) completion status
```

DATA_INPUT - This routine receives a data character from the printer. An incomplete status exists until a data character has successfully been received from the printer or a fault has occurred.

```
Required information - none
Returned information - (AH) character (if successful)
                       (AL) completion status
```

DESCRIPTION OF DATA STORAGE

DRIVER NAME - Text description identifying driver. The name should be left justified in a field size of 16, padded with spaces.

DRIVER CL/TYPE - Byte value identifying class and type of driver.

There are two bytes located at the front of all Vector 4 CP/M printer driver modules that provide information about the printer and its interface to the system. The first byte, located at DVR+40, contains the printer class and type codes. The printer class code is contained in the upper nibble (bits 4-7) while the printer type code is contained in the lower nibble (bits 0-3).

```
Bit 7---> CCCCTTTT <--- Bit 0
```

The assigned printer class codes are:

```
0000 =Undefined
0001 =Draft Printer (Line Printers)
0010 =Word Processing Printer (Letter Quality)
0011-1111 =Undefined
```

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

Type Code	Class 1	Class 2
0000	No Printer	No Printer
0001	Standard Serial (TI 810)	Qume (Parallel)
0010	Centronics	NEC (Parallel)
0011	Epson	Serial (ETX/ACK or XON/XOFF)
0100-1110	Undefined	Undefined
1111	Custom	Custom

DRIVER INT.DES - Byte value identifying interface characteristics.

The second byte, located at DVR+41, contains the Interface Descriptor byte. It is composed of four separate fields;

```
Bit 7 ---> HHH U PPP I <--- Bit 0
```

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

000	=I/O 2 Board
001	=ZCB Board
010	=Vector 4 SBC
011	=I/O 2 or ZCB Serial
100-111	=Undefined

Bit 4 (U) is an undefined field.

Bits 1 through 3 (PPP) contain the Protocol Code. The following values have been assigned:

000	=Hardware handshaking
001	=ETX/ACK Protocol
010	=XON/XOFF Protocol
011-111	=Undefined

Bit 0 (I) is the interface type bit. If this bit is set (1) then the interface is serial. If this bit is reset (0) then the interface is parallel.

SERIAL INIT - Byte value specifying serial communications parameters as defined below.

```
7         6         5         4         3         2         1         0

----baudrate-----             --parity-    stopbit    -wordlength-

  000 = 110                   x0 = none     0 = 1      10 = 7 bits
  001 = 150                   01 = odd      1 = 2      11 = 8 bits
  010 = 300                   11 = even
  011 = 600
  100 = 1200
  101 = 2400
  110 = 4800
  111 = 9600
```

FORMLENGTH - Byte value specifying number of physical lines per page.

TOP MARGIN - Byte value specifying number of blank lines in top margin.

LAST LINE - Byte value specifying last line printed on page.

AUTOPAGE - Byte value specifying auto page activity (NZ = active).

PRINTER CONDITION INFORMATION

The printer condition information is both general purpose and device dependent. The following section will describe the functions and organization of the general purpose bits. If the bit is set the condition exists.

bit 0	printer is busy
bit 1	printer is inoperative
bit 2	printer is offline
bit 3	I/O error
bit 4	input character available
bit 5	output buffer is full
bit 6	paper is out
bit 7	ribbon is out
bit 8	timeout
bit 9-15	device dependent information

COMPLETION STATUS

A completion status exists for each function. The status is checked on return and the routine is recalled if the function is marked incomplete. This requirement is necessary in order to maintain control within the operating system for real time required scheduling functions. When the operation is complete, the completion bit should be cleared along with the corresponding successful/not successful setting of the status bit. The function of the individual bits are defined as follows:

bit 0	function incomplete
bit 1	function not successful
bit 2	reserved

3.3 Communication Driver Entry Point and Data Block Descriptions

The entry points are located at the beginning of the driver. The entry points are used to pass control to the various routines within the driver.

```
driver+0     JMP    INITIALIZE
driver+3     JMP    STATUS
driver+6     JMP    DATA_OUTPUT
driver+9     JMP    DATA_INPUT
.
.      future expansion of table
.
```

Data storage is located at driver start + 24

```
driver+24    DRIVER NAME    byte * 16
driver+40    SERIAL INIT    byte
```

DESCRIPTION OF ENTRY POINTS

INITIALIZE - Initialize the communicating hardware. Initialize any serial ports using parameters specified in SERIAL INIT data byte. Initialize the connected device when applicable to a known state. Some communications devices may require a sequence of characters to be sent to prepare them for use. Clear any buffers that are used.

```
Required information - none
Returned information - (AL) completion status
                       (DX) communication condition information
```

STATUS - This routine should return the general I/O status of the communications channel.

```
Required information - none
Returned information - (AL) completion status
                       (DX) communication condition information
```

DATA_OUTPUT - This routine transmits a data character to the device. An incomplete status exists until the data character has successfully been transmitted or a fault has occurred in the hardware.

```
Required information - (BL) data to be transmitted
Returned information - (AL) completion status
```

DATA_INPUT - This routine receives a data character from the communicating device. An incomplete status exists until a data character has successfully been received or a fault has occurred in the hardware.

```
Required information - none
Returned information - (AH) character (if successful)
                       (AL) completion status
```

DESCRIPTION OF DATA STORAGE

DRIVER NAME - 16 characters for name left justified and padded with spaces.
SERIAL INIT - byte where each bit is described below

```
 7       6       5       4       3       2       1       0

 ----baudrate-----       --parity-    stopbit   -wordlength-

   000 = 110             x0 = none     0 = 1     10 = 7 bits
   001 = 150             01 = odd      1 = 2     11 = 8 bits
   010 = 300             11 = even
   011 = 600
   100 = 1200
   101 = 2400
   110 = 4800
   111 = 9600
```

COMMUNICATION CONDITION INFORMATION

The communications condition information specifies both the physical state of the communications port and any attached modem hardware. The function of the individual bits are defined as follows:

bit 0	delta clear to send
bit 1	delta data set ready
bit 2	trailing edge ring detector
bit 3	delta receive line signal detect
bit 4	clear to send
bit 5	data set ready
bit 6	ring indicator
bit 7	received line signal detect
bit 8	data ready
bit 9	overrun error
bit 10	parity error
bit 11	framing error
bit 12	break detect
bit 13	xmit holding register empty
bit 14	xmit shift register empty
bit 15	time out

COMPLETION STATUS

A completion status exists for each function. The status is checked on return and the routine is recalled if the function is marked incomplete. This requirement is necessary in order to maintain control within the operating system for real time required scheduling functions. When the operation is complete the completion bit should be cleared along with the corresponding successful/not successful setting of the status bit.
The function of the individual bits are defined as follows:

bit 0	function incomplete
bit 1	function not successful
bit 2	reserved

3.4 I/O Function Remapping

The I/O byte implementation will correspond as close as possible to the original INTEL standard of mapping logical to physical devices. The IOBYTE function creates a mapping of logical to physical devices which can be altered during CP/M-86 processing. The byte is broken up into 4 fields of 2 bits each. The following is a description of each field:

CONSOLE - The principal interactive console which communicates with the operator, accessed through CONST, CONIN and CONOUT. In this implementation, the default console is the memory mapped video screen and associated keyboard on the Vector 4 system.

LIST - The principal listing device. This implementation supports two online list device drivers. The current-list-device is the driver that is used for all list output and input. The current-list-device may be established through the use of a video driver command, extended function or command within the CONFIG utility.

AUXILIARY
OUTPUT - In this implementation, the AUXO is considered the communications output channel. The actual device driver is user definable and may be loaded interactively by the user.

AUXILIARY
INPUT - In this implementation, the AUXI is considered the communications input channel. The actual device driver is user definable and may be loaded interactively by the user.

I/O BYTE ORGANIZATION (ALL VALUES ARE IN BITS)

#	OFFSET	LENGTH	DESCRIPTION
1	0	2	CONSOLE mapping
2	2	2	AUXILIARY OUTPUT mapping (to communication device)
3	4	2	AUXILIARY INPUT mapping (from communication device)
4	6	2	LIST mapping

The following is a chart showing the route of information for the console and list logical device mapping. All mapping is done from a single logical to physical level. Any device mapping for the AUX0 and AUXI is not recognized by this implementation.

```
CONSOLE FIELD = 0 -- TTY ----------------------------+
              = 1 -- CRT -------------------------+  |
              = 2 -- BATCH --------------------+  |  |
              = 3 -- COMM ------------------+  |  |  |
                                            |  |  |  |
LIST FIELD    = 0 -- MONITOR ------------+  |  |  |  |
              = 1 -- CRT -------------+  |  |  |  |  |
              = 2 -- LPT ----------+  |  |  |  |  |  |
              = 3 -- COMM ------+  |  |  |  |  |  |  |
                                |  |  |  |  |  |  |  |
                                |  |  |  |  |  |  |  |
INPUT CHARACTER FROM LIST-------|  |  |  |  |  |  |  +
OUTPUT CHARACTER TO LIST--------|  +  |  +  |  +  |  +
                                |  |  |  |  |  |  |  |
INPUT CHARACTER FROM CONSOLE----|  |  |  |  +  +  +  +
OUTPUT CHARACTER TO CONSOLE-----|  |  +  +  +  +  +  |
                                |  |  |  |  |  |  |  |
INPUT CHARACTER FROM COMM.------|  |  |  |  +  +  |  |
OUTPUT CHARACTER TO COMM.-------+  |  |  |  +  |  |  |
```

SECTION IV - VECTOR 4 CP/M SIMULATOR

4.1 Overview of Vector 4 CP/M Simulator

The Vector 4 CP/M simulator is designed to allow eight bit CP/M programs to be run under the CP/M-86 Operating System. The simulator performs this task by converting eight bit operating system calls to sixteen bit operating system calls. The actual eight bit code is executed by the Z80 in the Vector 4. When the eight bit code makes an operating system request, the parameters are passed to the sixteen bit system by the pseudo BDOS contained in the simulator. Control is then passed to the sixteen bit system where the function is performed. The simulator also contains a pseudo BIOS to convert eight bit BIOS calls to sixteen bit BIOS calls.

Vector 4 CP/M contains many functions not available in CP/M-86. These functions are performed in the simulator. The simulator is designed to execute eight bit software in a manner as true to Vector 4 CP/M as possible. This does not mean that any CP/M program will run under the simulator. Only CP/M programs which make conventional interfaces to the operating system will execute properly (using same amount of memory in same environment). This is the same constraint that has existed with Vector 4 CP/M. No interrupt support is provided.

4.2 Vector 4 CP/M Simulator Operation

The simulator exists on CP/M-86 as a program file named RUN8. The command format for executing the simulator is as follows:[1]

```
A>RUN8 [program-name [ argument [ argument...] ] ]
```

Where :

```
     program-name : the name of a CP/M .COM program file excluding
                    .COM extension.
         argument : optional arguments separated by spaces.
```

The simulator will then be loaded from the currently logged logical disk and will be passed to the remainder of the command line. The simulator will then process the remaining command line into a program name and associated arguments. The simulator will then load the eight bit program into memory and place the remaining arguments in their respective locations. The simulator then passes control to the Z80 processor and jumps to location 100H, the normal eight bit CP/M entry point.

[1] NOTE: No spooling features will be supported initially.

The Z80 executes until a BDOS or BIOS call is performed. At that time, control is passed to the sixteen bit operating system and the function is performed. After the function is complete, the 8088 processor returns control to the Z80 and the eight bit program continues. When the eight bit program terminates, control is passed to the command processor of the sixteen bit system and the process is complete. The file system used for the simulator is that of the sixteen bit system. This means that the default drive, and logical drive layout will appear identical to both the eight bit system and the sixteen bit system. Also, all files from each system will be available to the other.

4.3 Simulator Implementation

The following sections describe each Vector 4 CP/M BDOS call and how it is simulated by the Vector 4 CP/M simulator program. The section after the "Entry Parameters" may refer to Appendix A. This appendix is found in the Vector 4 CP/M Programmer's Guide.

The "Simulator Action" Section contains the reference "See Description Above". This refers to the paragraph(s) which summarizes this function. This paragraph(s) is located under the "Entry Parameters" and/or the "Returned Values" subtitles.

4.3.1 BDOS Functions 0-15

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, or entering BYE from the CCP.

Simulator Action:

The simulator will process function 0 (SYSTEM RESET) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above). This call when completed will transfer control back to the host operating system CCP.

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 **[RETURN]**, line feed, and backspace, **[DEL]**, or **[CTRL H]**, are echoed to the console. Tab characters, **[CTRL I]**, are expanded in columns of eight characters. A check is made to terminate process **[CTRL C]**, start/stop scroll **[CTRL S]** and start/stop printer echo **[CTRL P]**.

The BDOS does not return to the calling program until a character has been typed, thus suspending execution if a character is not ready.

Simulator Action:

The simulator will process function 1 (CONSOLE INPUT) by making the corresponding host system BIOS calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 video driver, then to the console device. Similar to function 1, tabs are expanded and checks are made for terminate process, start/stop scroll and printer echo. Graphics or non-standard characters (such as foreign character sets) should be called through this function, rather than directly through the video driver PROM.

Simulator Action:

The simulator will process function 2 (CONSOLE OUTPUT) by making the corresponding host system BIOS calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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.

Simulator Action:

The simulator will process function 3 (READER INPUT) by making the corresponding host system BIOS calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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.

Simulator Action:

The simulator will process function 4 (PUNCH OUTPUT) by making the corresponding host system BIOS calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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.

Simulator Action:

The simulator will process function 5 (LIST OUTPUT) by making the corresponding host system BIOS calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

FUNCTION 6: DIRECT CONSOLE I/O

Entry Parameters:	Register C - ØH6H 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 special applications where simple 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 suspends 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.

An important difference between Function 6 and Function 1 (Console Input) is that Function 6 does not echo the character to the console. It is possible, using Function 6, to get the character, evaluate it against your program parameters, and throw it out if it does not meet the parameters or print it if it does.

Simulator Action:

The simulator will process function 6 (DIRECT CONSOLE I/O) by making the corresponding host system BIOS calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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.

Simulator Action:

The simulator will maintain a pseudo I/O byte. This is required because of the differences between the I/O byte in Vector 4 CP/M and the standard I/O byte configuration contained in CP/M-86. The simulator will implement the I/O functions of the I/O byte with a combination of BIOS calls Vector 4 System Function Calls.

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.

Simulator Action:

The simulator maintains a local pseudo I/O byte. This function causes the local I/O byte to be updated depending on the passed parameters. See function 7 (Get I/O Byte) for further information.

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 delimiter.

Simulator Action:

The simulator will process function 9 (PRINT STRING) by making the corresponding host system BIOS calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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, or when a **[RETURN]** is entered from the keyboard. 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 BDOS 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] Warm-Starts 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.

Simulator Action:

The simulator will simulate the actions performed by function 10 (Read Console Buffer). The simulator will input characters one at a time and perform all editing and control functions listed above. All register conventions and parameter passage conventions utilized in Vector 4 CP/M will be maintained.

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.

Simulator Action:

The simulator will process function 11 (GET CONSOLE STATUS) by making the corresponding host system BIOS calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 Graphic's 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.

Simulator Action :

The simulator will return the value '25H' in the L register regardless of the version of CP/M-86 it is executing under.

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 ($$$.SUB) exists 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).

Simulator Action:

The simulator will process function 13 (RESET DISK SYSTEM) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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, 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 codes)

Simulator Action:

The simulator will process function 14 (SELECT DISK) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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.

On systems operating in the 3.Ø mode, the following information applies as well:

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 provided that they do not access the same record (if that record was locked), and that they also open the file in the unlocked mode. Opening files in the Read-Only mode allows only read operations as well as access for other tasks to the file provided that they open it in the same mode.

A successfully 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.

In both 2.5 and 3.Ø modes, 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 successful, or ØFFH (255 decimal) if the file cannot be found. Register H is set to zero in both cases; 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 (3.Ø ONLY)**
Ø9	**- '?' IN FILENAME OR EXTENSION FIELD**
1Ø	**- PROCESS OPEN FILE LIMIT EXCEEDED (3.Ø ONLY)**
11	**- NO ROOM IN THE SYSTEM LOCK LIST (3.Ø ONLY)**

(See Appendix A, from the Vector 4 CP/M Programmers Guide, for a description of the error messages)

Simulator Action:

The simulator will process function 15 (OPEN FILE) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above). When the simulator is running under CP/M-86, file record locking will not be supported.

4.3.2 BDOS Functions 16-27

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 opposite 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.

In addition, on systems running in the 3.Ø mode, the interface attribute f5' is used to specify 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 (in the 3.Ø mode) 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 in either mode 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. Files opened by systems in the 3.Ø mode should always be closed for optimal system operations in systems using 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

Simulator Action:

The simulator will process function 16 (CLOSE FILE) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above). When running under CP/M-86, the record and file locking features are not implemented.

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 Error 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 'fl' 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 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
Ø4 - SELECT ERROR

Simulator Action:

The simulator will process function 17 (SEARCH FOR FIRST) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 occurrence 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

Simulator Action:

The simulator will process function 18 (SEARCH FOR NEXT) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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.

In the 3.Ø mode, it is permissible to delete files if:

the file is not open by any user;

the file has been opened in the LOCKED mode by the user requesting the deletion;

the file has been opened UNLOCKED by only the user requesting the deletion.

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

Simulator Action:

The simulator will process function 18 (DELETE FILE) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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

Simulator Action:

The simulator will process function 20 (READ SEQUENTIAL) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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

Simulator Action:

The simulator will process function 21 (WRITE SEQUENTIAL) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 BDOS creates the file and initializes the FCB in memory and disk as an empty file; if the file already exists, register H will return error Ø8 (FILE ALREADY EXISTS). Vector Graphic has changed the original CP/M function to prevent creation of duplicate files. The FCB is entered to the File Open list (if the system is running in the 3.Ø mode). The MAKE function has the side-effect of activating the FCB; 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 function.

This function returns ØFFH in register A if there were no directory space. If the operation is 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

Simulator Action:

The simulator will process function 22 (MAKE FILE) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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. 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

Simulator Action:

The simulator will process function 23 (RENAME FILE) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

FUNCTION 24: RETURN LOGIN VECTOR

Entry Parameters:	Register C - 18H
Returned Value:	Register HL - Login Vector

The LOGIN VECTOR 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, a drive that has been accessed since the last warm/cold boot, 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.

Simulator Action:

The simulator will process function 24 (RETURN LOGIN VECTOR) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

FUNCTION 25: RETURN CURRENT DISK

Entry Parameters:	Register C - 19H
Returned Value:	Register A - Current Disk

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

Simulator Action:

The simulator will process function 25 (RETURN CURRENT DISK) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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. Many computer systems use 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.

Simulator Action:

The simulator will process function 26 (SET DMA ADDRESS) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

FUNCTION 27: GET ADDR (ALLOC)

Entry Parameters:	Register C - 1BH
Returned Value:	Register HL - ALLOC Address

The system maintains an "allocation table" in the main memory for each on-line disk drive (on-line here meaning that the disk has an associated driver). 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.

Simulator Action:

The simulator will, upon initialization, map a 2K block of Z-80 address space over the system tables. When simulator function 27 is called, the simulator will return the address of the actual ALLOC table in the Z-80 HL register. This will allow the Z-80 direct access to the actual tables.

4.3.3 BDOS Functions 28-42

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.

Simulator Action:

The simulator will process function 28 (WRITE PROTECT DISK) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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.

Simulator Action:

The simulator will process function 29 (GET READ/ONLY VECTOR) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 (tl' 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 numbers), while f2' through f4' are not presently used, but may be useful for application 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

Simulator Action:

The simulator will process function 30 (SET FILE ATTRIBUTES) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

FUNCTION 31: GET ADDR (DISK PARAMS)

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.

Simulator Action:

When the simulator initializes the Z-80 addressing map, it will place a 2K block over the actual system tables. This will allow the Z-80 to have direct access to the tables. The simulator will return in register HL a pointer to the DPB relative to the Z-80 address space.

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).

Simulator Action:

The simulator will process function 32 (SET/GET USER CODE) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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)
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.

Simulator Action:

The simulator will process function 33 (READ RANDOM) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 begin 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.

Simulator Action:

The simulator will process function 34 (WRITE RANDOM) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 65,535), 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

Simulator Action:

The simulator will process function 35 (COMPUTE FILE SIZE) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 tabulating 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.

Simulator Action:

The simulator will process function 36 (SET RANDOM RECORD) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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. The drive is reset as it appears to the user who placed the call, but not to the system or to other users.

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.

Simulator Action:

The simulator will process function 37 (RESET DRIVE) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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) with the exception that a previously unallocated data block is filled with zeroes before the record is written.

If this function has been used to create a file, records accessed by a READ RANDOM operation that contain all zeroes 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, from the Vector 4 CP/M Programmer's Guide, for a description of returned values).

Simulator Action:

The simulator will process function 40 (WRITE RANDOM WITH ZERO FILL) by making the corresponding host system BDOS call. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 r∅, 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.

∅1 - READ MODE: READING UNWRITTEN DATA
∅3 - CANNOT CLOSE CURRENT EXTENT
∅4 - SEEK TO UNWRITTEN EXTENT
∅6 - SEEK PAST PHYSICAL END OF DISK (RANDOM RECORD NUMBER OUT OF RANGE)
∅7 - RECORD DID NOT MATCH

NOTE: This function is the only function which returns an error code of 'Ø7' in register A indicating non-matching sectors.

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 or logical error is returned, the four LSBs of register H 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

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

Simulator Action:

The simulator performs the test and write function by reading the existing record(s), comparing them to the test record(s), and writing the new record(s) if a match is found.

FUNCTION 42: LOCK RECORD (3.Ø MODE ONLY)

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:

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)
Ø8 - RECORD LOCKED BY ANOTHER PROCESS
1Ø - 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

Simulator action:

The simulator, when running under CP/M-86, returns a '0H' in the A register and an 'FFH' in the C register. This means that recording locking does not exist when running the simulator under CP/M-86.

4.3.4 BDOS Functions 43-228

FUNCTION 43: UNLOCK RECORD (3.Ø MODE ONLY)

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 the address of an FCB in which the Random Record field is filled with the 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).

Although the UNLOCK RECORD function will not unlock a record that is currently locked by another process, no error is returned if a process attempts the process. 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.

The UNLOCK RECORD function sets register A to zero if the 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

Simulator Action:

When running the simulator under CP/M-86, the simulator will return a '0H' in the A register and a 'FFH' in the C register. This means that the Unlock function is not supported when running the simulator under CP/M-86.

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.

Simulator Action:

When running under CP/M-86, this function is implemented by setting a local variable to the Multi-Sector Count passed to the function in the E register. If the value passed in the E register is within the range of one to sixteen, the local variable is set to that value and a '0H' is returned in the A register. If the value passed in the E register is out of the range one to sixteen, the local variable is left unaltered, and a 'FFH' is returned in the A register. This local variable is then used to control repeated calls to the BDOS to simulate Multi-Sector I/O. The default value of this local variable is one.

FUNCTION 45: SET BDOS ERROR MODE

Entry Parameters:	Register C - 2DH
	Register E - BDOS Error mode:
	ØØ: Default
	Ø1 : Return
	Ø2 : Print & Return
	All others: DEFAULT

The SET BDOS ERROR MODE function will determine how physical and logical errors are handled for a process. The function can exist in three modes: the Default mode, the Return Error mode, and the Print and 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 Print and Return Error mode, the system message is displayed as in the Default mode. Unlike the Default mode, however, the program is not terminated after the user presses any key. Instead, the error code is returned to the program as in the Return Error mode.

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 ØFFH (255 decimal), the error mode is set to Return Error mode; if set to any other value, the error mode is set to the default mode.

Simulator Action:

The simulator maintains a local variable containing the BDOS error mode. The set BDOS error mode alters the value of this local variable. If the function is called with the E register containing an 'FEH' then the local variable is set to the display and return mode. If the function is called with the E register containing an 'FFH', then the local variable will be set to the return error mode. Any other value in the E register when the function is called, will cause the error mode to be set the default mode (display and reset system).

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 compatibility, 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

Simulator Action:

The simulator will simulate function 46 (RETURN FREE DISK SPACE) by examining the Allocation Vector for the associated drive and calculating the free space. The simulator will obtain the address of the Allocation Vector by calling CP/M-86 BDOS function 27 (get Allocation Vector address) after selecting the requested drive. After the function is complete, the simulator will re-select the previously selected drive.

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.

Simulator Action:

The simulator will simulate the Chain to Program Function in the following manner. If the string passed to the function does not have an extension, the simulator will create a FCB containing the string with a .COM extension. The simulator will then attempt to find the .COM file. If the file is found, the simulator will then load the file at Z-80 location '100H'. The simulator will then initialize the Z-80 as if the program had been loaded by the CCP. The simulator will then execute the program at location '100H'.

If the string passed to the function can not be turned into an existing .COM file, or the string already contains an extension other than .COM, then the string will be passed in it's original form to CP/M-86 BDOS function 47 (Chain CCP function). At that time, the CP/M-86 CCP will process the string.

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.

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.

Simulator Action:

When running under CP/M-86, the flush buffer function will will return a '0H' in the A register and a 'FFH' in the C register. This means that when the simulator is running under CP/M-86, the flush buffer function is not supported.

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 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 > 3 characters, characters beyond 3 ignored. Extension has (*), all remaining bytes filled with (?).

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 not a delimiter, it will return the address of the delimiting blank or tab. 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.

Note: the FCB is first cleared out, so that any field not found is skipped. If the string to parse is a terminator (Ø), then the FCB would come back cleared, with zero fields zeroed and ASCII fields filled with spaces.

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.

Simulator Action :

The simulator will process the input string following the same procedures used in the Vector 4 CP/M operating system.

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. This function must be preceded by function 250 (SELECT LIST DEVICE) in order to select the list device.

Simulator Action:

The simulator will process function 158 (ATTACH LIST) on a local basis when running under CP/M-86. The simulator, under CP/M-86, is only concerned with despooler list device contention problems. The simulator will function as above if a contention exists. All parameter passage conventions are as in Vector 4 CP/M.

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. This function must be preceded by Function 250 (SELECT LIST DEVICE) in order to select the list device to be detached.

Simulator Action:

The simulator, running under CP/M-86, will process function 159 (DETACH LIST) on a local basis. The only contention under CP/M-86 is that of the despooler. The simulator maintains a local variable containing the currently selected list device and it's attached status. This function will allow the program to give up control of the list device so that another process may have access to it.

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 (if any) from the calling program and attach the newly specified list device passed in Register E. 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:

Ø	No List Device
1	Echo to Console
2	Logical List Device 1
3	Logical List Device 2

This function selects and attaches the specified list device. It is not necessary to select the list device through function 250 prior to running this function.

Simulator Action:

The simulator will process function 160 (SET LIST) on a local basis. The simulator maintains a local variable containing the currently selected list device. This function will cause that variable to be updated with the requested list device. If the device is currently attached to the despooler, this function will wait for the device to become available. (See description above for further information.)

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. As with functions 158 and 159, this requires function 250 (Select List Device) before running.

Simulator Action:

The simulator will process function 161 (CONDITIONAL ATTACH LIST) on a local basis. The simulator maintains a local variable containing the currently selected list device. Function 161 will attempt to attach the device. This function, unlike function 158 (Attach List), will return even if the selected list device is in use. All parameter passage is as in Vector 4 CP/M. (See Description Above).

FUNCTION 164: GET LIST NUMBER

Entry Parameters:	Register C - A3H
Returned Value:	Register HL - List Number

The GET LIST NUMBER function will return the value of the specified list device assignment of the calling program. Returned values will be in the following format:

Registers	**Values**
A (= L)	LD field from IOBYTE
B (= H)	Select status

Simulator Action:

The simulator processes function 164 (GET LIST NUMBER) on a local basis. The simulator maintains a variable containing the currently selected list device. This list device number will be returned in the HL register pair. (See above description for further information.)

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

Simulator Action:

The simulator will not support function 217 (GET/SET CONFIG BYTE). This function was never supported in the Vector 4 CP/M operating system. The call will return a '0H' in the A register and a 'FFH' in the C register.

FUNCTION 218: RETURN CURRENT CURSOR POSITION

Simulator Action:

The simulator will return the current cursor position by making an extended system function call to the video driver with a cursor position report request ("ESC","[","R"). The simulator will then convert the returned parameters from decimal to binary and return them in the HL register pair.

FUNCTION 222: OUTPUT TONE

Simulator Action:

The simulator will not support function 222 (OUTPUT TONE). The call will return with the registers unaltered.

FUNCTION 223: RETURN TONE GENERATOR STATUS

Simulator Action:

The simulator will not support function 223 (RETURN TONE GENERATOR STATUS). The call will return with a '0H' in the a register and a '0FFH' in the C register.

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.

Simulator Function:

The simulator will process function 224 (DETECT R-O STATUS) by making an Extended System Function Call to return such status. All parameter passage conventions from Vector 4 CP/M are maintained. (See Description Above)

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. Register L returns a BDOS error code related to a specific BDOS error.

NOTE: This function must be executed immediately after an error. Any intervening function call will clear the BIOS error code to Ø.

Simulator Action:

The simulator will process function 225 (RETURN BIOS ERROR CODE) by calling the Extended System Function Call to return that information. All parameter passage conventions are as in Vector 4 CP/M. (See Description Above)

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.

NOTE: This function was implemented by Vector Graphic prior to function 45 (SET BDOS ERROR MODE). Function 226 is maintained for historical compatibility; it is strongly recommended that programmers use Function 45 instead. Function 226 toggles between BDOS error modes Ø (default) and 1 (return error), with no provision for selecting BDOS error mode 2 (print and return error).

Simulator Action:

The simulator sets the local BDOS error mode variable depending on the E register. If the E register contains an 'FFH' then the variable is set to reflect the default error mode (report error and return to CCP). If the E register contains an '0H' then the variable is set to reflect the return error mode.

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.

Simulator Function:

The simulator will process function 227 (INHIBIT/ENABLE BIOS ERRORS) by making an Extended System Function Call. All parameter passage conventions are as in Vector 4 CP/M (See Description Above).

FUNCTION 228: GET/SET ACCOUNT CODE

Entry Parameters:	Register C - ØE4H
	Register HL - Account Code
	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.

Simulator Action:

The simulator maintains a local variable for current account code. This variable is initialized to zero in the simulator initialization routine. When BDOS function 228 (GET/SET USER CODE) is called with an 'FFH' in the E register, the current contents of the account code variable is returned in the A register. If any other value is passed in the E register, then the simulator will replace the contents of the account code variable with the value stored in the E register.

4.3.5 BDOS Functions 233-255

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.

Simulator Action:

When running the simulator under CP/M-86, function 233 (RELEASE TIME SLICE) will not be implemented. If this function is called when running the simulator under CP/M-86, the call will return with the Z-80 registers unaltered.

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.

Simulator Action:

The Simulator does not support BDOS function 234 (SET DMA TASK). This call is very specific to the Vector 4 CP/M Multi Bank Systems and can not be supported under sixteen bit systems. If a call to this function is performed, the simulator will display an error message and return to the host system CCP.

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.

Simulator Action:

The simulator will simulate the Chain CCP function in the following manner. If the string passed to the function does not have an extension, the simulator will create an FCB containing the string with a .COM extension. The simulator will then attempt to find the .COM file. If the file is found, the simulator will then load the file at Z-80 location '100H'. The simulator will then initialize the Z-80 as if the program had been loaded by the CCP. The simulator will then execute the program at location '100H'.

If the string passed to the function can not be turned into an existing .COM file, or the string already contains an extension other than .COM, then the string will be passed in it's original form to CP/M-86 BDOS function 47 (Chain CCP function). At that time, the CP/M-86 CCP will process the string.

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.

Simulator Action:

The simulator will process function 236 (RETURN OUTPUT STATUS) by making an Extended System Function Call to return the list device status. Parameter passage conventions are as in Vector 4 CP/M (See Description Above).

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.

Simulator Action:

The simulator will process function 237 (RETURN INPUT STATUS) by making an Extended System Function Call to return the input status. Parameter passage conventions are as in Vector 4 CP/M (See Description Above).

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.

Simulator Action:

The simulator will process function 238 (LIST INPUT) by making an Extended System Function Call to return the list input character. Parameter passage convention are as in Vector 4 CP/M (See Description Above).

FUNCTION 239: RETURN PRINTER TYPE

Entry Parameters:	Register C - ØEFH Register E - ØØH - Current List Device Ø1H - 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, Ø1H or Ø2H in Register E, depending on which list device type is wanted.

Simulator Action:

The simulator will process function 239 (RETURN PRINTER TYPE) by making the corresponding Extended System Function Calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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.

Simulator Action:

The simulator will process function 240 (INITIALIZE PRINTER) by making the corresponding Extended System Function Calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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.

Simulator Action:

The simulator does not support function 241 (ENABLE/DISABLE CIRCULAR BUFFERS). The simulator executes with input buffering always enabled. If function 241 is called, the simulator will return with the registers unaltered.

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.

Simulator Action:

The simulator implements function 242 (ENABLE/DISABLE KEYBOARD CONVERSIONS) by using the capabilities of the Video Driver. If the E register contains '0H', the Video Driver will be called to Disable all keyboard conversions. This means that the simulator will return physical keyboard codes. If the E register contains any other value, the Video Driver will be called to Enable all keyboard conversions currently programmed. This means that the simulator will return logical keyboard codes and strings.

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.

Simulator Action:

The simulator will process function 243 (ENABLE/DISABLE AUTO PAGING) by making the corresponding Extended System Function Calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

FUNCTION 244: RETURN REVISION LEVEL

Entry Parameters:	Register C - ØF4H
Returned Value:	Register H - System Code (hardware environment)
	H = Ø - Vector 3, 5 Series
	H = 1 - Vector 3E, 5E Series
	H = 2 - Vector 4 Series
	Register L - Revision Level

The RETURN REVISION LEVEL function will return the revision level of CP/M being used, as well as the type of system using it. The function will determine if certain BDOS features are installed. Use this along with Function 12 (RETURN VERSION NUMBER) to determine actual version, revision level, and mode (2.5 or 3.Ø).

Simulator Action:

When function 244 (RETURN REVISION LEVEL) is called, the simulator will return in the H register an '06H' and in the L register the revision level of the simulator.

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 to the 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.

Simulator Action:

The simulator will not support function 245 (ENABLE/DISABLE CROSS BANK CALLS). If function 245 is called, the simulator will issue an error message and return to the host system CCP. What this means is that if a program alters it's pseudo BIOS, all calls to the BIOS by the user program will be affected. Any calls made by the simulator, however, will be performed using the host system BIOS.

FUNCTION 246: CHANGE BDOS BASE ADDRESS

Entry Parameters:	Register C - ØF6H
	Register E - Page Address

This function provides a means of lowering the base address of the BDOS. This is useful for programs that want to intercept system calls, to perform additional tasks (plotting, asynchronous communications, etc.), and yet not be disturbed by a warm boot, or the end of execution of a program. Note that both the default pseudo-BDOS and the relocated BDOS will exist; that is, the space taken up by the default pseudo-BDOS at the top of memory will not be cleared for program access. There will be a 'window' of free, 'protected' RAM between the top of the relocated pseudo-BDOS and the base of the default BDOS.
Function 246 will adjust both the BDOS vector at location 5 and the pseudo-BDOS located at the top of the TPA to the address specified in E (i.e., if E register equals ØCEH, the vector at location 5 will change to JMP CEØ6H; and the pseudo-BDOS will be copied 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 is initiated, the BDOS will return to its default location.

NOTE: Versions of Vector 4 CP/M prior to Release 7 (2.56 and earlier) use a slightly different procedure. For these systems, the new page address is passed in Register D, and Register E must be reset to Ø.

Simulator Action:

The simulator will process function 246 (CHANGE BDOS BASE ADDRESS) by moving it's pseudo BDOS and reflecting the change at location 5 (Z-80 relative). Parameter passage conventions are as in Vector 4 CP/M (See Description Above).

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 fairly accurate time delays, and these delays need to be adjusted according to the clock speed of the processor.

Simulator Action:

The simulator will process a function 247 (RETURN MICROPROCESSOR CLOCK SPEED) by returning an '04H' in the A register. This value reflects the speed of the Z-80 processor and not the sixteen bit 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 DE 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.

Simulator Action:

The simulator will process function 248 (READ ACTUAL RECORD) by making the corresponding Extended System Function Calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

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 function is the complement 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 function 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 a record 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 written 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.

Simulator Action:

The simulator will process function 249 (WRITE ACTUAL RECORD) by making the corresponding Extended System Function Calls required to perform the function. The registers will be passed and returned in the same manner used in Vector 4 CP/M (See Description Above).

FUNCTION 250: SELECT LIST DEVICE

Entry Parameters:	Register E - List Device
Returned Values:	None

This function selects the desired list device, which can now be used by other attach or conditional attach functions (see functions 158, 159, 161, 164). Possible values for Register E are listed with Function 160 (SET LIST).

Simulator Action:

The simulator maintains a local variable containing the currently selected list device. This variable is initialized by the simulator to zero (Discard Output). When function 250 (SELECT LIST DEVICE) is processed by the simulator, the value passed in the E register is stored as the currently selected list device.

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.

Simulator Action:

When the simulator is executing under CP/M-86, function 255 (RELEASE FILE RESOURCES) is not supported. If function 255 is called, the simulator will return with the registers unaltered.

4.3.6 BDOS Function Chart

FUNCT. NUMBER	FUNCT. NAME	INPUT PARAMETERS[1]	OUTPUT RESULTS[1]
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[2]
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

[1] See Def = See BDOS Definition within text.
Ret Code = Returned Code
Dir Code = Directory Code
Err Code = Error Code

[2] A = L, and B = H upon return.

FUNCT. NUMBER	FUNCT. NAME	INPUT PARAMETERS[1]	OUTPUT RESULTS[1]
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 Vect[2]
25	RETURN CURRENT DISK	None	A = Cur Disk[2]
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[2]
3Ø	SET FILE ATTRIBUTES	DE = FCB	See Def
31	GET ADDR (Disk Params)	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
42	LOCK RECORD	See Def	See Def
43	UNLOCK RECORD	See Def	See Def
44	SET MULTI-SECTOR COUNT	See Def	See Def

[1] See Def = See BDOS Definition within text.
Ret Code = Returned Code
Dir Code = Directory Code
Err Code = Error Code

[2] A = L, and B = H upon return.

FUNCT. NUMBER	FUNCT. NAME	INPUT PARAMETERS[1]	OUTPUT RESULTS[1]
45	SET BDOS ERROR MODE	See Def	None
46	RETURN FREE DISK SPACE	See Def	See Def
47	CHAIN TO PROGRAM	None	None
48	FLUSH BUFFERS	See Def	See Def
152	PARSE FILENAME	See Def	See Def
158	ATTACH LIST	None	None
159	DETACH LIST	None	None
160	SET LIST	See Def	See Def
161	CONDITIONAL ATTACH LIST	None	A = Ret Code
164	GET LIST NUMBER	None	A = List Number
217	GET/SET CONFIG BYTE	See Def	See Def
224	DETECT R/O STATUS	None	A = Boolean Value
225	RETURN BIOS ERROR CODE	None	HL = Err Code
226	INHIBIT/ENABLE BDOS ERRORS	See Def	None
227	INHIBIT/ENABLE BIOS ERRORS	See Def	None
228	GET/SET ACCOUNT CODE	See Def	See Def
233	RELEASE TIME SLICE	None	None

[1] See Def = See BDOS Definition within text.
Ret Code = Returned Code
Dir Code = Directory Code
Err Code = Error Code

FUNCT. NUMBER	FUNCT. NAME	INPUT PARAMETERS[1]	OUTPUT RESULTS[1]
234	SET DMA TASK	See Def	None
235	CHAIN CCP COMMAND	See Def	None
236	RETURN OUTPUT STATUS	None	A = Boolean Value
237	RETURN INPUT STATUS	None	A = Boolean Value
238	LIST INPUT	See Def	See Def
239	RETURN PRINTER TYPE	See Def	See Def
24∅	INITIALIZE PRINTER	None	None
241	ENABLE/DISABLE CIRC BUFFERS	See Def	See Def
242	ENABLE/DISABLE KEY CONVERS	See Def	See Def
243	ENABLE/DISABLE AUTO-PAGING	See Def	See Def
244	RETURN REVISION LEVEL	None	HL = Rev Level
245	ENABLE/DISABLE CROSS-BANK CALLS	See Def	None
246	CHANGE BDOS BASE ADDRESS	See Def	None
247	RETURN MICROPROCESSOR CLOCK SPEED	C = ∅F7H	A = Clock Speed
248	READ ACTUAL RECORD	See Def	See Def
249	WRITE ACTUAL RECORD	See Def	See Def
255	RELEASE FILE RESOURCES	None	None

[1] See Def = See BDOS Definition within text.
Ret Code = Returned Code
Dir Code = Directory Code
Err Code = Error Code